Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari

Diagnosis and Identification of Discrete Event Systems using Petri Nets

Maria Paola Cabasino

Advisors: Alessandro Giua
Carla Seatzu

Curriculum: ING-INF/04 Automatic Control

XXI Cycle
March 2009

Ph.D. in Electronic and Computer Engineering
Dept. of Electrical and Electronic Engineering
University of Cagliari

Diagnosis and Identification of Discrete Event Systems using Petri Nets

Maria Paola Cabasino

Advisors: Alessandro Giua
Carla Seatzu

Curriculum: ING-INF/04 Automatic Control

XXI Cycle
March 2009

Dedicated to my parents

Acknowledgements

I would like to thank all the people that in these three years supported me in many ways, making possible this work.

First and foremost I want to thank Alessandro Giua and Carla Seatzu, my supervisors, for their support and their encouragement. They dedicated to me a lot of time and guided me during my PhD. I felt so lucky to work with them, not only for their deep knowledge and expertise in the field of Petri nets, but because they are so beautiful people. This thesis is the result of the work done together in these three years.

Special thanks to Stéphane Lafortune, Christoforos Hadjicostis and Manuel Silva for their ospitality during my stay in their group and for the great opportunity to work with them. I would like to thank Cristian Mahulea, Laura Recalde, Yu Ru, Stefano Lai and Davide Nessi for their fruitful collaboration.

I would like to thank Joerg Raisch, Stéphane Lafortune and Christoforos Hadjicostis for reviewing this thesis.

Finally, I would like to thank all my family. In particular my parents Francesca and Franco, and my brother Alessandro, that strongly supported me during these three years, helping me to always go ahead in the best way. I really could not have done most of the things I have enjoyed doing. Thanks again. Last but not least, all my friends that are really a lot. They always supported me during this period and gave me the right strength to live in happiness these three years.

Diagnosi e identificazione di sistemi ad eventi discreti mediante reti di Petri

Riassunto

In questa tesi viene affrontato il problema della diagnosi (parte II) e dell'identificazione (parte III) di sistemi ad eventi discreti mediante l'uso delle reti di Petri.

Il problema della diagnosi consiste nell'individuare se e quando si verifica un comportamento anomalo nel sistema considerato. L'approccio presentato in questa tesi utilizza come modello le reti di Petri discrete ipotizzando che alcune delle transizioni della rete siano non osservabili, incluse tutte quelle transizioni che modellano i guasti. Tale approccio è basato sul concetto di marcatura di base e di giustificazione, che ci permettono di caratterizzare l'insieme delle marcature che sono consistenti con l'osservazione corrente, e l'insieme delle transizioni non osservabili il cui scatto abilita l'osservazione. Questa procedura può essere applicata a tutte quelle reti la cui sottorete non osservabile è aciclica. Se la rete è anche limitata l'approccio proposto può essere notevolmente semplificato spostando molti dei calcoli della procedura off-line, grazie alla costruzione di un grafo che è chiamato grafo base di raggiungibilità. Tale approccio è applicabile anche a reti di Petri etichettate, dove cioè due o più transizioni possono avere la stessa etichetta, e dove alcune transizioni sono non osservabili. In questa tesi è affrontato anche il problema della diagnosticabilità per reti di Petri limitate. Un sistema è detto diagnosticabile se una volta che il guasto è accaduto siamo in grado di individuarlo in un tempo finito. Infine la nostra procedura di diagnosi è stata confrontata dal punto di vista dell'applicabilità e della complessità computazionale con una ben nota procedura di diagnosi per sistemi ad eventi discreti basata su automi.

Il problema dell'identificazione consiste nel determinare, dati in ingresso una coppia di segnali ingresso/uscita, un sistema che abbia come ingresso/uscita un segnale che approssima quanto meglio quello osservato. Nel caso delle rete di Petri il comportamento osservato è il linguaggio delle rete. In questa tesi verrà proposta una tecnica di identificazione che consiste nel determinare una rete di Petri che genera il linguaggio dato in ingresso. Dapprima abbiamo considerato il problema dell'identificazione di una rete di Petri free-labeled, ossia una rete di Petri dove a ogni transizione è associata un'etichetta diversa. L'insieme delle transizioni e il numero di posti della rete da identificare è dato, mentre la struttura della

rete e la marcatura iniziale sono calcolate risolvendo un problema di programmazione intera. Dopodiché abbiamo considerato diverse estensioni di questo approccio introducendo informazioni aggiuntive sul modello o sulla marcatura iniziale. Inoltre abbiamo risolto il problema di sintetizzare una rete di Petri limitata a partire da un automa che genera il suo linguaggio. Infine, abbiamo mostrato come tale approccio può essere generalizzato al caso delle reti etichettate. In entrambi i casi il problema di identificazione può essere risolto tramite un problema di programmazione intera. Per ridurre la complessità computazionale del problema di identificazione è stata proposta anche una procedura che utilizza la programmazione lineare. Un altro problema affrontato in questa tesi è il seguente: dato un automa che rappresenta il grafo di copertura di una rete di Petri, determinare un sistema di rete di Petri il cui grafo di copertura sia isomorfo a quello dell'automa dato. Infine abbiamo affrontato il problema dell'identificazione del modello dei guasti in una rete di Petri. In particolare, abbiamo ipotizzato di conoscere il sistema senza guasti e il nostro obiettivo è stato quello di identificare la struttura del sistema con i guasti. Dapprima abbiamo ipotizzato che il linguaggio del sistema con i guasti sia completamente noto. In seguito, abbiamo considerato che i guasti siano non osservabili.

Contents

List of Figures

List of Tables

Part I

Preliminary

Chapter 1

Introduction

Summary

In this chapter first we present an overview on the problem of diagnosis and identification and we give a motivation for these studies. Secondly we describe the organization of the thesis and the topics of each chapter. Finally, we discuss the contribution of this thesis.

1.1 Introduction

Petri nets (PNs) were first introduced in the early 1960s by Carl Adam Petri in his PhD dissertation [70]. Over the years they have been extended in many directions including time, data, and hierarchy. PNs are particularly useful for modeling concurrent, distributed, asynchronous behavior in a system, and offer a good trade-off between modeling power and analytical tractability. Nowadays they are considered as one of the main formalisms for modeling, analysis and control of *discrete event systems* (DES), together with automata.

Even if PNs have been extensively studied and used in many application areas, such as manufacturing, transportation and communication, there exist several problems that are still open. In this thesis we focus on two related problems, namely the *fault diagnosis* and *identification* of DES using PNs. These problems are described in detail below.

1.2 Diagnosis problem

Failure detection and isolation in industrial systems is a subject that has received a lot of attention in the past few decades. A failure is defined to be any deviation of a system from its normal or intended behavior. *Diagnosis* is the process of detecting an abnormality in the system behavior and isolating the cause or the source of this abnormality. Failures in industrial systems could arise from several sources such as design errors, equipment malfunctions, operator mistakes, and so on. Diagnosis of DES, such as complex automation systems, has become more difficult and cannot be performed manually based on empirical information. Systematic approaches for the diagnosis problem are urgently needed.

Failures are inevitable in today's complex industrial environment. As technology advances, as we continue to build systems of increasing size and functionality, and as we continue to place increasing demands on the performance of these systems, then so do we increase the complexity of these systems. Consequently (and unfortunately), we enhance the potential for systems to fail, and no matter how safe our designs are, how improved our quality control techniques are, and how better trained the operators are, system failures become unavoidable. Given the fact that failures are inevitable, the need for effective means of detecting them is quite apparent if we consider their consequences and impacts not just on the systems involved but on the society as a whole. Moreover we note that effective methods of failure diagnosis can not only help avoid the undesirable effects of failures, but can also enhance the operational goals of industries. Improved quality of performance, product integrity and reliability, and reduced cost of equipment maintenance and service are some major benefits that accurate diagnosis schemes can provide, especially for service and product oriented industries such as home and building environment control, office automation, automobile manufacturing, and semiconductor manufacturing. Thus, we see that accurate and timely methods of failure diagnosis can enhance the safety, reliability, availability, quality, and economy of industrial processes.

The need of automated mechanisms for the timely and accurate diagnosis of failures is well understood and appreciated both in industry and in academia. A great deal of research effort has been and is being spent in the design and development of automated diagnostic sys-

tems, and a variety of schemes, differing both in their theoretical framework and in their design and implementation philosophy, have been proposed. From the conceptual view-point most existing methods of failure diagnosis can be classified as (i) fault-tree based methods ([55, 56, 88, 89]); (ii) quantitative, analytical model-based methods ([42, 92, 87]); (iii) expert systems and other knowledge-based methods ([78]); (iv) model-based reasoning methods ([33, 34, 40]); and (v) DES based methods ([6, 7, 9, 12, 30, 35, 37, 39, 44, 46, 48, 51, 57, 59, 60, 65, 71, 74, 75, 76, 77, 81, 82, 93, 94]).

Diagnosis approaches can solve two different types of problems: the problem of diagnosis and the problem of diagnosability. These two problems are also known in literature as *diagnosis on-line* and *diagnosis off-line*, respectively, for their implementation standpoint. However for the sake of clarity (since in our approach to solve the problem of diagnosis we move some calculations off-line) in the rest of the thesis we refer to them as the *diagnosis* and the *diagnosability problems*.

Diagnosability implies the ability to locate a fault after a finite number of observations for *any* sequence (any behavior) of the system. Thus, to verify diagnosability one would need to verify the ability to locate a fault after a finite number of observations for (most-likely) an infinite number of behaviors; this is a challenging problem and, in the case of automata, it can be solved with the construction of the diagnoser (albeit at a high computational cost). On the other hand, solving a diagnosis problem means associate to each observed string of events a diagnosis state, such as “normal” or “faulty” or “uncertain”. It is performed on-line based on the observed sequence and it can be done relatively efficiently in the case of automata because if one knows the system model then one only needs to keep track of the possible pairs of states and faults at any instant in time. Of course, if one already has built a diagnoser, one can use the diagnoser for this purpose but that would not be advisable.

1.3 Identification problem

Identification is a classical problem in system theory: given a pair of observed input-output signals it consists in determining a system such that the input-output signals approximate the observed ones [86]. Different problems can be solved in this framework:

- *Model identification:* we want to find the model of an existing system from "external" measurements. In the literature two main problems are addressed and in general they are associated with time-driven systems. In the first case we know the input and the output of the system and we want to identify its internal structure. This problem is know as *black box*. In the second case we know not only the input and the output of the system but also we have some information about its structure, e.g. we know that the system is an electrical circuit containing a resistance and a capacitor in parallel but we do not know their value. This problem is known as *grey box*, because we have more information than the previous case.
- *Model synthesis:* we have examples of admissible/forbidden behaviors and want to design a system that satisfies these constraints ([1, 2, 3, 4, 10, 13, 28, 32, 38, 47, 49, 62, 63, 64, 67, 68, 80]). This problem is in general associated with DES.

- *Fault identification:* we want to find the model of a fault affecting a known system ([16]). This problem can be associated both with DES and time driven systems.

In this thesis we present two procedures to solve the second and the third problem mentioned above by solving an *integer programming* problem. Moreover we discuss another approach to solve the second problem using *linear programming* techniques.

When we deal with the problem of identification using PNs it is common to consider as observed behavior the language of the net, i.e., the set of transition sequences that can be fired starting from the initial marking. One can also assume that some partial information on the state is also known, however this is not the case that we present in this thesis.

1.4 Structure of the thesis

The thesis is divided into four parts. The first part, from Chapter 1 to Chapter 5, is an introductive part. The second part, from Chapter 6 to Chapter 9, is about the diagnosis of DES using PNs. The third part, from Chapter 10 to Chapter 13, is about the identification of DES using PNs. Finally, the forth part is dedicated to the conclusions.

In Chapter 2 we present a literature review based on the most important results found in diagnosis and identification of DES.

In Chapter 3 we give a background on PNs and we introduce some notation that we will use in the rest of the thesis.

In Chapter 4 we present one of the most known approach to the problem of failure diagnosis using automata ([76], [77]). The research group of the University of Michigan introduce a notion of diagnosability of DES in the framework of formal languages. Moreover, they present a systematic procedure for detection and isolation of failure events using diagnosers and provide necessary and sufficient conditions for a language to be diagnosable. The diagnoser performs diagnostics using on-line observations of the system behavior; it is also used to state and verify off-line necessary and sufficient conditions for diagnosability. These conditions are stated on the diagnoser.

Chapter 5 is dedicated to a brief description on the theory of regions. In particular, we propose two methods taken from the literature for synthesizing a PN starting from a labeled graph ([45]) and starting from a given language ([64]).

The second part, dedicated to diagnosis of PNs, starts with Chapter 6 where is presented a fault detection approach for DES using PNs. We assume that some of the transitions of the net are unobservable, including all those transitions that model faulty behaviors. Our diagnosis approach is based on the notion of *basis marking and justification*, that allow us to characterize the set of markings that are consistent with the actual observation, and the set of unobservable transitions whose firing enable it. Four diagnosis states are defined, each one corresponding to a different degree of alarm. This approach applies to all net systems whose unobservable subnet is acyclic. If the net system is also bounded the proposed approach

may be significantly simplified moving the most burdensome part of the procedure off-line, thanks to the construction of a graph, called the *basis reachability graph*.

In Chapter 7 we focus on the diagnosis of labeled PNs, i.e., nets where two or more transitions may share the same label. In particular we present an approach for diagnosis. The proposed procedure is based on results on unlabeled PNs presented in Chapter 6 and allows us to also consider events that are indistinguishable, namely events that produce an output signal that is observable, but that is common to other events. Four diagnosis states are defined, each one corresponding to a different degree of alarm. A procedure is given to compute the actual diagnosis state given the current observation. We show that also in the case of labeled bounded PNs the most burdensome part of the procedure can be moved off-line defining a particular graph, that we call basis reachability graph.

In Chapter 8 we present an approach to solve the problem of diagnosability of bounded Petri net systems. In particular, we first give necessary and sufficient conditions for diagnosability. Then, we present a method to test diagnosability that is based on the analysis of two graphs that depend on the structure of the net, including the fault model, and the initial marking. The first graph is called *basis reachability diagnoser*, the second one is called *modified basis reachability graph*. At the end of the chapter a comparison between our diagnosis procedure and Diagnoser approach, presented in Chapter 4, is made.

The part dedicated to the diagnosis of PNs is concluded by Chapter 9. Here we consider two diagnosis procedures for DES based respectively on automata and Petri nets. The first procedure has been developed by the University of Michigan group and is presented in Chapter 4 while the second procedure is the one presented in this thesis in Chapter 7. We apply them to a diagnosis benchmark and compare them in terms of computational complexity.

The part dedicated to the identification of PNs starts with Chapter 10 where we deal with the problem of identifying a PN system, given a finite language generated by it. First we consider the problem of identifying a free labeled PN system, i.e., a Petri net in which all transition labels are distinct. The set of transitions and the number of places is assumed to be known, while the net structure and the initial marking are computed solving an integer programming problem. Then we extend this approach in several ways introducing additional information about the model (structural constraints, conservative components, stationary sequences) or about its initial marking. We also treat the problem of synthesizing a bounded net system starting from an automaton that generates its language. Moreover, we show how the approach can also be generalized to the case of labeled PNs, where two or more transitions may share the same label. In particular, in this case we impose that the resulting net system is deterministic. In both cases the identification problem can still be solved via an integer programming problem. Finally, we analyze the complexity of the identification approach in terms of computational time required to get an admissible solution, that may also be optimal according to a given performance criterion. In particular, we want to investigate how the computational time depends on the cardinality of the set of finite length strings that describe the language, and on the chosen performance index. To this aim we consider the language generated by a particular PN system that models a sender-receiver process. Different cases are examined with different number of places and transitions, and thus different languages generated. The numerical simulations we carried out enabled us to conclude that the computational time becomes prohibitive for languages that are described by a large

number of finite length strings, if we want to determine a solution that is *optimal* with respect to a given performance index. On the contrary, computational times are negligible if we limit to consider any *admissible* net system, e.g., the first admissible solution computed by CPLEX when solving the optimization problem. We believe that this is not a drawback of our procedure because in effect, when solving identification problems like this, the main requirement is that of determining an *admissible* solution, not necessarily an optimal one.

In Chapter 11 we solve the following problem: given an automaton that represents the coverability graph of a PN, determine a PN system whose coverability graph is isomorphic to the automaton. The proposed approach requires solving an integer programming problem whose set of unknowns contains the elements of the pre and post incidence matrices and the initial marking of the net.

In Chapter 12 we show how to tackle the problem presented in Chapter 10 using linear programming techniques, thus significantly reducing the complexity of finding a solution. However, such kind of solution cannot be optimized with respect to an objective function, as in Chapters 10, 11. The procedure we propose identifies a net whose number of places is equal to the cardinality of the set of disabling constraints. We provide a criterion to check if the computed solution has a minimal number of places, and, if this is not the case, we discuss two approaches to reduce this number. At the end of the chapter there is a comparison between our identification procedure and theory of regions approach (presented in Chapter 5).

The identification part is concluded by Chapter 13. It is well known that most of the fault identification problems in the DES literature assume knowledge of the structure of the net system, including the nature (and behavior) of the possible faults. In this chapter we deal with this problem within the framework of PNs by removing the requirement that the nature (and behavior) of the fault is known. In particular, we devise a way to identify the structure of the faulty transitions of the system given its language. Then, we generalize this procedure to unobservable faults, in which case the structure of the faulty system needs to be recognized from the knowledge of the structure of the fault-free system, and the projection of the faulty system language on the set of non-faulty events, that are assumed to be observable.

In the forth and last section of this thesis conclusions for diagnosis and identification of Petri nets are drawn.

1.5 Contributions of the thesis

The contributions of the thesis are all collected in the second and third part and may be summarized in the following items.

- A fault detection approach for DES using unlabeled PNs, where some transitions are unobservable, based on the notion of basis marking and justification is given in Chapter 6 (publication on the topic [21]).
- An extension of the fault diagnosis approach to labeled PNs is presented in Chapter 7 (publication on the topic [24]).

- Necessary and sufficient conditions for diagnosability of bounded PNs are provided in Chapter 8 where a method to testify the diagnosability of the system is also given (publication on the topic [23]).
- A comparison in terms of computational complexity between our diagnostic procedure and the diagnostic procedure presented in Chapter 4 is presented in Chapter 9 (publication on the topic [54]).
- An identification procedure based on integer programming for free-labeled and labeled PNs is given in Chapter 10 (publications on the topic [18], [20]). Moreover in the same chapter is presented an analysis on the computational complexity of this identification approach (publication on the topic [17]).
- A procedure to identify a PN starting from its coverability graph is presented in Chapter 11 (publications on the topic [19], [25]).
- A procedure based on linear programming to identify a Petri net given a finite prefix of its language is given in Chapter 12 (publication on the topic [22]).
- A way to identify the structure of the faulty transitions of a faulty system given the language of the fault-free system is presented in Chapter 13 (publication on the topic [16]).

Chapter 2

Literature review

Summary

Fault diagnosis and identification of systems have received considerable attention in the last decades. In this Chapter we present a survey of the state-of-the art of these topics within the framework of discrete event systems.

2.1 State of art for diagnosis

The diagnosis of discrete event systems (DES) is a research area that has received a lot of attention in the last years and has been motivated by the practical need of ensuring the correct and safe functioning of large complex systems.

Fault-tree based methods

The most widely used scheme for alarm analysis, especially in the process control industry, is based on *fault trees* [55, 56, 88, 89]. Fault trees provide a graphical representation of cause-effect relationships of faults in a system. Starting from a goal violation, or, a system failure event that is indicated by an alarm condition, a fault tree is built by reasoning backwards from the system failure to basic or primal failures that represent the root cause of the failure. The main drawback of this approach is that fault trees require a great deal of effort in their construction. Moreover they pose difficulties in handling feedback systems.

Analytical redundancy methods

A vast majority of the approaches to failure diagnosis proposed in the control systems literature are based on *analytical redundancy* (see [42, 92, 87]). The analytical redundancy method, addressed for continuous systems, can roughly be divided into two major steps: (i) generation of residuals and (ii) decision and fault isolation. The residual generation process typically involves generating residual signals by comparing predicted values of system variables (from mathematical models of the system) with the actual observed values. These signals are nominally near zero and get accentuated when failures do occur. In the decision and fault isolation stage, the residuals are examined for the likelihood of faults. A major advantage of this approach is the ability to detect, not only abrupt faults (or hard failures) but also slowly developing (or incipient) faults via trend analysis. The primary drawbacks of this approach are the computational expenditure for the detailed on-line modeling of the process, and more importantly, the sensitivity of the detection process with respect to modeling errors and measurement noise. The issue of robust failure detection using analytical models has been and is being investigated in detail.

Expert systems

Failure diagnosis by *expert systems* is an approach eminently suited for systems that are difficult to model, i.e., systems involving subtle and complicated interactions (among and within components) whose outcomes are hard to predict (see [78] and references therein). The chief drawback of expert systems is that a considerable amount of time may elapse before enough knowledge is accumulated to develop the necessary set of heuristic rules for reliable diagnosis, coupled with the fact that this approach is very domain dependent, i.e., expert systems are not easily portable from one system to another. Further, it is difficult to validate an expert system.

Model-based reasoning methods

Another approach to failure diagnosis that has been investigated in the artificial intelligence (AI) literature is that of *model-based reasoning* ([33, 34, 40]). The fundamental paradigm of this approach, much like the analytical redundancy methods, is that of observation and prediction. These model-based methods employ a general purpose model of the structure and behaviour of the system which are constructed using standard AI technology such as predicate logic, frames, constraints, and rules. The algorithms for diagnosis are also based on standard techniques in AI, like theorem proving, heuristic search, constraint satisfaction, and qualitative simulation. In general, the model based methods deal only with models of correct behavior. There is no a priori specification of how components might fail, and a failure is taken to be any anomaly as compared to normal behaviour.

Automata based methods

In the context of DES several original theoretical approaches have been proposed using *automata*.

In [59] and [60] Lin *et al.* propose a state-based DES approach to failure diagnosis. The problems of off-line and on-line diagnosis are addressed separately and notions of diagnosability in both of these cases are presented. The authors give an algorithm for computing a diagnostic control, i.e., a sequence of test commands for diagnosing system failures. This algorithm is guaranteed to converge if the system satisfies the conditions for on-line diagnosability.

In [76] and [77] Sampath *et al.* propose an approach to failure diagnosis where the system is modeled as a DES in which the failures are treated as unobservable events; diagnosis is the process of detecting occurrences of these events from observed event sequences. The level of detail in a discrete event model appears to be quite adequate for a large class of systems and for a wide variety of failures to be diagnosed. The approach is applicable whenever failures cause a distinct change in the system status but do not necessarily bring the system to a halt. In [76] the authors provide a definition of diagnosability in the framework of formal languages and establish necessary and sufficient conditions for diagnosability of systems. Also presented in [76] is a systematic approach to solve the problem of diagnosis using diagnosers. A more detailed presentation of this approach is presented in Chapter 4.

In [75] Sampath *et al.* present an integrated approach to control and diagnosis. More specifically, authors present an approach for the design of diagnosable systems by appropriate design of the system controller and this approach is called active diagnosis. They formulate the active diagnosis problem as a supervisory control problem. The adopted procedure for solving the active diagnosis problem is the following: given the non-diagnosable language generated by the system of interest, they first select an “appropriate” sublanguage of this language as the legal language. Choice of the legal language is a design issue and typically depends on considerations such as acceptable system behavior (which ensures that the system behavior is not restricted more than necessary in order to eventually make it diagnosable) and detection delay for the failures. Once the appropriate legal language is chosen, they then design a controller (diagnostic controller), that achieves a closed-loop language that is within the legal language and is diagnosable. This controller is designed based on the

formal framework and the synthesis techniques that supervisory control theory provides, with the additional constraint of diagnosability.

In [35] Debouk *et al.* address the problem of failure diagnosis in DES with decentralized information. They propose a coordinated decentralized architecture consisting of two local sites communicating with a coordinator that is responsible for diagnosing the failures occurring in the system. They extend the notion of diagnosability, originally introduced in [76] for centralized systems, to the proposed coordinated decentralized architecture. In particular, they specify three protocols that realize the proposed architecture and analyze the diagnostic properties of these protocols.

In [12] Boel and van Schuppen address the problem of synthesizing communication protocols and failure diagnosis algorithms for decentralized failure diagnosis of DES with costly communication between diagnosers. The costs on the communication channels may be described in terms of bits and complexity. The costs of communication and computation force the trade-off between the control objective of failure diagnosis and that of minimization of the costs of communication and computation. The result of this paper is an algorithm for decentralized failure diagnosis of DES for the special case of only two diagnosers.

In [94] Zad *et al.* present a state-based approach for on-line passive fault diagnosis. In this framework, the system and the diagnoser (the fault detection system) do not have to be initialized at the same time. Furthermore, no information about the state or even the condition (failure status) of the system before the initiation of diagnosis is required. The design of the fault detection system, in the worst case, has exponential complexity. A model reduction scheme with polynomial time complexity is introduced to reduce the computational complexity of the design. Diagnosability of failures is studied, and necessary and sufficient conditions for failure diagnosability are derived.

In [51] Jiang and Kumar present a method for failure diagnosis of DES with linear-time temporal logic (LTL) specifications. The LTL formulas are used for specifying failures in the system. The LTL-based specifications make the specification specifying process easier and more user-friendly than the formal language/automata-based specifications. They can capture failures representing the violation of both liveness and safety properties, whereas the prior formal language/automaton-based specifications can capture the failures representing the violation of only the safety properties (such as the occurrence of a faulty event or the arrival at a failed state). Prediagnosability and diagnosability of DES in the temporal logic setting are defined. The problem of testing prediagnosability and diagnosability is reduced to the problem of model checking. An algorithm for the test of prediagnosability and diagnosability, and the synthesis of a diagnoser is obtained. The complexity of the algorithm is exponential in the length of each specification LTL formula, and polynomial in the number of system states and the number of specifications.

In [65] Lunze *et al.* describe a method for detecting and identifying faults that occur in the sensors or in the actuators of dynamical systems with discrete-valued inputs and outputs. The model used in the diagnosis is a stochastic automaton. The generalized observer scheme (GOS), which has been proposed for systems with continuous-variable inputs and outputs, are developed for discrete systems. This scheme solves the diagnostic problem as an observation problem. As the system under consideration is described by a stochastic automaton rather than a differential equation, the mathematical background and the diagnos-

tic algorithms obtained are completely different from the well-known observers developed for continuous-variable systems. The GOS is extended here by a fault detection module to cope with plant faults that are different from actuator or sensor faults. The diagnostic algorithm consists of two steps, the first detecting the existence of a fault and the second isolating possible sensor or actuator faults or identifying plant faults.

Petri nets based methods

Although automata models are suitable for describing DES, the use of *Petri nets* (PNs) offers significant advantages because of their twofold representation: graphical and mathematical. Moreover, the intrinsically distributed nature of PNs where the notion of state (i.e., marking) and action (i.e., transition) is local reduces the computational complexity involved in solving a diagnosis problem.

Among the first pioneer works dealing with PNs, we recall the work of Prock in [71] who proposes an on-line technique for fault detection based on monitoring the number of tokens residing into P-invariants: when the number of tokens inside P-invariants changes, then an error is detected.

In [81] Sreenivas and Jafari employ time PNs to model the DES controller and backfiring transitions to determine whether a given state is invalid. Later on, time PNs are employed by Ghazel *et al.* [46] to propose a monitoring approach for DES with unobservable events and to represent the "a priori" known behavior of the system, and track on-line its state to identify the events that occur.

Hadjicostis and Veghese in [48] use PN models to introduce redundancy into the system and additional P-invariants allow the detection and isolation of faulty markings.

Redundancy into a given PN is used by Wu and Hadjicostis [93] to enable fault detection and identification using algebraic decoding techniques. They consider two types of faults: place faults that corrupt the net marking, and transition faults that cause an incorrect update of the marking after event occurrence. Although this approach is general, the net marking has to be periodically observable even if unobservable events occur. Analogously, Lefebvre and Delherm [57] investigate the determination of the set of places that must be observed for the exact and immediate estimation of faults occurrence.

Ramirez-Treviño *et al.* [74] employ Interpreted PNs to model the system behavior that includes both events and states partially observable. Based on the Interpreted PN model derived from an on-line methodology, a scheme utilizing a solution of a programming problem is proposed to solve the problem of diagnosis.

In [39] Dotoli *et al.* present a novel event-based approach for DES on-line monitoring, ensuring timely and accurate detection of system failures. The monitor model is based on first–order hybrid PNs, i.e., nets that make use of first order fluid approximation [5]. The proposed fault analysis technique relies on a modular framework, so that elementary monitors can be connected with other monitors to check more complex systems while avoiding the state space explosion problem. In addition, the presented monitor detects system faults as soon as possible, before the maximum execution time assigned to each task.

Note that, all papers in this topic assume that faults are modeled by unobservable transitions. However, while the above mentioned papers assume that the marking of certain places may be observed, a series of papers have been recently presented that are based on the assumption that no place is observable [6, 7, 9, 44].

In particular, Genc and Lafortune [44] propose a diagnoser on the basis of a modular approach that performs the diagnosis of faults in each module. Subsequently, the diagnosers recover the monolithic diagnosis information obtained when all the modules are combined into a single module that preserves the behavior of the underlying modular system. A communication system connects the different modules and updates the diagnosis information. Even if the approach does not avoid the state explosion problem, an improvement is obtained when the system can be modeled as a collection of PN modules coupled through common places.

The main advantage of the approaches of Genc and Lafortune [44] consists in the fact that, if the net is bounded, the diagnoser may be constructed off-line, thus moving off-line the most burdensome part of the procedure. Nevertheless, a characterization of the set of markings consistent with the actual observation is needed. Thus, large memory may be required.

An improvement in this respect has been given by Benveniste *et al.* [9], Basile *et al.* [6, 7] and Dotoli *et al.* [37].

In particular, Benveniste *et al.* [9] use a net unfolding approach for designing an on-line asynchronous diagnoser. The state explosion is avoided but the on-line computation can be high due to the on-line building of the PN structures by means of the unfolding.

In [6, 7] the diagnoser is built on-line by defining and solving Integer Linear Programming (ILP) problems. Assuming that the fault transitions are not observable, the net marking is computed by the state equation and, if the marking has negative components, an unobservable transition has occurred. The linear programming solution provides the sequence and detects the fault occurrences. Moreover, an off-line analysis of the PN structure reduces the computational complexity of the ILP problem.

Dotoli *et al.* in [37], in order to avoid the redesign and the redefinition of the diagnoser when the structure of the system changes, proposed a diagnoser that works on-line. In particular, it waits for an observable event and an algorithm decides whether the system behavior is normal or may exhibit some possible faults. To this aim, some ILP problems are defined and provide eventually the minimal sequences of unobservable transitions containing the faults that may have occurred. The proposed approach is a general technique since no assumption is imposed on the reachable state set that can be unlimited, and only few properties must be fulfilled by the structure of the PN modeling the system fault behavior.

Note that none of the above mentioned papers regarding PNs deal with *diagnosability*, namely none of them provide a procedure to determine a priori if a system is *diagnosable*, i.e., if it is possible to reconstruct the occurrence of fault events observing words of finite length.

In fact, whereas this problem has been extensively studied within the framework of automata as discussed above, in the PN framework very few results have been presented.

The first contribution on diagnosability of PNs was given by Ushio *et al.* [82]. They extend

a necessary and sufficient condition for diagnosability given by Sampath *et al.* [76, 77] to unbounded PN. They assume that the set of places is partitioned in observable and unobservable places, while all transitions are unobservable in the sense that their occurrences cannot be observed. Starting from the PN they build a diagnoser called *simple ω diagnoser* that gives them sufficient conditions for diagnosability of unbounded PNs.

Chung in [30], in contrast with Ushio's paper, assumes that part of the transitions of the PN modelling is observable and shows that the additional information from observed transitions in general adds diagnosability to the analysed system. Moreover starting from the diagnoser he proposes an automaton called *verifier* that allows a polynomial check mechanism on diagnosability but for finite state Petri net models.

In [90] Wen and Jeng proposed an approach to test diagnosability by checking the structure property of T-invariant of the nets. They use Ushio's diagnoser to prove that their method is correct, however they don't construct a diagnoser for the system to do diagnosis. In [91] Wen *et al.* present an algorithm, based on a linear programming problem, of polynomial complexity in the number of nodes for computing a sufficient condition of diagnosability of DES modeled by PN.

Future research in the fault diagnosis field could follow several directions. First, diagnosability conditions are not fully investigated when PNs are used. More precisely, it is necessary to establish if it is possible to detect with a finite delay occurrences of failures using a record of observed events. Second, the proposed procedures are not applicable to labeled PNs that exhibit some form of non-determinism, such as two or more transitions that share the same label. Third, the on-line approaches have to be improved in their computational complexity in order to apply the technique to large scale systems.

2.2 State of art for identification

Identification is a classical problem in system theory: given a pair of observed input-output signals identification consists in determining a system such that the input-output signals approximate the observed ones [86].

Input data are usually given in terms of behavioral descriptions (e.g., transition system, language), and the *identification* (or *synthesis*) problem aims to address two main issues. First, decide whether for the given behavioral specification there exists a PN (of a given class) whose behavior coincides with the specified behavior. Secondly, provide a constructive procedure to determine the PN that satisfies the given specifications.

The idea of learning the structure of an automaton from positive examples and from counterexamples has been explored since the early 80's in the formal language domain. As an example, we recall the early work of Gold [47] and Angluin [1].

Identification of free labeled Petri nets

One of the first original approaches to the identification of safe PNs was discussed by Hiraishi [49], who presented an algorithm for the construction of a *free labeled PN model*, i.e., PNs where transitions are not labeled, from the knowledge of a finite set of its firing sequences. In a first phase, a language is identified in the form of a finite state automaton from given firing sequences. In a second phase, the dependency relation is extracted from the language, and the structure of a PN is guessed. Provided that the language is generated by a special class of nets, the algorithm uniquely identifies the original net if a sufficiently large set of firing sequences is given.

In [67, 68] Meda and Mellado present an approach for the identification of free labeled Interpreted PNs. Their approach consists in observing the marking of a subset of places and, given some additional information on the dependency between transitions, allows one to reconstruct the part of the net structure related to unobservable places.

Bourdeaud'huy and Yim [13] propose an approach to reconstruct the incidence matrix and the initial marking of a free labeled net given some structural information on the net, such as the existence of P-invariants or T-invariants. This approach, that is based on logic constraints, can only deal with positive examples but not with counterexamples, namely it guarantees that the identified system generates all strings in the given language L, but it does not guarantee that all strings that are not in L are forbidden.

In [38] Dotoli *et al.* present a procedure to identify a labeled PN system by the *real time* observation of its dynamical evolution, assuming that the marking is partially known, the set of transitions is unknown, and only an upper bound on the number of places is given. The identification problem is solved using an algorithm that observes in real time the occurred events and the corresponding output vectors: an ILP problem is defined at each observation that allows to recursively identify a PN system. Necessary and sufficient conditions for the correct identification of the net are given.

Theory of regions

An alternative approach, that has inspired most of the contributions in identification topic since the early nineties, is based on the *theory of regions* [2, 4, 3, 10, 28, 32, 62, 63, 64]. Given a prefix-closed language $\mathscr{L}$ over some alphabet T, the language-based theory of regions tries to find a finite PN $N(\mathscr{L})$ in which the transitions correspond to the symbols in the alphabet of the language and of which all words in the language are firing sequences. The PN $N(\mathscr{L})$, whose set of places is empty and whose set of transitions contains all transitions in $\mathscr{L}$, is a finite PN in which all words are firing sequences. However, its behavior is not minimal. Therefore, the behavior of this PN needs to be reduced, such that the PN still reproduces all words in the language, but does not allow for more behavior. This is achieved by adding places to the PN. The theory of regions provides a method to calculate these places, using regions. Regions-based approaches mainly differ in the PN class and the model for the behavioral specifications considered.

The synthesis problem was first addressed by Ehrenfeucht and Rozenberg [41] introducing

regions to model the sets of states that characterize marked places. Interesting surveys on such PN synthesis approaches have been proposed by Badouel and Darondeau in [4], and very recently by Lorenz *et al.* in [64].

In [2, 4] the objective was that of deciding whether a given graph is isomorphic to the reachability graph of some free labeled net and then constructing it. In [2] Badoeul *et al.* give explicit algorithms for solving in polynomial time in the size of automata the synthesis problem for pure weighted PNs from a restricted class of regular languages or from finite automata. These methods have been implemented in the tool SYNET [26]. In [32] Cortadella *et al.* present a method which, given a finite state model, called transition system, synthesizes a safe, place-irredundant PN with a reachability graph that is bisimilar to the original transition system. In [3] Badouel and Darondeau provide an adaptation of the synthesis algorithm that works in polynomial time with respect to the number of states and to the cardinality of the alphabet for general PNs with the sequential firing rule and for PNs with step firing rule.

In [28] Carmona *et al.* provide an efficient synthesis approach for concurrent systems. An algorithm for bounded PNs synthesis based on the theory of general regions is presented. A bounded PN is always provided in case it exists. Otherwise, the events are split into several transitions to guarantee the synthesis of a PN with bisimilar behavior. Starting from the algorithms for synthesizing safe PNs in [32], the theory and algorithms are extended by generalizing the notion of excitation closure from sets of states to multisets of states. The extension covers the case of the k-bounded PNs with weighted arcs.

In [63] given a set of firing sequences, Lorenz and Juhás answer the question whether this set equals the set of all executions of a PN. They propose a definition of regions for a partial language. Moreover, given a partial language of firing sequences they prove a necessary and sufficient condition (based on regions) for the partial language to be the partial language of executions of a PN.

In [62] Lorenz *et al.* present an algorithm to synthesize a finite PN from a finite set of labeled partial orders (a finite partial language). This PN has minimal non-sequential behavior including the specified partial language. Consequently, either this net has exactly the non-sequential behavior specified by the partial language, or there is no such PN. They finally develop an algorithm to test whether the synthesized net has exactly the non-sequential behavior specified by the partial language.

In [10] Bergenthum *et al.* present an algorithm to synthesize a finite unlabeled PN from a possibly infinite partial language, which is given by a term over a finite set of labeled partial orders using operators for union, iteration, parallel composition and sequential composition. The synthesis algorithm is based on the theory of regions for partial languages presented in [63] and produces a PN having minimal net behavior including the given partial language. The algorithm uses linear programming techniques that were already successfully applied in [62].

In Chapter 5 will be presented in more detail the regions-based approach.

Process mining and workflow problems

The discovery of formal models from event logs in information systems is known as *process mining*. The term process mining refers to methods for distilling a structured process description from a set of real executions. The synthesis problem is related to process mining. However, process mining differs from synthesis in the knowledge assumption: while in synthesis one assumes a complete description of the system, only a partial description of the system is assumed in process mining. Therefore, bisimulation is no longer a goal to achieve in process mining. Instead, obtaining approximations that succinctly represent the log under consideration are more valuable. More in detail, the essential differences with classical approaches based on language learning are:

- there are no counterexamples;
- only a fraction of all possible behavior has been observed, i.e., processes allow for much more traces than the ones that were observed;
- processes are concurrent and not sequential.

A survey of approaches adopted in this field can be found in [83]. In [85] van der Werf *et al.* consider the problem of process discovery [84], i.e., they construct a process model describing the processes controlled by the information system by simply using the execution log, where an execution log is a finite set of traces. They restrict their attention to the control flow, i.e., they focus on the ordering of activities executed, rather than on the data recorded.

In [84] van der Aalst *et al.* address the workflow rediscovery problem. This problem is formulated as follows: "Find a mining algorithm able to rediscover a large class of sound workflow-nets on the basis of complete workflow logs." Under the assumption that the workflow log contains "sufficient" information, authors investigate whether it is possible to rediscover the workflow process, i.e., for which class of workflow models is it possible to accurately construct the model by merely looking at their logs. For this purpose, they use workflow nets that are a class of PNs specifically tailored toward workflow processes.

In [27] Carmona *et al.* present a new method for the synthesis of PNs from event logs in the area of process mining. This paper aims at constructing (mining) a PN that covers the behavior observed in the event log, i.e., traces in the event log will be feasible in the PN. Moreover, the PN may accept traces not observed in the log. Additionally, a minimality property is demonstrated on the mined PN: no other net exists that both covers the log and accepts less traces than the mined PN. The methods presented in the paper can mine a particular k-bounded PN, for a given bound k.

Identification of arbitrary Petri nets

Finally we present some approaches for *arbitrary PNs*, namely PNs where two or more transitions can share the same label and some transitions are labeled with the empty string.

In [80] Sreenivas studies the minimization of PN models. Given a measure of size of a labeled PN, he considers the existence of a procedure that takes as input a description of an arbitrary, labeled PN, and returns a description of a (possibly different) labeled PN with the smallest size that generates the same language as the input. This is a procedure of minimization, in fact, in [80] is investigated the existence of minimization procedures for a variety of measures and is shown that these procedures cannot exist for PN languages for a large class of measures. However, for families of PN languages where controllability is decidable, there can be minimization procedures for a restricted class of measures. After showing that minimization procedures for a family of measures are intractable for languages generated by bounded PNs, it is argued that a similar conclusion has to be reached for any family of PN languages that includes the family of regular languages for which there are minimization procedures.

Problems related to identification of labeled PNs have been recently studied by Ru and Hadjicostis in [73] and by Li *et al.* in [58].

In [73] Ru and Hadjicostis show that if the initial marking is known, but the transition may either be indistinguishable (they share the same label and can be simultaneously enabled) or silent (labeled with the empty string) the number of markings that are consistent with a given observation is at most polynomial in the length of the observation sequence (i.e., in the number of labels observed) even though the set of possible firing sequences can be exponential in the length of the observation sequence. The result applies to general PNs without any specific assumption on the structure of the PN or the nature of the labeling function.

Finally, Li *et al.* [58] assume that to each transition in the given net is associated a nonnegative cost which could represent its likelihood (e.g., in terms of the amount of workload or power required to execute the transition). A recursive algorithm is proposed that, given the structure of a labeled PN and the observation of a sequence of labels, finds the transition firing sequence(s) that has (have) the least total cost and is (are) consistent with both the observed label sequence and the PN. The complexity of the procedure is polynomial in the length of the observed label sequence and is thus amenable to on-line event estimation and monitoring.

Chapter 3

Background on Petri nets

Summary

In this chapter we recall the formalism of Petri nets used in the rest of the thesis.

3.1 Petri net model

Petri nets (PNs) are a discrete event system model developed in the early 1960s by C.A. Petri in his Ph.D. dissertation[70]. They represent one of the most efficient methods to analyze discrete event systems. PNs can be divided in discrete and timed. Moreover timed PNs are partitioned in deterministic and stochastic. In this thesis we deal with discrete PNs that is a logic model that allows us to represent the order of the occurrence events but not their temporization.

The reasons because PNs are so widely used are multiple.

- PNs are a mathematical and graphical formalism to model discrete event systems.
- PNs give a compact representation of the state space. In fact they do not require to explicitly represent all possible reachable states, but only the evolution rules.
- PNs can represent a discrete event system with an infinite number of states via a graph with a finite number of nodes.
- PNs can represent the concept of concurrency.
- PNs give a modular representation. In fact if a system is composed by more than one subsystems and these subsystems interact each other, then it is possible to represent each subsystem as a PN and then, using specific constructs, combine the single units to obtain the entire model.

3.2 Basic definitions

A *Place/Transition net* (P/T net) is a structure $N = (P, T, Pre, Post)$, where P is a set of m places; T is a set of n transitions; $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the pre– and $post$– incidence functions that specify the weight of the arcs directed from places to transitions and from transitions to places, respectively; $C = Post - Pre$ is the incidence matrix. The *preset* and *postset* of a node $X \in P \cup T$ are denoted ${}^\bullet X$ and $X^\bullet$ while ${}^\bullet X^\bullet = {}^\bullet X \cup X^\bullet$.

A *marking* is a vector $M : P \to \mathbb{N}$ that assigns to each place of a P/T net a nonnegative integer number of tokens, represented by black dots. We denote $M(p)$ the marking of place p. A P/T *system* or *net system* $\langle N, M_0 \rangle$ is a net N with an initial marking M_0.

A transition t is enabled at M iff $M \geq Pre(\cdot, t)$ and may fire yielding the marking $M' = M + C(\cdot, t) = M + C \cdot \vec{t}$, where $\vec{t} \in \mathbb{N}^n$ is a vector whose components are all equal to 0 except the component associated to transition t that is equal to 1. Given a sequence $\sigma \in T^*$, where T^* is the set of all finite sequences of transitions in T including the empty string ε, we write $M\,[\sigma\rangle$ to denote that the sequence of transitions σ is enabled at M, and we write $M\,[\sigma\rangle\,M'$ to denote that the firing of σ yields M'. Note that in this paper we always assume that two or more transitions cannot simultaneously fire (non-concurrency hypothesis).

Given a sequence $\sigma \in T^*$, we call $\pi : T^* \to \mathbb{N}^n$ the function that associates to σ a vector $y \in \mathbb{N}^n$, named the *firing vector* of σ. In particular, $y = \pi(\sigma)$ is such that $y(t) = k$ if the transition t is

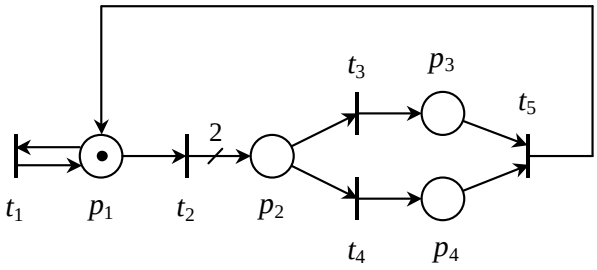


Figure 3.1: An example of Petri net

contained k times in σ. We denote $|\sigma|_t$ the number of occurrences of transition t in sequence σ.

Example 3.1. Let us consider the PN system in Figure 3.1. The set of places is $P = \{p_1, p_2, p_3, p_4\}$ and the set of transitions is $T = \{t_1, t_2, t_3, t_4, t_5\}$. The two matrices Pre and $Post$ are:

$$Pre = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad Post = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and the incidence matrix is

$$C = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 \\ 0 & 2 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}.$$

The initial marking is $M_0 = [1\ 0\ 0\ 0]^T$. Transitions t_1 and t_2 are enabled at M_0 and their firing yield to markings M_0 and $M_1 = [0\ 2\ 0\ 0]^T$, respectively. The preset and postset for transition t_2 and place p_2 are $\bullet t_2 = \{p_1\}$, $t_2^\bullet = \{p_2\}$, $\bullet p_2 = \{t_2\}$ and $p_2^\bullet = \{t_3, t_4\}$, respectively.

Let us consider the firing sequence $\sigma = t_1 t_1 t_2 t_3$ firable in the considered PN at M_0. The firing vector of σ is $\pi(\sigma) = [2\ 1\ 1\ 0\ 0]^T$, thus $|\sigma|_{t_1} = 2$, $|\sigma|_{t_2} = |\sigma|_{t_3} = 1$ and $|\sigma|_{t_4} = |\sigma|_{t_5} = 0$. ■

A marking M is *reachable* in $\langle N, M_0\rangle$ iff there exists a firing sequence σ such that $M_0\ [\sigma\rangle\ M$. In such a case the state equation $M = M_0 + C \cdot y$ holds, where $y = \pi(\sigma)$. The set of all markings reachable from M_0 defines the *reachability set* of $\langle N, M_0\rangle$ and is denoted $R(N, M_0)$. Finally, we denote $PR(N, M_0)$ the *potentially reachable set*, i.e., the set of all markings $M \in \mathbb{N}^m$ for which there exists a vector $y \in \mathbb{N}^n$ that satisfies the *state equation* $M = M_0 + C \cdot y$, i.e., $PR(N, M_0) = \{M \in \mathbb{N}^m \mid \exists\ y \in \mathbb{N}^n : M = M_0 + C \cdot y\}$. It holds that $R(N, M_0) \subseteq PR(N, M_0)$.

A PN having no directed circuit is called *acyclic*. For this subclass the following result holds.

Theorem 3.2. *[31] Let N be an acyclic PN.*

(i) If the vector $y \in \mathbb{N}^n$ satisfies the equation $M_0 + C \cdot y \geq \vec{0}$ there exists a firing sequence σ firable from M_0 whose firing vector is $\pi(\sigma) = y$.

(ii) A marking M is reachable from M_0 iff there exists a non negative integer solution y satisfying the state equation $M = M_0 + C \cdot y$, i.e., $R(N, M_0) = PR(N, M_0)$.

3.3 Net language

Given a PN system $\langle N, M_0 \rangle$ we define its *free-language*[1] as the set of its firing sequences

$$L(N, M_0) = \{\sigma \in T^* \mid M_0[\sigma\rangle\}.$$

We also define the set of firing sequences of length less than or equal to $k \in \mathbb{N}$ as:

$$L_k(N, M_0) = \{\sigma \in L(N, M_0) \mid |\sigma| \leq k\}.$$

Finally given a language $\mathscr{L} \subset T^*$ and a vector $y \in \mathbb{N}^n$ we denote

$$\mathscr{L}(y) = \{\sigma \in \mathscr{L} \mid \pi(\sigma) = y\}$$

the set of all sequences in $\mathscr{L}$ whose firing vector is y.

3.4 Structural properties

In this section we introduce some structural properties of PNs that will be used in Subsection 10.2.1.

Definition 3.3. *Let us consider a PN with m places, n transitions and incidence matrix C. A P-vector $\vec{x} : P \to \mathbb{N}$, with $\vec{x} \neq \vec{0}$, is called:*

- P-invariant*: if $\vec{x}^T \cdot C = \vec{0}^T$;*
- P-increasing*: if $\vec{x}^T \cdot C \gneq \vec{0}^T$;*
- P-decreasing*: if $\vec{x}^T \cdot C \lneq \vec{0}^T$.*

It can be shown that if $\vec{x}$ is a P-invariant (resp., P-increasing, P-decreasing) along any evolution the sum of the markings weighted with vector $\vec{x}$ remains constant (resp., does not decrease, does not increase).

A T-vector $\vec{y} : T \to \mathbb{N}$, with $\vec{y} \neq \vec{0}$, is called:

- T-invariant*: if $C \cdot \vec{y} = \vec{0}$;*
- T-increasing*: if $C \cdot \vec{y} \gneq \vec{0}$;*

[1] As it will appear in the following, *free* specifies that no labeling function is assigned to the considered PN system.

- T-decreasing*: if* $C \cdot \vec{y} \lneq \vec{0}$.

It can be shown that if $\vec{y}$ *is a T-invariant the firing of a sequence of transitions whose firing vector is* $\vec{y}$ *does not modify the number of tokens, i.e., it is a stationary sequence. If* $\vec{y}$ *is a T-increasing the firing of a sequence of transitions whose firing vector is* $\vec{y}$ *increases the number of tokens, i.e., it is a repetitive non stationary sequence. Finally if* $\vec{y}$ *is a T-decreasing the firing of a sequence of transitions whose firing vector is* $\vec{y}$ *decreases the number of tokens.* ■

A PN is said *ordinary* if $Pre, Post \in \{0,1\}^{m \times n}$, i.e., if each arc has weight equal to one.

A *marked graph* is an ordinary PN such that each place has exactly one input and one output transition.

A *state machine* is an ordinary PN where each transition has exactly one input and one output place.

A net system $\langle N, M_0 \rangle$ is *bounded* if there exists a positive constant k such that, for $M \in R(N, M_0)$, $M(p) \leq k$. A net is said *structurally bounded* if it is bounded for any initial marking.

Finally, let us consider the following definition.

Definition 3.4. *Given a net* $N = (P, T, Pre, Post)$*, and a subset* $T' \subseteq T$ *of its transitions, we define the* $T'-$induced subnet of N *as the new net* $N' = (P, T', Pre', Post')$ *where* $Pre', Post'$ *are the restriction of* $Pre, Post$ *to* T'*. The net* N' *can be thought as obtained from N removing all transitions in* $T \setminus T'$*. We also write* $N' \prec_{T'} N$*.* ■

3.5 Labeled Petri nets

When observing the evolution of a net, it is common to assume that each transition t is assigned a label $\varphi(t)$ and the occurrence of t generates an observable output $\varphi(t)$. If $\varphi(t) = \varepsilon$, i.e., if the transition is labeled with the empty string, its firing cannot be observed. This leads to the definition of labeled nets.

Definition 3.5. *Given a PN* N *with set of transitions* T*, a* labeling function $\varphi : T \to E \cup \{\varepsilon\}$ *assigns to each transition* $t \in T$ *a symbol, from a given alphabet* E*, or assigns to it the empty string* ε*.*

A labeled *PN system is a 3-tuple* $G = \langle N, M_0, \varphi \rangle$ *where* $N = (P, T, Pre, Post)$*,* M_0 *is the initial marking, and* $\varphi : T \to E \cup \{\varepsilon\}$ *is the labeling function.* ■

Four classes of labeling functions may be defined.

Definition 3.6. *The labeling function of a labeled PN system* $\langle N, M_0, \varphi \rangle$ *can be classified as follows.*

- Free*: if all transitions are labeled distinctly, namely a different label is associated to each transition, and no transition is labeled with the empty string.*

- Deterministic*: if no transition is labeled with the empty string, and the following condition*[2] *holds: for all* $t, t' \in T$*, with* $t \neq t'$*, and for all* $M \in R(N, M_0)$*:* $M[t\rangle \wedge M[t'\rangle \Rightarrow [\varphi(t) \neq \varphi(t')]$ *i.e., two transitions simultaneously enabled may not share the same label. This ensures that the knowledge of the firing label* $\varphi(t)$ *is sufficient to reconstruct the marking that the firing of* t *yields.*
- λ-free*: if no transition is labeled with the empty string*[3].
- Arbitrary*: if no restriction is posed on the labeling function* φ.

■

Each of these types of labeling is a generalization of the previous one. Furthermore all types of labeling only depend on the structure of the net, but for the deterministic labeling that depends both on the structure and on the behavior of the net.

In the particular case in which the labeling function is free, being an isomorphism between the alphabet E and the set of transitions T, it is usual to choose $E = T$, or equivalently to assume that the transitions are not labeled and their firing can be directly observed.

[2] A looser condition is sometimes given: for all $t, t' \in T$, with $t \neq t'$, and for all $M \in R(N, M_0)$: $M[t\rangle \wedge M[t'\rangle \Rightarrow [\varphi(t) \neq \varphi(t')] \vee [Post(\cdot, t) - Pre(\cdot, t) = Post(\cdot, t') - Pre(\cdot, t')]$. Thus two transitions with the same label may be simultaneously enabled at a marking M, if the two markings reached from M by firing t and t' are the same.

[3] In the PN literature the empty string is denoted λ, while in the formal language literature it is denoted ε. In this thesis we denote the empty string ε but, for consistency with the PN literature, we still use the term λ*-free* for a non erasing labeling function $\varphi : T \to E$.

Chapter 4

State of the art for diagnosis: Diagnoser Approach

Summary

This chapter presents the seminal approach to the problem of failure diagnosis of discrete event systems using automata ([76, 77]). Sampath *et al.* introduce a notion of diagnosability of DES in the framework of formal languages. Moreover, they present a systematic procedure for detection and isolation of failure events using diagnosers and provide necessary and sufficient conditions for a language to be diagnosable. The diagnoser performs diagnostics using on-line observations of the system behavior; it is also used to state and verify off-line necessary and sufficient conditions for diagnosability. These conditions are stated on the diagnoser.

In this chapter we present the seminal approach to the diagnosis of automata developed by M. Sampath, S.Lafortune, D. Teneketzis, R. Sengupta and K. Sinnamohideen in [76, 77]. Definitions, propositions and algorithms of this chapter are taken from [29] to which the reader is addressed for more details.

4.1 The system model

The system to be diagnosed is modeled as a *finite state automaton.*

Definition 4.1. *A* finite state automaton *is a quintuple denoted as* $G = (X, E, \delta, x_0, X_m)$*, where:*

- X *is a finite set of states,*
- E *is a finite set of events,*
- $\delta : X \times E \rightarrow X$ *is the partial transition function,*
- x_0 *is the initial state of the system,*
- $X_m \subseteq X$ *is the set of marked states.* ■

An automaton can be represented by a graph where each state is represented by a node. The initial state is denoted by an arrow and a final state by a double circle. If $\bar{x} = \delta(x, e)$ then there exists an arc directed from node x to node $\bar{x}$ labeled with symbol e. If we are not interested to take into account a set of marked states, the automaton is reduced to $G = (X, E, \delta, x_0)$.

Example 4.2. In Figure 4.1 is shown a finite state automaton G, where $X = \{x_0,\ x_1,\ x_2\}$, $E = \{a,\ b,\ c,\ d\}$, the initial state is equal to x_0 and the set of marked states is $X_m = \{x_0\}$. The partial transition function is given by the following table:

δ	a	b	c	d
x_0	x_1			
x_1		x_2		x_0
x_2			x_2	x_0

■

The behavior of the system is described by the prefix-closed language (see Appendix A) $\mathscr{L}(G)$ generated by G. Henceforth, we shall denote $\mathscr{L}(G)$ by $\mathscr{L}$. $\mathscr{L}$ is a subset of E^*, where E^* denotes the Kleene closure of the set E (see Appendix A).

The model G includes both the normal and the faulty behavior. The set of events E is partitioned as $E = E_o \cup E_u$ in two disjoint subsets, where E_o is the set of observable events and E_u is the set of unobservable events. Let $E_f \subseteq E$ denote the set of failure events which are to be diagnosed. Let us assume that $E_f \subseteq E_u$, since it is straightforward diagnose an observable failure event. The set of fault events is partitioned into m disjoint subsets that represent the set of fault classes:

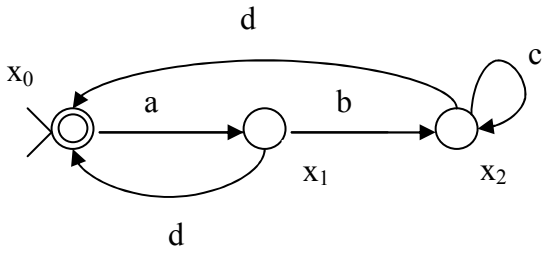


Figure 4.1: A finite state automaton.

$$E_f = E_{f1}, E_{f2}, \ldots, E_{fm}.$$

The aim is that of identifying the occurrence, if any, of failure events, given the set of generated words containing only observable events.

In the rest of the chapter the following assumptions hold.

(A1) The language L generated by G is live. This means that there not exists a state $x \in X$ from which no event is possible.

(A2) There does not exist in G any cycle of unobservable events.

Assumption (A1) is made for the sake of simplicity. On the contrary, assumption (A2) is necessary and ensures that the system G does not generate sequences of unobservable events whose length can be infinite.

Let us define the projection operator $P : E^* \rightarrow E_o^*$ as

$$\begin{cases} P(\varepsilon) = \varepsilon & \\ P(\sigma) = \sigma, & \text{if } \sigma \in E_o\,; \\ P(\sigma) = \varepsilon, & \text{if } \sigma \in E_u\,; \\ P(s\sigma) = P(s)P(\sigma), & s \in E^*, \sigma \in E\,. \end{cases}$$

Thus, P simply "erases" the unobservable events in a trace.

4.2 Observer automata

To solve the diagnosis problem the considered system is represented by an automaton whose set of events is partitioned into two disjoint subsets: the set of observable events and the set of unobservable events. Faults are modeled by unobservable events. Thus, the considered system is a nondeterministic automaton with unobservable transitions. A nondeterministic automaton G_{nd} can always be converted in a deterministic one, that has the same generated and marked language of G_{nd}. Here in the following we recall from [29] the algorithm to transform a non deterministic automaton in a deterministic one. This automaton is called *observer* corresponding to G_{nd} and is denoted as $Obs(G_{nd})$.

Algorithm 4.3. *([29] Section 2.5.2)**Construction of the Observer** $Obs(G_{nd})$ **of Nondeterministic Automaton** G_{nd}.*

Let $G_{nd} = (X, E, f_{nd}, x_0, X_m)$ be a nondeterministic automaton, where X is the set of states, E (con $E = E_o \cup E_u$), f_{nd} is the transition function, x_0 is the initial state and X_m is the set of marked states. Then, the observer is $Obs(G_{nd}) = (X_{obs}, E_o, f_{obs}, x_{0,obs}, X_{m,obs})$. Note that the set of events of the observer is restricted to the observable events.

Step 1: *Define $x_{0,obs} = \varepsilon R(x_0)$ [1]. Set $X_{obs} = \{x_{0,obs}\}$.*

Step 2: *For each $B \in X_{obs}$ and $e \in E$, define*

$$f_{obs}(B, e) = \varepsilon R(\{x \in X : (\exists x_e \in B)[x \in f_{nd}(x_e, e)]\})$$

whenever $f_{nd}(x_e, e)$ is defined for some $x_e \in B$. In this case, add the state $f_{obs}(B, e)$ to X_{obs}. If $f_{nd}(x_e, e)$ is not defined for any $x_e \in B$, then $f_{obs}(B, e)$ is not defined.

Step 3: *Repeat Step 2 until the entire part of $Obs(G_{nd})$ has been constructed.*

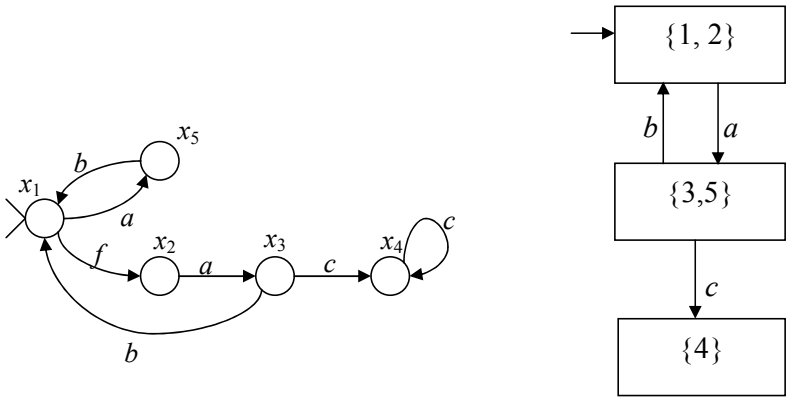
Step 4: $X_{m,obs} = \{B \in X_{obs} : B \cap X_m \neq \emptyset\}$. ■

The algorithm constructs the $Obs(G_{nd})$ starting from the initial node $x_{0,obs}$ to which corresponds the initial state of G_{nd} and all those states that are reached from x_0 firing one or more unobservable transitions (step 1). Now, we consider all the events that are enabled at $x_{0,obs}$ and we put them in a set called active set of $x_{0,obs}$. For each event $e \in E$ in the active set of $x_{0,obs}$ we consider all states $x \in X$ that can be reached from a state in $x_{0,obs}$. This set of states x'_{obs} is then extended to all those state that can be reached from one a state in x'_{obs} firing one or more unobservable transitions. The extended set x'_{obs} is equal to $f_{obs}(x_{0,obs}, e)$ of $Obs(G_{nd})$. The arc going from the initial node of the observer $x_{0,obs}$ to the new node $x'_{obs} = f_{obs}(x_{0,obs}, e)$ is labeled e and is added to the state transition diagram of $Obs(G_{nd})$. The procedure is iterated until the entire part of $Obs(G_{nd})$ has been constructed, i.e., until of the successors of $x_{0,obs}$ and then of the successors of the other states are explored (step 3). Finally, at step 4, are computed the marked states of the observer $X_{m,obs}$ as all nodes that contain at least one $x \in X_m$.

In simple words, the observer $Obs(G_{nd})$ gives us an estimation on the possible states of the system after an observed word w. In fact, $Obs(G_{nd})$ is a deterministic automaton that has the same generated and accepted language of G_{nd}.

Example 4.4. Let us consider the automaton G in Figure 4.2.(a). The set of observable events is $E_o = \{a, b, c\}$ and the set of unobservable events, that is equal to the set of fault events, is $E_u = E_f = \{f\}$. The observer $Obs(G)$ is shown in Figure 4.2.(b). The initial state of the observer is $x_{0,obs} = \{1, 2\}$. It means that if no event is observed the system G can be either in state 1 or in state 2. Analogously, if word $w = a$ is observed the system G can be either in state 3 or in state 5. Finally, if $w = ac$ is observed, we are sure that the system is in state 4. ■

[1] $\varepsilon R(x) = f_{nd}(x, \varepsilon)$, i.e., is the set of all states that are reached starting from x firing ε-transitions.



(a) (b)

Figure 4.2: (a) automaton G of Example 4.4; (b) its observer

4.3 Diagnosis

Diagnosis problem is the problem of associate to each observed string of events a diagnosis state, such as "normal" or "faulty" or "uncertain". The uncertainty can be reduced continuing to make observations. In Example 4.4 for instance (G is shown in Figure 4.2.(a) and $E_u = \{f\}$), after observing string $t = a$, we do not know if the system has executed the fault event f or not. However, after observing $w = ac$, we know with certainty that event f must have occurred. Thus we have diagnosed the occurrence of unobservable event f after observing w. We can automate this kind of inferencing about the past constructing an automaton that is similar to the observer, but that contains as additional information regarding the occurrence of fault transitions. We call this modified observer a *diagnoser automaton.*

Let us denote the diagnoser built from G by $Diag(G)$. In the following, for the sake of simplicity, we consider that we are interested to diagnose a single fault event f. Diagnoser $Diag(G)$ is similar to the observer $Obs(G)$ with the difference that in each node is contained not only a set of states where the system can be, but to each of these states is associated either a label N or a label Y. Label N means that "f has not occurred yet" while Y means that "yes, f has occurred". If a label is attached to a state $x \in X$ we write xN or xY as abbreviated notation for (x, N) or (x, Y), respectively.

From Section 2.5.3 in [29] we recall the key modifications to the construction of $Obs(G)$ (see Algorithm 4.3) for the purpose of building $Diag(G)$:

M1. When building the unobservable reach of the initial state x_0 of G:

(a) Attach the label N to states that can be reached from x_0 by unobservable strings in $[E_u \setminus f]^*$;

(b) Attach the label Y to states that can be reached from x_0 by unobservable strings that contain at least one occurrence of f;

(c) If state z can be reached both with and without executing f, then create two entries in the initial state set of $Diag(G)$: zN and zY.

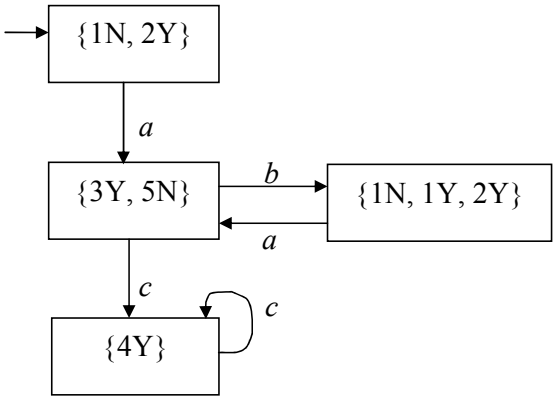


Figure 4.3: Diagnoser $Diag(G)$ of automaton G in Figure 4.2.(a).

M2. When building subsequent reachable states of $Diag(G)$:

(a) Follow the rules for the transition function of $Obs(G)$, but with the above modified way to build unobservable reaches with state labels;

(b) Propagate the label Y. Namely, any state reachable from state zY should get the label Y to indicate that f has occurred in the process of reaching z and thus in the process of reaching the new state.

M3. No set of marked states is defined for $Diag(G)$.

Diagnoser $Diag(G)$ is a deterministic automaton, whose set of events is $E = E_o$ and that generates a language $\mathscr{L}(Diag(G)) = P[L(G)]$. Each state of $Diag(G)$ is a subset of $X \times \{N, Y\}$. Modification $M1(c)$ implies that if a state can be reached by two paths having the same observable projection and such that one path contains the fault f and the other one does not in a node of $Diag(G)$ will exist two pairs xN and xF. It means that the cardinality of $Diag(G)$ is always greater than or equal to the cardinality of $Obs(G)$.

Example 4.5. Figure 4.3 shows the diagnoser $Diag(G)$ of automaton G in Figure 4.2.(a), where f is the event to be diagnosed. The cardinality of diagnoser is greater than the cardinality of observer, since node 1 of the system G appears once with label N and once with both labels N and F. By $Diag(G)$ is evident that after the occurrence of event c we are sure that the fault has occurred. On the other hand if we observe word $w = ab$ we have an uncertainty situation, due to the fact that we may be either in state 1, before or after the occurrence of the fault, or in state 2 after the occurrence of the fault. ■

As shown by Example 4.5 we can perform diagnosis by examination of the diagnoser states. In $Diag(G)$ we can distinguish three different kind of states:

- *negative state:* if in the node of $Diag(G)$ for all pairs (x, l) $\quad l = N$. Thus reaching this node we are sure that fault f has not occurred yet;
- *positive state:* if in the node of $Diag(G)$ for all pairs (x, l) $\quad l = F$. Thus reaching this node we are sure that fault f has occurred;

- *uncertain state:* if in the node of $Diag(G)$ there exists at least one pair (x,l) such that $l = N$ and at least one pair (x,l), such that $l = F$. Thus, we cannot say nothing about the occurrence of fault f.

Note that, In many cases, one can modify the system so that the state explicitly tracks a fault that has occurred. For example, by adding states $2F, 3F, 4F$ in the model of Figure 4.2 (so that the number of states increases to 8), one simply needs to track the states of the system and not the events. This essentially reduces the diagnoser to an observer at the cost of more states in the system model.

4.4 Diagnosability

The problem of diagnosability is the problem to determine if the system is diagnosable, i.e., if once a fault has occurred the system can detect its occurrence in a finite number of steps. Let us introduce a definition of diagnosability for language that are live.

Definition 4.6. *([29] Section 2.5.3) Unobservable event f is* not diagnosable *in live language $\mathscr{L}(G)$ if there exist two strings sN and sY in $\mathscr{L}(G)$ that satisfy the following conditions:*

(i) *sY contains f and sN does not;*

(ii) *sY is of arbitrarily long length after f;*

(iii) $P(sN) = P(sY)$.

When no such pair of strings exists, f is said to be diagnosable in $\mathscr{L}(G)$. ■

In other words, diagnosability requires that each fault event leads to distinct observations, sufficient to allow the identification of the fault with a finite delay. Assumption (A1) allows us to state that when the property of diagnosability is satisfied, we are sure that if f occurs, then $Diag(G)$ will enter a positive state in a bounded number of events after the occurrence of f.

Let us now introduce the definition of indeterminate cycle that is fundamental to test the property of diagnosability in the diagnoser.

Definition 4.7. *([76]) Let us consider a system G and its diagnoser $Diag(G)$. We say that a cycle in $Diag(G)$ is an* indeterminate cycle *if it is composed exclusively of uncertain states for which there exist:*

- *a corresponding cycle (of observable events) in G involving only states that carry Y in their labels in the cycle in $Diag(G)$ and*
- *a corresponding cycle (of observable events) in G involving only states that carry N in their labels in the cycle in $Diag(G)$.* ■

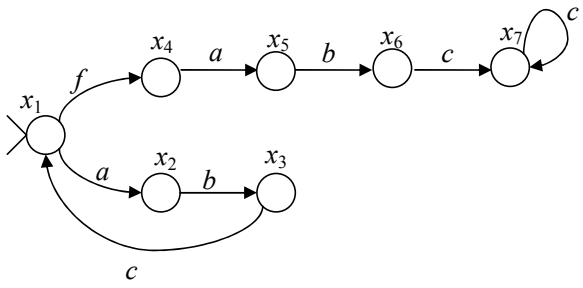


Figure 4.4: Automaton G of Example 4.10.

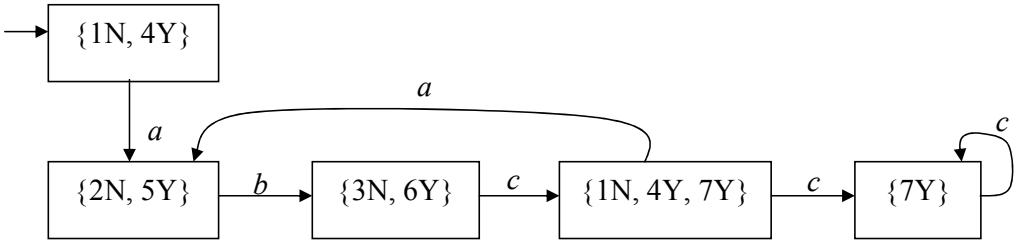


Figure 4.5: Diagnoser $Diag(G)$ of automaton G in Figure 4.4.

The notion of indeterminate cycles is very important because their analysis gives us necessary and sufficient conditions for diagnosability and gives a method to testify the property of diagnosability of the system.

Proposition 4.8. *[76] A language $\mathcal{L}$ without multiple failures of the same type is* diagnosable *if and only if its diagnoser $Diag(G)$ has no indeterminate cycles for all failure types E_{fi}.*

Let us consider the following example.

Example 4.9. Consider the system G and its diagnoser $Diag(G)$ shown respectively in Figures 4.2.(a), 4.3 and previously introduced in Examples 4.4, 4.5. The diagnoser has a potential indeterminate cycle. In fact, it has a cycle of uncertain cycles. Let us verify if conditions of Definition 4.7 are satisfied. In G there exists the cycle " $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$" involving only states that carry Y in their labels and there exists another cycle " $1 \rightarrow 5 \rightarrow 1$" involving only states that carry N in their labels. Thus, this cycle is indeterminate thus the fault is not diagnosable.

■

It is important to emphasize that the presence of a cycle of uncertain states in a diagnoser does not necessarily imply inability to diagnose with certainty an occurrence of event f. Consider the following example.

Example 4.10. Consider the system G and its diagnoser $Diag(G)$ shown respectively in Figures 4.4, 4.5, where the unique unobservable and fault event is f. The diagnoser has a potential indeterminate cycle. In fact, it has a cycle of uncertain states. However the only cycle that can cause the diagnoser to remain in its cycle of uncertain states is the cycle " $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$", and these states all have the N label in the corresponding diagnoser states. The cycle of uncertain states in the diagnoser is therefore not indeterminate. ■

For the case of multiple faults, namely events belonging to the same fault class, we refer to [29].

Chapter 5

State of the art for identification: Theory of regions

Summary

In this chapter we present in detail the theory of regions. In particular, we propose two methods taken from the literature for synthesize a Petri net starting from a labeled graph ([45]) and starting from a given language ([64]).

One of the most popular identification approach for Petri nets(PNs) is based on the *theory of regions*. In this framework two main problems have been solved:

- *synthesis problem for labeled graphs:* given a labeled graph decide if there exists a PN such that its reachability graph is isomorphic to the input graph, and if such PN exists construct it;
- *synthesis problem for languages* given a language $\mathscr{L}$ synthesize a PN such that $L(N, M_0) = \mathscr{L}$, namely the language of the PN is equal to the input language.

The two synthesis problems are very similar, although the first one requires stronger assumptions.

5.1 Synthesis problem for labeled graphs

One of the major contributions on the synthesis problem for labeled graph is given by [2, 4]. These works were done from a pure theoretical computer scientist's point of view and terminologies used are unfamiliar to the control community. A new and very useful interpretation of the theory of regions using the basic notions of PNs was presented by Ghaffari *et al.* in [45]. Although these authors use this new interpretation to design a control policy, we report here their interpretation because it explains in a very easy way how the theory of regions synthesize a PN starting from a labeled graph.

Let T be a set of transitions and G a finite oriented graph where arcs are labeled by transitions in T. Assume that there exists a node S_0 in G such that there exists a path from it to any node. The objective of the theory of regions is to find a pure PN (N, M_0), having T as its set of transitions and characterized by its incidence matrix C and its initial marking M_0, such that its reachability graph is isomorphic to G and the marking of the node S_0 is M_0. Let M to denote both a reachable marking and its corresponding node in G.

Let us consider any place p of the net (N, M_0) we are synthesizing. Since (N, M_0) is pure, i.e., the net is loop free, p can be fully characterized by its corresponding incidence vector $C(p, \cdot)$.

For any transition t that is firable at any marking M, i.e., t is the label of an outgoing arc of the node M in G, the following equation holds:

$$M'(p) = M(p) + C(p,t), \quad \forall (M, M') \in G \text{ and } M[t\rangle M' \tag{5.1}$$

where M' is the new marking, or equivalently, the destination node of arc t.

Consider now any nonoriented cycle γ of the reachability graph. Applying the state equation to nodes in γ and summing them up gives the following *cycle equation*:

$$\sum_{t \in T} C(p, y) \cdot \vec{\gamma}[t] = 0, \quad \forall \gamma \in S \tag{5.2}$$

where $\vec{\gamma}[t]$ denotes the algebraic sum of all occurrences of t in γ and S is the set of nonoriented cycles of the graph. $\vec{\gamma}$ denotes the firing vector of γ.

Consider now each node M of the reachability graph G. According to the definition of G, there exists a nonoriented path Γ_M from the initial state M_0 to M. Applying (5.1) along the path leads to $M(p) = M_0(p) + C(p,\cdot)\vec{\Gamma}_M$, where $\vec{\Gamma}_M$ is the firing vector of the path Γ_M. There may exist several paths from M_0 to M. Under the cycle equations, the product $C(p,\cdot)\vec{\Gamma}_M$ is the same for all these paths. As a result, the path Γ_M can be arbitrarily chosen.

The reachability of any marking M in G implies that

$$M_0(p) + C(p,\cdot)\vec{\Gamma}_M \geq 0, \quad \forall M \in G \tag{5.3}$$

which will be called the *reachability condition*.

It is now clear that the cycle equations and the reachability conditions hold for any place p of the net (N, M_0). Unfortunately, these equations are not sufficient to obtain the reachability graph G. In order to obtain exactly the reachability graph G, for each pair (M, t) such that M is a reachable marking of G and t is a transition not firable at M, t should be prevented from happening by some place p. Since the net is pure, t is prevented from happening at M by a place p iff

$$M_0(p) + C(p,\cdot)\vec{\Gamma}_M + C(p,t) \leq -1. \tag{5.4}$$

Relation (5.4) is called the *event separation condition of* (M, t).

The set of all possible pairs (M, t) where M is a reachable marking and t is not firable at M are called the set of event separation instances.

One last condition for having the same reachability graph is to ensure that the markings are different one from another. This is the so-called *state separation condition* and is as follows:

$$\forall M, M' \in G, \exists p \text{ such that } M(p) \neq M'(p). \tag{5.5}$$

The following theorem gives necessary and sufficient conditions to find a PN whose reachability graph is isomorphic to the one that is given.

Theorem 5.1. *[45] There exists a PN* (N, M_0) *with G as its reachability graph iff there exists a set P of places* $(M_0(p), C(p,\cdot))$ *such that*

1. *each place p satisfies cycle equation* (5.2) *and reachability conditions* (5.3)*,*
2. *the set of places P satisfies the state separation conditions* (5.5)*, and*
3. *for each transition t not firable at a reachable marking M, there exists a place p that satisfies the event separation condition* (5.4) *of* (M, t)*.*

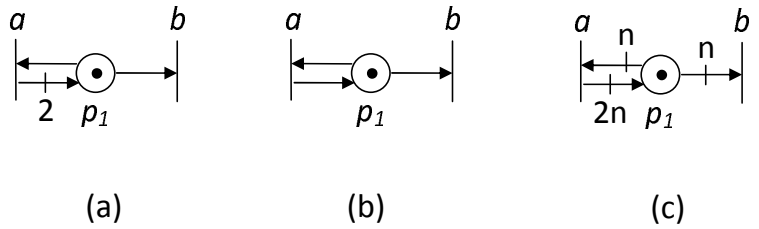


Figure 5.1: (a) A feasible place and (b) a non-feasible place w.r.t. the language $\mathscr{L} = \{a, b, ab, aab, aba\}$.

5.2 Synthesis problem for languages

If the behavior of the PN system to be identified is expressed in terms of the language $\mathscr{L}$ it can generate, the basic idea of region-based approaches can be summarized as follows [64].

The set of transitions is initially assumed equal to the alphabet E over which $\mathscr{L}$ is defined, namely $T = E$, while the set of places is assumed empty. In such a case, the resulting net generates each execution in $\mathscr{L}$. To find an exact solution to the identification problem, this language is then restricted adding *feasible* places, namely places guaranteeing that $\mathscr{L} \subseteq L(N, M_0)$, where $\langle N, M_0 \rangle$ is the resulting system.

In particular, the basic idea is to add *all* places that are feasible so as to *minimize* the language $L(N, M_0)$, while guaranteing that $\mathscr{L} \subseteq L(N, M_0)$. However, since the set of feasible places is in general *infinite* [64], a finite subset should be appropriately selected. Feasible places are defined through a so called *region* of the given language $\mathscr{L}$, that identifies a dependency between two sets of transitions.

Let us consider an example to better clarify the concept.

Example 5.2. Let us consider the language $\mathscr{L} = \{a, b, ab, aab, aba\}$ over the finite alphabet $E = \{a, b\}$. We want to synthesize a PN such that $L(N, M_0) = \mathscr{L}$. If we consider the PN having two transitions $t_1 = a$ and $t_2 = b$ and no places then for sure $\mathscr{L} \subset L(N, M_0)$. In Figure 5.1.(a) is shown an example of feasible place and in Figure 5.1.(b) is shown an example of non-feasible place w.r.t. the language $\mathscr{L}$. In fact, while in the PN shown in Figure 5.1.(a) $\mathscr{L} \subset L(N, M_0)$, in the PN in Figure 5.1.(b) $\mathscr{L} \not\subset L(N, M_0)$, in fact the word aba is not an execution of the PN. It is easy to see, looking at Figure 5.1.(c), that the set of feasible places is in general infinite. In fact for each $n \in \mathbb{N}$ the place is feasible. ■

In [64] the synthesis problem is presented for several class of PNs (Petri nets, elementary nets, PNs with inhibitor arcs and elementary nets with inhibitor arcs) and for different language type (finite or regular classical language, trace languages and partial languages). Moreover two different types of regions (transition-region and token flow-region) are used for the definition of feasible places. They show that for both types the set of all regions of a given language $\mathscr{L}$ equals the set of non-negative integral solutions of a possibly infinite linear system of the form $A_L \cdot x \leq b_L$. For finite or regular languages, this representation of regions is used to define computable finite representations of the infinite set of all regions.

In this chapter we present only one of these cases that is the closest to our approach: synthesis problem where the input language is a classical language, the net to identify is a PN and the region type used is the transition-region. Let us show how a region for the definition of a feasible place can be defined for this case.

A *transition-region* r directly defines a PN place p_r by determining the numbers $m_0(p_r)$ and $Pre(p_r,t)$, $Post(p_r,t)$ for $t \in T$. If $T = t_1,...,t_n$, then r is given as a $(2n+1)$-tuple $r = (r_0,...,r_{2n})$ of non-negative integers. Its components define these numbers via $m_0(p_r) = r_0$, $Pre(p_r,t_i) = r_i$ and $Post(p_r,t_i) = r_n + i$ for $i \in \{1,...,n\}$.

Example 5.3. Let us consider the PNs in Figure 5.1.(a),(b). If we consider $t_1 = a$ and $t_2 = b$ the place of the PN in Figure 5.1.(a) is defined by $r_{(a)} = (1,1,1,2,0)$ and the place of the PN in Figure 5.1.(b) by $r_{(b)} = (1,1,1,1,0)$. ■

Since a region r is intended to define a feasible place p_r, it is required to satisfy a property $(f)_L$ ensuring that p_r is feasible w.r.t. $\mathscr{L}$. Let us define a transition-region.

Definition 5.4. *A tuple r as above is called a* transition-region *if it satisfies $(f)_L$.* ■

The property $(f)_L$ depends on the considered net class and on the type of $\mathscr{L}$.

Theorem 5.5. *A tuple r satisfies $(f)_L$ if and only if p_r is feasible w.r.t. $\mathscr{L}$.*

That means the regions of $\mathscr{L}$ exactly define feasible places.

Remember that p_r is feasible w.r.t. $\mathscr{L}$ if the net resulting from adding p_r still generates $\mathscr{L}$. The property $(f)_L$ given a classical language as input and a PN as net to synthesize, is defined as follows.

For each $w = w'a_m \in \mathscr{L}$ it holds

$$r_0 - \sum_{i=1}^{n} |w|_{t_i}{}^1 r_i + \sum_{i=1}^{n} |w'|_{t_i} r_{n+i} \geq 0.$$

This is the case if and only if $a_w \cdot r \leq 0$ for $a_w = (a_{w,0},\ldots,a_{w,2n})$ defined by

$$a_{w,j} = \begin{cases} -1 & \text{if } j = 0, \\ |w|_{t_j} & \text{if } j \in \{1,\ldots,n\}, \\ -|w'|_{t_{j-n}} & \text{if } j \in \{n+1,\ldots,2n\}. \end{cases}$$

Example 5.6. Let us denote $t_1 = a$ and $t_2 = b$. For the the place of the PN in Figure 5.1.(b) defined by $r_{(b)} = (1,1,1,1,0)$ there holds for $w = aba$: $a_w = (-1,|w|_{t_1},|w|_{t_2},-|ab|_{t_1},-|ab|_{t_2}) = (-1,2,1,-1,-1)$ and $a_w \cdot r_{(b)} = 1 > 0$. Thus $r_{(b)}$ is not a region of $\mathscr{L} = \{aba\}$ and this place is non-feasible w.r.t. $\mathscr{L}$. On the other side, for the place of the PN in Figure 5.1.(a), defined by $r_{(a)} = (1,1,1,2,0)$, there holds for $w = aba$: $a_w \cdot r_{(a)} = 0$. Thus $r_{(a)}$ is a region of $\mathscr{L}$ and this place is feasible w.r.t. $\mathscr{L}$. ■

[1] For a classicla language $\mathscr{L}$ and $w \in \mathscr{L}$, $|w|_a$ denotes the number of a's occurring in w. For example $|aba|_a = 2$.

5.2.1 Computing finite representations

As shown in the last paragraph, property $(f)_L$ can be encoded by a system of linear equations or inequations of the form $A_L \cdot r = 0$ for a matrix A_L. Each row of A_L reflects one condition of $(f)_L$, that means A_L has in general infinite many rows. But there are several cases, when A_L can be chosen with finite many rows. If $\mathscr{L}$ is finite then this is obviously the case. If $\mathscr{L}$ is infinite, then there are finite representations of $\mathscr{L}$ allowing to chose A_L to be finite.

In other words, r is a region if and only if it is possible to define a finite matrix A_L such that r is a non-negative integer solution of $A_L \cdot r \leq 0$. Then the set of regions can be computed as the set of nonnegative integer solutions of such a system. There are two possibilities to define a finite representation of such a set of solutions: the basis representation and the separating representation, that we briefly explain in the following two subsections.

5.2.2 Basis representation

For systems of the form $A_L \cdot r \leq 0$ there is a so called *basis representation* of the set of all nonnegative solutions. That means there are nonnegative basis-solutions $y_1, \ldots, y_n$ such that each solution x is a nonnegative linear combination of $y_1, \ldots, y_n$ of the form $x = \sum_{i=1}^{n} \lambda_i y_i$ for real numbers $\lambda_1, \ldots, \lambda_n \geq 0$.

In the case that all values in A_L are integral (this is the case here) also the values of $y_1, \ldots, y_n$ can be chosen integral. If p_i is the place defined by y_i and N_{fin} is the net containing exactly the places $p_1, \ldots, p_n$, then it is shown that $L(N_{fin}) = L(N_{sat})$, where $L(N_{sat})$ is the smallest net language satisfying $\mathscr{L} \subseteq L(N_{sat})$. That means N_{fin} is the best approximation to a solution of the synthesis problem generating $\mathscr{L}$. Since from the construction it is not clear whether N_{fin} is an exact solution of the synthesis problem, it is finally necessary to test whether $L(N_{fin}) = \mathscr{L}$ or not. N_{fin} is called basis representation of N_{sat}.

5.2.3 Separating representation

Another idea is to separate behavior specified in $\mathscr{L}$ from behavior not specified in $\mathscr{L}$ by a finite set of regions. For this, one defines a finite set $\mathscr{L}_c$ with $\mathscr{L} \cap \mathscr{L}_c = \emptyset$ satisfying that $L(N) \cap \mathscr{L}_c = \emptyset \Longrightarrow L(N) = \mathscr{L}$ for each net N. Then for each $w \in \mathscr{L}_c$ one tries to find a region r_w such that w is not an execution of the net having the place p_{r_w}, i.e., a region which separates $\mathscr{L}$ from w. If such a region exists, then the corresponding place is added to the net N_{fin} called *separating representation* of N_{sat}. There is an exact solution of the synthesis problem if and only if for each $w \in \mathscr{L}_c$ there is such a region r_w. In case $\mathscr{L}$ is a net language, it holds $L(N_{fin}) = L(N_{sat}) = \mathscr{L}$, that means N_{fin} is a possible solution. In case $\mathscr{L}$ is not a net language, it holds $L(N_{sat}) \subseteq L(N_{fin})$ but in general not $L(N_{sat}) = L(N_{fin})$. Thus, the separating representation in general is not the best approximation to a solution of the synthesis problem generating $\mathscr{L}$ (in opposite to the basis representation). On the other hand, it can be defined and computet not only for PNs and inhibitor nets, but also for sane and ordinary PNs, while basis representation cannot. It is easy to define and compute a separating representation for finite languages. The definition of the finite set $\mathscr{L}_c$ depends on the considered

language type. The elements of $\mathscr{L}_c$ are called wrong continuations. In each case, a wrong continuation is an execution in $\mathscr{L}$ extended by some subsequent event.

Example 5.7. Let us consider the classical language $\mathscr{L} = b, a, ab, aba, aab$. The finite set $\mathscr{L}_c = \{wt \mid w \in \mathscr{L}, t \in T, wt \notin \mathscr{L}\} = \{ba, bb, abb, abab, abaa, aabb, aaba\}$. ■

A region separating $\mathscr{L}$ from some $w \in \mathscr{L}_c$ is computed as an arbitrary solution of an adequate linear inequation system. Such a system consists of the equations/inequations given by A_L defining regions and an additional row d_w. The row is defined in such a way that $d_w \cdot r < 0$ if and only if w is not an execution of the net having the place p_r.

For the case that we treat above (transition-regions and PNs), the row d_w is defined in a similar way as the rows of A_L. For each $w \in \mathscr{L}$ there is a row a_w of A_L such that w is an execution of a net N if and only if all places p_r of N satisfy $a_w \cdot r \leq 0$. That means, w is not an execution of a net N if and only if there is a place p_r of N satisfying $a_w \cdot r > 0$. Thus d_w can be defined as $d_w = -a_w$ for $w \in \mathscr{L}_c$.

5.3 Computational complexity

Regarding the synthesis problem for labeled graph, in literature can be found that theory of regions approaches have a complexity that is polynomial in the number of states and events of the graph taken as input. However to verify if there is a set of regions admissible for the given graph and compute them, it is necessary to verify that for all states of the graph separation properties are verified (see [3]) and this can require an exponential time.

Analogously, for the problem of synthesizing a PN given a language as input the computational complexity is exponential. In fact, the separating representation can be computed in polynomial time, if the regular language is given by a regular expression in a special form. However computing this special form (from an arbitrary one) needs exponential size in the worst case.

Part II

Diagnosis

Chapter 6

Fault detection for discrete event systems using Petri nets with unobservable transitions

Summary

In this chapter we present a fault detection approach for discrete event systems using Petri nets. We assume that some of the transitions of the net are unobservable, including all those transitions that model faulty behaviors. Our diagnosis approach is based on the notion of basis marking and justification, that allow us to characterize the set of markings that are consistent with the actual observation, and the set of unobservable transitions whose firing enable it. This approach applies to all net systems whose unobservable subnet is acyclic. If the net system is also bounded the proposed approach may be significantly simplified moving the most burdensome part of the procedure off-line, thanks to the construction of a graph, called the *basis reachability graph*.

6.1 Consistent markings

In this chapter we assume that the set of transitions T is partitioned in two subsets T_o and T_u, i.e., $T = T_o \cup T_u$ and $T_o \cap T_u = \emptyset$. The set T_o includes all transitions that are *observable*, while T_u includes *unobservable* or *silent* transitions.

We denote as n_o (resp., n_u) the cardinality of set T_o (resp., T_u), and as C_o (resp., C_u) the restriction of the incidence matrix to T_o (T_u).

Definition 6.1. *Let $N = (P, T, Pre, Post)$ be a net with $T = T_o \cup T_u$. We define the following two operators.*

- *The* projection over T_o *is $P_o : T^* \to T_o^*$ defined as: (i) $P_o(\epsilon) = \epsilon$; (ii) for all $\sigma \in T^*$ and $t \in T$, $P_o(\sigma t) = P_o(\sigma)t$ if $t \in T_o$, and $P_o(\sigma t) = P_o(\sigma)$ otherwise.*
- *The* projection over T_u *is $P_u : T^* \to T_u^*$ defined as: (i) $P_u(\epsilon) = \epsilon$; (ii) for all $\sigma \in T^*$ and $t \in T$, $P_u(\sigma t) = P_u(\sigma)t$ if $t \in T_u$, and $P_u(\sigma t) = P_u(\sigma)$ otherwise.* ■

Given a sequence $\sigma \in L(N, M_0)$, we denote $w = P_o(\sigma)$ the corresponding *observed word*.

Definition 6.2. *Let $\langle N, M_0 \rangle$ be a net system where $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$. Let $w \in T_o^*$ be an observed word. We define*

$$\mathscr{L}(w) = P_o^{-1}(w) \cap L(N, M_0) = \{\sigma \in L(N, M_0) \mid P_o(\sigma) = w\},$$

the set of firing sequences consistent *with $w \in T_o^*$.* ■

Definition 6.3. *Let $\langle N, M_0 \rangle$ be a net system where $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$. Let $w \in T_o^*$ be an observed word. We define*

$$\mathscr{C}(w) = \{M \in R(N, M_0) \mid \exists \sigma \in \mathscr{L}(w) : M_0[\sigma\rangle M\},$$

the set of markings consistent *with $w \in T_o^*$.* ■

In plain words, given an observation w, $\mathscr{L}(w)$ is the set of sequences that may have fired, while $\mathscr{C}(w)$ is the set of markings in which the system may actually be.

Example 6.4. Let us consider the net system in Fig. 6.1. It represents a production line processing damaged parts, namely metallic slabs where two plates instead of one, have been placed in a wrong decentralized position. When a damaged part is ready to be processed (tokens in p_1) slabs and plates are separated (transition t_1) and the two plates are sent in the upper line (modeled by places p_2, p_3, p_4, p_5, p_6), while the slab is sent in the lower line (modeled by places p_7, p_8, p_9, p_{10}, p_{11}). In the two lines parts are processed, namely smoothed, cleaned up, painted and polished (this corresponds to the firing of transitions ε_3 to ε_{10}). Finally one metallic plate is inserted in the slab in the correct position (transition t_2). The second plate is used again for other slabs, but this part of the process is not modeled here.

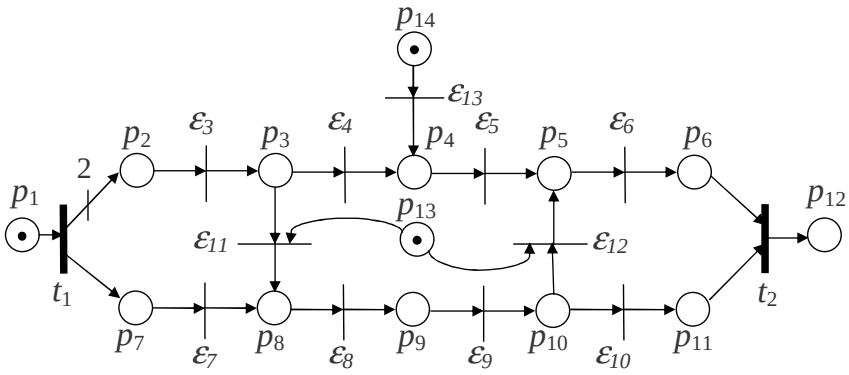


Figure 6.1: A Petri net modeling a part of a production line.

We assume that $T_o = \{t_1, t_2\}$ and $T_u = \{\varepsilon_3, \varepsilon_4, \ldots, \varepsilon_{13}\}$, where for a better understanding unobservable transitions have been denoted ε_i rather than t_i.

Assume no event is observed, namely $w = \varepsilon$. It is $\mathscr{L}(\varepsilon) = \{\varepsilon, \varepsilon_{13}, \varepsilon_{13}\varepsilon_5, \varepsilon_{13}\varepsilon_5\varepsilon_6\}$ and $\mathscr{C}(\varepsilon) = \{M_0, M_1, M_2, M_3\}$, where M_0 is the initial marking, $M_1 = [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^T$, $M_2 = [1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^T$ and $M_3 = [1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]^T$.

Now, assume t_1 is observed. Transition t_1 is enabled at the initial marking, thus the firing of no unobservable transition is necessary to enable it. After the firing of t_1 several sequences of unobservable transitions are enabled, and several markings are thus consistent with the actual behavior. In particular, all sequences $t_1\varepsilon_3$, $t_1\varepsilon_3\varepsilon_3$, $t_1\varepsilon_3\varepsilon_3\varepsilon_4$, $t_1\varepsilon_3\varepsilon_3\varepsilon_4\varepsilon_4\varepsilon_{13}$, …, $t_1\varepsilon_7$, $t_1\varepsilon_7\varepsilon_8$, …, etc. may have fired given the actual observation, and $\mathscr{C}(w)$ includes all markings that are reached firing the above sequences.

Now, let us consider $w = t_2$. In such a case no sequence of unobservable transitions may enable it. Therefore, $\mathscr{C}(t_2) = \mathscr{L}(t_2) = \emptyset$.

Finally, let us consider $w = t_1 t_2$. In such a case we obtain

$$\begin{aligned}\mathscr{L}(t_1 t_2) \quad = \quad & \{t_1\varepsilon_3\varepsilon_4\varepsilon_5\varepsilon_6\varepsilon_7\varepsilon_8\varepsilon_9\varepsilon_{10}t_2,\ \varepsilon_{13}t_1\varepsilon_3\varepsilon_4\varepsilon_5\varepsilon_6\varepsilon_5\varepsilon_6\varepsilon_7\varepsilon_8\varepsilon_9\varepsilon_{10}t_2,\\ & t_1\varepsilon_3\varepsilon_{11}\varepsilon_8\varepsilon_9\varepsilon_{10}\varepsilon_{13}\varepsilon_5\varepsilon_6t_2,\ \varepsilon_{13}t_1\varepsilon_3\varepsilon_7\varepsilon_8\varepsilon_9\varepsilon_{10}\varepsilon_5\varepsilon_6, \quad \ldots\},\end{aligned}$$

$$\begin{aligned}\mathscr{C}(t_1 t_2) = \{&[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1]^T, [0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0]^T,\\ &[0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^T, [0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0]^T, \ldots\},\end{aligned}$$

where dots denote all other sequences that may have fired and all other markings consistent with $t_1 t_2$, respectively (that have not been reported here for the sake of conciseness). ■

6.2 Minimal explanations and minimal e-vectors

In this section we provide some basic definitions that will be useful in the following.

Definition 6.5. *Given a marking M and an observable transition $t \in T_o$, we define*

$$\Sigma(M,t) = \{\sigma \in T_u^* \mid M[\sigma\rangle M',\ M' \geq Pre(\cdot,t)\}$$

the set of explanations *of t at M, and we define*

$$Y(M,t) = \{e \in \mathbb{N}^{n_u} \mid \exists \sigma \in \Sigma(M,t) : \pi(\sigma) = e\}$$

the e-vectors *(or* explanation vectors*), i.e., firing vectors associated to the explanations.* ■

Thus $\Sigma(M,t)$ is the set of unobservable sequences whose firing at M enables t.

Among the above sequences we want to select those whose firing vector is minimal. The firing vector of these sequences are called *minimal e-vectors*.

Definition 6.6. *Given a marking M and a transition $t \in T_o$, we define*

$$\Sigma_{\min}(M,t) = \{\sigma \in \Sigma(M,t) \mid \nexists\, \sigma' \in \Sigma(M,t) : \pi(\sigma') \lneqq \pi(\sigma)\}$$

the set of minimal explanations *of t at M, and we define*

$$Y_{\min}(M,t) = \{e \in \mathbb{N}_u^n \mid \exists \sigma \in \Sigma_{\min}(M,t) : \pi(\sigma) = e\}$$

the corresponding set of minimal e-vectors. ■

Similar definitions have also been given in [11, 52].

Example 6.7. Let us consider the net system in Figure 6.1 already considered in Example 6.4. We compute the set of the minimal e-vectors for some markings. The e-vectors correspond to the unobservable transitions from ε_3 to ε_{13}.

It holds that $\Sigma(M_0,t_1) = \mathscr{J}(t_1) = \{\varepsilon\}$. Then $\Sigma(M_0,t_2) = \mathscr{J}(t_2) = \emptyset$. Finally, let $M_1 = M_0 + C(\cdot,t_1) = [0\,2\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1]^T$. Then $\Sigma(M_1,t_2) = \mathscr{J}(t_1t_2)$ (see Example 6.4) and

$$\begin{aligned} Y_{\min}(M_1,t_2) \quad = \{ \quad & [1\,1\,1\,1\,1\,1\,1\,1\,0\,0\,0]^T,\ [2\,1\,1\,1\,0\,1\,1\,1\,1\,0\,0]^T, \\ & [1\,0\,1\,1\,0\,1\,1\,1\,1\,0\,1]^T,\ [0\,0\,1\,1\,1\,1\,1\,1\,0\,0\,1]^T\}, \end{aligned}$$

i.e., the firing vectors relative to sequences in $\mathscr{J}(t_1t_2)$. ■

In [31] Corona *et al.* proved the following important result. We say that a P/T net is *backward conflict-free* if $\forall p \in P\ |{}^\bullet p| \leq 1$, i.e., if each place has at most one input transition.

Theorem 6.8. *[31] Let $N = (P,T,Pre,Post)$ be a Petri net with $T = T_o \cup T_u$. If the T_u-induced subnet is backward conflict-free, then $|Y_{\min}(M,t)| = 1$.*

Different approaches can be used to compute $Y_{\min}(M,t)$, e.g., see [11, 52].

In this thesis we suggest an approach that find all vectors in $Y_{\min}(M,t)$ if applied to nets whose T_u-induced subnet is acyclic (but not necessarily backward conflict-free). It simply requires algebraic manipulations, and is inspired by the procedure proposed by Martinez

and Silva [66] for the computation of minimal P-invariants. It can be briefly summarized by the following algorithm.

Note that the proposed approach can also be applied to T_u-induced subnets that are not acyclic. However, in this case the algorithm may enter a loop: to guarantee to terminate in a finite number of steps we need to add suitable termination criteria.

Algorithm 6.9. ***[Computation of*** $Y_{\min}(M,t)$***]***

Input*: a Petri net N whose unobservable subnet is acyclic,*
the set of unobservable transitions T_u
a marking M,
an observable transition t.
Output*: $Y_{\min}(M,t)$.*
1. *Let* $\Gamma := \left|\begin{array}{c|c} C_u^T & I_{n_u \times n_u} \\ \hline A & B \end{array}\right|$ *where* $A := (M - Pre(\cdot,t))^T$, $B := \vec{0}_{n_u}^T$.
2. *While* $A \geq 0^T$
2.1 *Choose an element* $A(i^*, j^*) < 0$.
2.2 *Let* $\mathscr{I}^+ = \{i \mid C_u^T(i, j^*) > 0\}$.
2.3 *For all* $i \in \mathscr{I}^+$
2.3.1 *add to* $[A \mid B]$ *a new row* $[A(i^*,\cdot) + C_u^T(i,\cdot) \mid B(i^*,\cdot) + e_i^T]$
where e_i is the i-th canonical basis vector.
2.4 *Remove the row* $[A(i^*,\cdot) \mid B(i^*,\cdot)]$ *from the table.*
End *while*
3. *Remove from B any row that covers other rows.*
4. *Each row of B is a vector in* $Y_{\min}(M,t)$.

■

The above algorithm can be explained as follows.

Given a marking M and a transition t, Algorithm 6.9 computes the *minimal e-vectors* , i.e., the firing vectors of unobservable sequences whose firing at M is necessary to enable t.

At step 1 a row vector is defined, $A = A(1,\cdot)$, that has a number of columns equal to the number of places of the net. This vector contains a negative element $A(1,j)$ if place p_j does not enable t at M, and the value $|A(i,j)|$ denotes the number of tokens missing from p_j to enable t. B initially is a null firing vector.

At step 2.1 we check if there exists an unobservable transition whose firing may increase the number of tokens in p_j, if so we consider all possible such firings (of one single transition) computing the markings reached by each of these firings, and vector B, in the right part of the table, denotes the corresponding firing vector. These new markings and the corresponding firing vectors will be the new rows of matrix A, while the previous row is removed.

The *while* loop is repeated until all markings represented by matrix A have nonnegative components.

Note that at step 2.3 it may be possible that the new row $[A(i^*,\cdot)+C_u^T(i,\cdot) \mid B(i^*,\cdot)+e_i^T]$ is identical to a row already in the table: if such is the case it is not necessary to add it.

A detailed example of application of Algorithm 6.9 is given in Example 6.10.

Example 6.10. Let us consider the net in Figure 6.1. Let $M = [0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1]^T$ and $t = t_2$. Being

$$C_u = \begin{array}{c} \begin{matrix} \varepsilon_3 & \varepsilon_4 & \varepsilon_5 & \varepsilon_6 & \varepsilon_7 & \varepsilon_8 & \varepsilon_9 & \varepsilon_{10} & \varepsilon_{11} & \varepsilon_{12} & \varepsilon_{13} \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \end{array}, \quad Pre(\cdot,t_2) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

we first assume

$\Gamma =$

$$\left[\begin{array}{cccccccccccccc|cccccccccc} 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

thus there are two elements of A, namely $A(1,6)$ and $A(1,11)$, that are negative. We consider $A(1,11) < 0$. In such a case $\mathscr{I}^+ = \{8\}$, thus by step 2.3.1 of Algorithm 6.9 we add the following row to Γ:

$$\left[\begin{array}{ccccccccccccc|cccccccccc} 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right]$$

obtained from the first row of A by adding the 8th row of Γ.

Now we remove the row $\Gamma(12,\cdot)$ from the table (step 2.4) and we restart the procedure because the matrix A still contains negative elements, namely $A(1,6)$ and $A(1,10)$. Let us consider $A(1,10)$. In such a case $\mathscr{I}^+ = \{7\}$, thus we add the following row to Γ:

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right|$$

obtained from the first row of A by adding the 7th row of Γ.

Now we remove $\Gamma(12,\cdot)$ from the table and we restart the procedure because there are still negative elements in A: $A(1,6)$ and $A(1,9)$. We focus on $A(1,6)$ thus $\mathscr{I}^+ = \{4\}$. By step 2.3.1 we add the following row to Γ:

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right|$$

and remove $\Gamma(12,\cdot)$ from the table. We have still two negative elements in A: $A(1,5)$ and $A(1,9)$. we select $A(1,9)$ thus $\mathscr{I}^+ = \{6\}$. The new row to add to Γ is:

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}\right|$$

obtained from the first row of A by adding the 6th row of Γ. Now we remove $\Gamma(12,\cdot)$ from the table and because there are still negative elements in A, namely $A(1,5)$ and $A(1,8)$, we restart the procedure. We consider $A(1,8)$ thus $\mathscr{I}^+ = \{5,9\}$ and we add two new rows to Γ:

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}\right|$$

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{array}\right|$$

obtained from the first row of A by adding the 5th and the 9th row of Γ, respectively. Now we remove $\Gamma(12,\cdot)$ from the table and we restart the procedure because A still contains negative elements. These are: $A(1,5), A(2,3), A(2,5), A(2,13)$. We choose $A(2,13)$ thus $\mathscr{I}^+ = \emptyset$. We remove the second line of A and $A(1,5)$ is the only negative element. In such a case $\mathscr{I}^+ = \{3,10\}$ thus we add the following two rows to Γ:

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}\right|$$

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{array}\right|$$

obtained from the first row of A by adding the 3rd and the 10th row of Γ, respectively. Now we remove $\Gamma(12,\cdot)$ from the table and we restart the procedure. Matrix A contains the following negative elements: $A(1,4), A(2,10), A(2,13)$. We choose $A(2,13)$ thus $\mathscr{I}^+ = \emptyset$, and we have to remove the second line of A. Now, in A there is only one negative element, i.e., $A(1,4)$, and $\mathscr{I}^+ = \{2,11\}$. By step 2.3.1 of Algorithm 6.9 we add the following two new rows to Γ:

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}\right|$$

$$\left|\begin{array}{cccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right| \left|\begin{array}{cccccccccccc} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right|$$

obtained from the first row of A by adding the 2nd and the 11th row of Γ, respectively. Now we remove $\Gamma(12,\cdot)$ from the table and we restart the procedure because there is a negative element in A, namely $A(1,3)$. Moreover, it holds $\mathscr{I}^+ = \{1\}$, and the new row to add to Γ is:

$$\left|\begin{array}{cccccccccccccc} 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right| \left|\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{array}\right|$$

obtained from the first row of A by adding the 1st row of Γ. We remove $\Gamma(12,\cdot)$ and observe that $A(2,2)$ is negative, and $\mathscr{I}^+$ is empty. Thus, we remove the second line of A. Now we stop because all elements of A are non negative.

Because we have only one line, obviously no line covers the others, and we conclude that the row of B, namely

$$| \ 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \ |$$

is the only element of $Y_{\min}(M,t)$. ■

6.3 Basis markings and j-vectors

In this section we introduce two of the most important concepts for our approach: *basis markings* and *j-vectors*. A basis marking M_b is a marking reached from the initial marking M_0 with the firing of the observed word w and of all unobservable transitions whose firing is necessary to enable w. A j-vector $y \in J_{\min}(M_0,w)$ is the firing vector of a sequence of unobservable transitions whose firing is necessary to reach M_b.

Definition 6.11. *Let $\langle N, M_0\rangle$ be a net system where $N = (P,T,Pre,Post)$ and $T = T_o \cup T_u$. Let $\sigma \in L(N,M_0)$ be a firable sequence and $w = P_o(\sigma)$ the corresponding observed word. We define the set of* justifications of w *as*

$$\mathscr{J}(w) = \left\{\sigma_u \in T_u^* \mid [\exists \sigma \in \mathscr{L}(w) \ : \sigma_u = P_u(\sigma)] \wedge \left[\nexists \sigma' \in \mathscr{L}(w) : \ \sigma'_u = P_u(\sigma') \wedge \pi(\sigma'_u) \lneq \pi(\sigma_u)\right]\right\}.$$

Moreover, we define

$$J_{\min}(M_0,w) = \{y \in \mathbb{N}_u^n \mid \exists \sigma_u \in \mathscr{J}(w) : \pi(\sigma_u) = y\}$$

the corresponding set of j-vectors. ■

In simple words, $\mathscr{J}(w)$ is the set of sequences of unobservable transitions interleaved with w whose firing enable w and whose firing vector is minimal. The firing vectors of these sequences are called j-vectors.

Definition 6.12. *Let $\langle N, M_0\rangle$ be a net system where $N = (P,T,Pre,Post)$ and $T = T_o \cup T_u$. Let w be a given observation and $\sigma_u \in \mathscr{J}(w)$ be one of its justifications. The marking*

$$M_b = M_0 + C_u \cdot y + C \cdot y', \quad \textit{where } \ y = \pi(\sigma_u), \ \ y' = \pi(w),$$

i.e., the marking reached firing w interleaved with the justification σ_u, is called basis marking *and y is called its* j-vector *(or* justification-vector*).* ■

Obviously, because in general more than one justification exists for a word w (the set $\mathscr{J}(w)$ is generally not a singleton), the basis marking may be not unique as well. Furthermore, two or more j-vectors may correspond to the same basis marking.

Note however that under appropriate assumptions on the T_u-induced subnet, the uniqueness of M_b may be ensured by the uniqueness of the j-vector. In particular, in [31] Corona *et al.* proved that this is true if the T_u-induced subnet is backward conflict-free, since in this case for each observation w there is only one justification.

Definition 6.13. *Let $\langle N, M_0\rangle$ be a net system where $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$. Let $w \in T_o^*$ be an observed word. We define*

$$\mathscr{M}(w) = \left\{(M, y) \mid \exists \sigma \in \mathscr{L}(w) : M_0[\sigma\rangle M \wedge M \in M_b \wedge y \in J_{\min}(M_0, w)\right\}$$

the set of couples (basis marking - relative j-vector) that are consistent *with $w \in T_o^*$.* ■

Note that the set $\mathscr{M}(w)$ does not keep into account the sequences of observable transitions that may have actually fired. It only keeps track of the basis markings that can be reached and of the firing vectors of the sequences of unobservable transitions that have fired to reach them. Indeed this is the information really significant when performing diagnosis. The notion of $\mathscr{M}(w)$ is fundamental to provide a recursive way to compute the set of j-vectors.

Proposition 6.14. *Given a net system $\langle N, M_0\rangle$ where $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$ and whose T_u-induced subnet is acyclic. Let $w = w't$ be a given observation. The set $J_{\min}(M_0, w)$ is defined as*

$$J_{\min}(M_0, w) \subseteq \{y \in \mathbb{N}^{n_u} \mid y = y' + e : y' \in J_{\min}(M_0, w'),\ e \in J_{\min}(M_b', t)\}$$

where $M_b' = M_0 + C_u \cdot y' + C_o \cdot \pi(w')$.

Proof: It follows from Definitions 6.6, 6.11 and 6.12. In particular, it follows from the fact that in Petri nets where the unobservable subnet is acyclic basis markings completely characterized the set of consistent markings, as it will be shown in Theorem 6.20. □

In simple words, the set of j-vectors $J_{\min}(M_0, w)$ is a subset of the set obtained by summing up the j-vectors in $J_{\min}(M_0, w')$ that lead to a basis marking M_b' that either enable t or enable a sequence of unobservable transitions enabling t, plus the vectors in $J_{\min}(M_b', t)$. It is a subset since a firing vector $y \in J_{\min}(M_0, w)$ obtained summing the two vectors $y' \in J_{\min}(M_0, w')$ and $e \in J_{\min}(M_b', t)$ could be not minimal as shown by the following example.

Example 6.15. Let us consider the Petri net in Figure 6.2. Let $M_0 = [0\ 0]^T$.

Let us consider the observation $w = t_1$. The set of justification is $\mathscr{J}(t_1) = \{\varepsilon_1, \varepsilon_2\}$.

Let us now consider $w = t_1 t_2$. The set of justification is $\mathscr{J}(t_1 t_2) = \{\varepsilon_2\}$. The firing sequence $\varepsilon_1\varepsilon_2$ is not a justification, because it is not minimal since it covers ε_2, i.e., $\pi(\varepsilon_2) < \pi(\varepsilon_1\varepsilon_2)$. ■

The following example will clarify the concepts previously introduced.

Example 6.16. Let us consider the net system in Figure 6.1. Let M_0 be the marking shown in the figure.

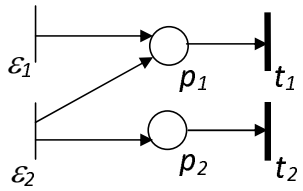


Figure 6.2: An example of Petri net.

Let us consider the observation $w = t_1$. It holds $\mathscr{J}(t_1) = \{\varepsilon\}$ and $J_{\min}(M_0, t_1) = \{\vec{0}\}$, thus the basis marking associated to $w = t_1$ is

$$M_b = M_0 + C(\cdot, t_1) = [0\ 2\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1]^T$$

and its j-vector is $\vec{0}$. Thus $\mathscr{M}(t_1) = \{(M_b, \vec{0})\}$.

Now, let us consider $w = t_1 t_2$. In such a case the set of justifications are

$$\begin{aligned}\mathscr{J}(t_1 t_2) \quad = \quad \{ & \varepsilon_3\varepsilon_4\varepsilon_5\varepsilon_6\varepsilon_7\varepsilon_8\varepsilon_9\varepsilon_{10},\ \varepsilon_3\varepsilon_4\varepsilon_5\varepsilon_6\varepsilon_3\varepsilon_{11}\varepsilon_8\varepsilon_9\varepsilon_{10},\\ & \varepsilon_3\varepsilon_{11}\varepsilon_8\varepsilon_9\varepsilon_{10}\varepsilon_{13}\varepsilon_5\varepsilon_6,\ \varepsilon_7\varepsilon_8\varepsilon_9\varepsilon_{10}\varepsilon_{13}\varepsilon_5\varepsilon_6,\quad \ldots\}\end{aligned}$$

where dots denote all other sequences (that have not been reported here for sake of conciseness) that are enabled at M_b and that have the same firing vector of the previous ones. The set of j-vectors is

$$\begin{aligned}J_{\min}(M_1, t_2) \quad = \{ \quad & [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0]^T,\ [2\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0]^T,\\ & [1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1]^T,\ [0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1]^T\}.\end{aligned}$$

Now, let

$$\begin{aligned}e_1 = [1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0]^T, \quad e_2 = [2\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 0]^T,\\ e_3 = [1\ 0\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1]^T, \quad e_4 = [0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 1]^T\},\end{aligned}$$

the basis markings reached after the firing of w are:

$$\begin{aligned}M_b^1 &= M_b + C_u \cdot e_1 + C(\cdot, t_2) = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1]^T,\\ M_b^2 &= M_b + C_u \cdot e_2 + C(\cdot, t_2) = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1]^T,\\ M_b^3 &= M_b + C_u \cdot e_3 + C(\cdot, t_2) = [0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0]^T,\\ M_b^4 &= M_b + C_u \cdot e_4 + C(\cdot, t_2) = [0\ 2\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0]^T,\end{aligned}$$

thus $\mathscr{M}(t_1 t_2) = \{(M_b^1, e_1), (M_b^2, e_2), (M_b^3, e_3), (M_b^4, e_4)\}$ and $J_{\min}(M_0, w)$ $= J_{\min}(M_1, t_2)$ being $J_{\min}(M_0, t_1) = \{\vec{0}\}$.

Now, let us consider $w = t_1 t_2 t_2$. It is easy to verify that

$$\begin{aligned}J_{\min}(M_b^1, t_2) = \{e_3\}, \quad J_{\min}(M_b^2, t_2) = \{e_4\},\\ J_{\min}(M_b^3, t_2) = \{e_1\}, \quad J_{\min}(M_b^4, t_2) = \{e_2\}.\end{aligned}$$

As a consequence at $w = t_1 t_2 t_2$ we only have one j-vector because

$$\begin{aligned}M_b^5 &= M_b^1 + C_u \cdot e_3 + C(\cdot, t_2)\\ &= M_b + C_u \cdot (e_1 + e_3) + 2C(\cdot, t_2),\\ M_b^6 &= M_b^2 + C_u \cdot e_4 + C(\cdot, t_2)\\ &= M_b + C_u \cdot (e_2 + e_4) + 2C(\cdot, t_2),\\ M_b^7 &= M_b^3 + C_u \cdot e_1 + C(\cdot, t_2)\\ &= M_b + C_u \cdot (e_3 + e_1) + 2C(\cdot, t_2),\\ M_b^8 &= M_b^4 + C_u \cdot e_2 + C(\cdot, t_2)\\ &= M_b + C_u \cdot (e_4 + e_2) + 2C(\cdot, t_2).\end{aligned}$$

In particular, it holds that $y_1 = e_1 + e_3 = e_2 + e_4$. Moreover,

$$M_b^5 = M_b^6 = M_b^7 = M_b^8 = [0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,2\,0\,0]^T,$$

thus there is only one basis marking at $w = t_1 t_2 t_2$.

Finally, $\mathscr{M}(t_1 t_2 t_2) = \{(M_b^5, y_1)\}$ and $J_{\min}(M_0, w) = \{y_1\}$. ■

The set $\mathscr{M}(w)$, that includes all couples (basis marking - relative j-vector) that are consistent with an observation w, can be easily constructed using the procedure informally presented in the previous example. The following algorithm formalizes this procedure. Note that, as for the Algorithm 6.9 the Petri net used as input must have the T_u-induced subnet acyclic.

Algorithm 6.17. ***[Computation of the basis markings and j-vectors]***

Input*: a Petri net $\langle N, M_0\rangle$ whose unobservable subnet is acyclic,*
the set of unobservable transitions T_u,
the observed word w.
Output*: $\mathscr{M}(w)$.*
1. *Let $w = \varepsilon$.*
2. *Let $\mathscr{M}(w) = \{(M_0, \vec{0})\}$.*
3. *Wait until a new transition t fires.*
4. *Let $w' = w$ and $w = w't$.*
5. *Let $\mathscr{M}(w) = \emptyset$.*
6. *For all M' such that $(M', y') \in \mathscr{M}(w')$, do*
6.1. *for all $e \in J_{\min}(M', t)$, do*
6.1.1. *let $M = M' + C_u \cdot e + C(\cdot, t)$,*
6.1.2. *for all y' such that $(M', y') \in \mathscr{M}(w')$, do*
6.1.2.1. *let $y = y' + e$,*
6.1.2.2. *let $\mathscr{M}(w) = \mathscr{M}(w) \cup \{(M, y)\}$.*
7. *Remove from $\mathscr{M}(w)$ any pair (M, y) for which there exists another pair (M', y') such that y covers y'.*
8. *Goto step 3.*

■

In simple words, the above algorithm that is based on Proposition 6.14 can be explained as follows. We assume that a certain word w (that is equal to the empty string at the initial step) has been observed. Then, a new observable transition t fires. We consider all basis markings at the observation w' before the firing of t, and we select among them those that may have allowed the firing of t, also taking into account that this may have required the firing of appropriate sequences of unobservable transitions. In particular, we focus on the minimal explanations, and thus on the corresponding minimal e-vectors (step 6.1). Finally, we update the set $\mathscr{M}(w)$ including all couples of new basis markings and j-vectors, taking into account that for each basis marking at w' it may correspond more than one j-vector.

Let us now introduce the following result that will be useful in the rest of the paper.

Definition 6.18. *Let $\langle N, M_0\rangle$ be a net system where $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$. Let $w \in T_o^*$ be an observed word. We denote*

$$\mathscr{M}_{basis}(w) = \{M \in \mathbb{N}^m \mid \exists y \in \mathbb{N}^{n_u} \text{ and } (M, y) \in \mathscr{M}(w)\}$$

the set of basis markings at w. Moreover, we denote as

$$\mathscr{M}_{basis} = \bigcup_{w \in T_o^*} \mathscr{M}_{basis}(w)$$

the set of all basis markings for any possible observation w. ■

Fact 6.19. *Given a bounded net system $\langle N, M_0\rangle$ with $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$, the set $\mathscr{M}_{basis}$ is* finite.

Proof: It follows from the fact that the set of basis markings is a subset of the reachability set, that is finite because the net system is bounded. □

We conclude this section showing that our approach based on basis markings is able to characterize completely the reachability set under partial observation.

We start with a result that characterizes the firing sequences. In the following theorem we show that a sequence $\tilde{\sigma}$ is consistent with observation w if and only if there exists an equivalent sequence (i.e., a sequence with the same observed sequence) that is the concatenation of two subsequences: the first one reaches a basis marking in $\mathscr{M}(w)$ and the second one contains only unobservable transitions.

Theorem 6.20. *Let us consider a net system $\langle N, M_0\rangle$ whose unobservable subnet is acyclic. There exists a sequence $\tilde{\sigma} \in T^*$ such that $M_0[\tilde{\sigma}\rangle \tilde{M}$ with observable projection $P_o(\tilde{\sigma}) = w$ and firing vector $\pi(\tilde{\sigma}) = \tilde{y}$* if and only if *there also exists a couple $(M, y) \in \mathscr{M}(w)$ and an unobservable sequence $\sigma'' \in T_u^*$ such that $M[\sigma''\rangle \tilde{M}$ and $\tilde{y} = \pi(w) + y + \pi(\sigma'')$.*

Proof: We prove this result by induction on the length of the observed string w.

(Basis step) For $w = \varepsilon$ the result obviously holds.

(Inductive step) Assume the result holds for $v \in T_o^*$. We prove it holds for $w = vt$ where $t \in T_o$.

(Only if). If there exists a sequence $\tilde{\sigma} \in T^*$ such that $M_0[\tilde{\sigma}\rangle \tilde{M}$ with $P_o(\tilde{\sigma}) = w$ and $\pi(\tilde{\sigma}) = \tilde{y}$ then there exist sequences σ' and σ'' such that

$$M_0[\sigma'\rangle M'[t\rangle M''[\sigma''\rangle \tilde{M}$$

where $P_o(\sigma') = v$, and $\sigma'' \in T_u^*$. By induction, there exists $(M, y) \in \mathscr{M}(v)$ such that

$$M_0[\sigma'_a\rangle M[\sigma'_b\rangle M'[t\rangle M''[\sigma''\rangle \tilde{M} \tag{6.1}$$

where $P_o(\sigma'_a) = v$, $\pi(\sigma'_a) = \pi(v) + y$ and $\sigma'_b \in T_u^*$.

Now, by definition of minimal explanation, there exists at least one sequence $\sigma'_c \in \Sigma_{\min}(M, t)$ such that $\pi(\sigma'_c) \leq \pi(\sigma'_b)$ and

$$M_0[\sigma'_a\rangle M[\sigma'_c\rangle M'_c[t\rangle M'_d \tag{6.2}$$

where $(M'_d, y + \pi(\sigma'_c)) \in \mathscr{M}(v\,t) = \mathscr{M}(w)$.

Furthermore, from (6.1) and (6.2) the following state equation holds

$$M'_d + C_u(\pi(\sigma'_b) - \pi(\sigma'_c)) = M'' \geq \vec{0}$$

Since the T_u-induced subnet is acyclic, by Theorem 3.2.(i) there exists a sequence $\sigma'_d \in T_u^*$ with $\pi(\sigma'_d) = \pi(\sigma'_b) - \pi(\sigma'_c)$ such that $M'_d[\sigma'_d\rangle M''$ and we can finally write

$$M_0[\sigma'_a\rangle M[\sigma'_c\rangle M'_c[t\rangle M'_d[\sigma'_d\rangle M''[\sigma''\rangle \tilde{M}$$

where $\sigma'_d\sigma'' \in T_u^*$. This proves the result.

(If). If there exists a couple $(M, y) \in \mathscr{M}(w)$ and a $\sigma'' \in T_u^*$ such that $M[\sigma''\rangle \tilde{M}$ and $\tilde{y} = \pi(w) + y + \pi(\sigma'')$ then there exists $\sigma' \in T^*$ such that $M_0[\sigma'\rangle M[\sigma''\rangle \tilde{M}$ with $P_o(\sigma') = v\,t = w$ and hence $M_0[\sigma\rangle \tilde{M}$ with $\sigma = \sigma'\sigma''$.

Note that the *if* statement holds even if the unobservable subnet is not acyclic. □

Based on the above theorem we can prove that, for any $w \in T_o^*$ the set of consistent markings $\mathscr{C}(w)$ may be characterized in terms of a number of linear algebraic constraints. In particular, the number of constraints depends on the number of basis markings at w.

Corollary 6.21. *Let us consider a net system $\langle N, M_0\rangle$ whose unobservable subnet is acyclic. For any $w \in T_o^*$ it holds that*

$$\mathscr{C}(w) = \{M \in \mathbb{N}^m \mid M = M_b + C_u \cdot y : y \geq \vec{0} \text{ and } M_b \in \mathscr{M}_{basis}(w)\}.$$

Proof: Trivially follows from Theorems 3.2 and 6.20. □

The above result is particularly important in the case of bounded net systems, because by Proposition 6.19, in such a case the number of constraints is finite for any observation w.

6.4 Diagnosis states

Let us consider a system modeled as a P/T net whose transitions set is partitioned into the set of observable and unobservable transitions, i.e., $T = T_o \cup T_u$.

Assume that a certain number of *anomalous* (or *fault*) behaviors may occur in the system. The occurrence of a fault behavior corresponds to the firing of an unobservable transition, but there may also be other transitions that are unobservable as well, but whose firing corresponds to regular behaviors. Then, assume that fault behaviors may be divided into r main

classes (*fault classes*), and we are not interested in distinguishing among fault events in the same class.

This can be easily modeled in Petri net terms assuming that the set of unobservable transitions is partitioned into two subsets, namely

$$T_u = T_f \cup T_{reg}$$

where T_f includes all fault transitions and T_{reg} includes all transitions relative to unobservable but regular events. The set T_f is further partitioned in r subsets, namely,

$$T_f = T_f^1 \cup T_f^2 \cup \ldots \cup T_f^r$$

where all transitions in the same subset correspond to the same fault class. We will say that the i-th fault has occurred when a transition in T_f^i has fired.

In the following subsection we introduce the definition of *diagnoser* and *diagnosis state*.

6.4.1 Basic definitions

Definition 6.22. *A* diagnoser *is a function* $\Delta : T_o^* \times \{T_f^1, T_f^2, \ldots, T_f^r\} \to \{0,1,2,3\}$ *that associates to each observation* w *and to each fault class* T_f^i, $i = 1, \ldots, r$, *a* diagnosis state.

$\Delta(w, T_f^i) = 0$ *if for all* $\sigma \in \mathscr{L}(w)$ *and for all* $t_f \in T_f^i$ *it holds* $t_f \notin \sigma$.

In such a case the i*th fault cannot have occurred, because none of the firing sequences consistent with the observation contains fault transitions of class* i.

$\Delta(w, T_f^i) = 1$ *if:*

(i) there exist $\sigma \in \mathscr{L}(w)$ *and* $t_f \in T_f^i$ *such that* $t_f \in \sigma$ *but*

(ii) for all $\sigma \in \mathscr{J}(w)$ *and for all* $t_f \in T_f^i$ *it holds that* $t_f \notin \sigma$.

In such a case a fault transition of class i *may have occurred but is not contained in any justification of* w.

$\Delta(w, T_f^i) = 2$ *if there exist* $\sigma, \sigma' \in \mathscr{J}(w)$ *such that:*

(i) there exists $t_f \in T_f^i$ *such that* $t_f \in \sigma$;

(ii) for all $t_f \in T_f^i$, $t_f \notin \sigma'$.

In such a case a fault transition of class i *is contained in one (but not in all) justification of* w.

$\Delta(w, T_f^i) = 3$ *if for all* $\sigma \in \mathscr{L}(w)$ *there exists* $t_f \in T_f^i$ *such that* $t_f \in \sigma$.

In such a case the i*th fault must have occurred, because all firable sequences consistent with the observation contain at least one fault transition of class* i. ■

The diagnosis states 1 and 2 correspond both to cases in which a fault may have occurred but has not necessarily occurred. The main reason to distinguish between them is the following. In the state 1 the observed behavior does not suggest that a fault has occurred because all minimal sequences leading to w are fault free. On the contrary, in the state 2 at least one of the justifications of the observed behavior contains one transition in the class.

Note that in practice diagnosis state 1 represents a situation that is common in many real applications. As an example, breaking a valve in a chemical plant may occur anytime thus all the states reached without a fault never fall into the diagnosis state 0 but in the diagnosis state 1.

Example 6.23. Consider the net system in Figure 6.1. Assume that two different fault behaviors (fault classes) may occur: (1) either a plate is moved to the lower line or a slab is moved to the upper line ($T_f^1 = \{\varepsilon_{11}, \varepsilon_{12}\}$); (2) a plate of a different type (e.g., different material, or different size) enters the upper line ($T_f^2 = \{\varepsilon_{13}\}$)

Finally, let all the other unobservable transitions belong to set T_{reg}, this is $T_{reg} = \{\varepsilon_3, \varepsilon_4, \ldots, \varepsilon_{10}\}$.

Consider $\omega = \varepsilon$. As discussed in Example 6.16, it holds $\mathscr{J}(\varepsilon) = \mathscr{L}(\varepsilon) = \{\varepsilon\}$. Then, we may observe that transition $\varepsilon_{13} \in T_f^2$ may fire at M_0, while the other fault transitions are not enabled at M_0. Therefore, we conclude that $\Delta(\varepsilon, T_f^1) = 0$ and $\Delta(\varepsilon, T_f^2) = 1$.

Now, let us consider $\omega = t_1$. As already discussed in Example 6.16, it is $\mathscr{J}(t_1) = \{\varepsilon\}$ thus no fault transition may be contained in a justification of w. On the contrary, all fault transitions are contained in at least one sequence in $\mathscr{L}(t_1)$. Thus, $\Delta(t_1, T_f^1) = \Delta(t_1, T_f^2) = 1$.

Let us now focus on the observation $\omega = t_1 t_2$. By looking at Example 6.4, it is easy to conclude that $\Delta(t_1 t_2, T_f^1) = \Delta(t_1 t_2, T_f^2) = 2$. In fact, all fault transitions are contained in at least one sequence in $\mathscr{J}(t_1 t_2)$, but there are also justifications of $t_1 t_2$ that do not contain fault transitions.

Finally, let $\omega = t_1 t_2 t_2$. In such a case it holds $\Delta(\omega, T_f^1) = \Delta(\omega, T_f^2) = 3$ because as it can be easily verified, all justifications of w contain transitions of both classes. ■

The on-line computation of the sets $\mathscr{L}(w)$ and $\mathscr{J}(w)$ may be computational demanding in large scale systems, thus in the following we suggest two alternative procedures to compute diagnosis states. These procedures are based on the notions of *minimal explanations*, *minimal e-vectors*, and *basis markings*, that are presented in the following two sections. Both procedures apply to net systems whose *unobservable subnet is acyclic*, and the second one also requires that the net system is *bounded*.

6.4.2 Characterization of diagnosis states

In this subsection we provide some results that enable us to characterize the diagnosis states starting from the knowledge of the set $\mathscr{M}(w)$. The following proposition allows us to estimate the value of a diagnosis state, reached after the observation of a word w, from the analysis of the set $\mathscr{M}(w)$ defined in Definition 7.8.

Proposition 6.24. *Consider an observed word $w \in T_o^*$.*

$\Delta(w, T_f^i) \in \{0,1\}$ *iff for all* $(M,y) \in \mathscr{M}(w)$ *and for all* $t_f \in T_f^i$ *it holds* $y(t_f) = 0$.

$\Delta(w, T_f^i) = 2$ *iff there exist* $(M,y) \in \mathscr{M}(w)$ *and* $(M',y') \in \mathscr{M}(w)$ *such that:*
(i) there exists $t_f \in T_f^i$ *such that* $y(t_f) > 0$,
(ii) for all $t_f \in T_f^i$, $y'(t_f) = 0$.

$\Delta(w, T_f^i) = 3$ *iff for all* $(M,y) \in \mathscr{M}(w)$ *there exists* $t_f \in T_f^i$ *such that* $y(t_f) > 0$.

Proof: It trivially follows from the definition of the diagnosis states and from Theorem 6.20.

□

From the only analysis of $\mathscr{M}(w)$ it is possible to determine the states 2 and 3, while to distinguish between states 0 and 1 further analysis is necessary. The following proposition shows how the states 0 and 1 can be distinguished with respect to the reachability of the unobservable net.

Proposition 6.25. *Consider an observed word $w \in T_o^*$ such that $\forall (M,y) \in \mathscr{M}(w)$ and $\forall\, t_f \in T_f^i$ it holds $y(t_f) = 0$.*

$\Delta(w, T_f^i) = 0$ *if $\forall$ $(M,y) \in \mathscr{M}(w)$ and $\forall\, t_f \in T_f^i$ there does not exist a sequence $\sigma \in T_u^*$ such that $M[\sigma\rangle$ and $t_f \in \sigma$.*

$\Delta(w, T_f^i) = 1$ *if $\exists$ at least one $(M,y) \in \mathscr{M}(w)$ and a sequence $\sigma \in T_u^*$ such that for at least one $t_f \in T_f^i$, $M[\sigma\rangle$ and $t_f \in \sigma$.*

Proof: It follows from the fact that by Proposition 6.24 $\Delta(w, T_f^i) \in \{0,1\}$ if all the minimal justifications of w contain no fault transition of class i. However, by Definition 7.11, the diagnosis state is equal to zero if at each basis marking M at w no fault transition of class i is enabled. On the contrary the diagnosis state is equal to one if at least one fault transition of class i is enabled at one basis marking M at w. □

If the *unobservable subnet is acyclic* the following proposition allows us to distinguish between the states 0 and 1 solving a trivial integer linear programming problem.

Proposition 6.26. *For a Petri net whose unobservable subnet is acyclic, let $w \in T_o^*$ be an observed word such that for all $(M,y) \in \mathscr{M}(w)$ it holds $y(t_f) = 0$ $\forall$ $t_f \in T_f^i$.*

Let us consider the constraint set

$$\mathscr{T}(M) = \begin{cases} M + C_u \cdot z \geq \vec{0}, \\ \sum\limits_{t_f \in T_f^i} z(t_f) > 0, \\ z \in \mathbb{N}^{n_u}. \end{cases} \tag{6.3}$$

$\Delta(w, T_f^i) = 0$ *if* $\forall\ (M, y) \in \mathscr{M}(w)$ *the constraint set* (6.3) *is not feasible.*

$\Delta(w, T_f^i) = 1$ *if* $\exists\ (M, y) \in \mathscr{M}(w)$ *such that the constraint set* (6.3) *is feasible.*

Proof: Follows from Proposition 6.25 and the fact that if the unobservable subnet is acyclic, the constraint set (6.3) characterizes the reachability set of the unobservable net. Thus, there exists a sequence containing a transition $t_f \in T_f^i$ firable at M on the unobservable subnet if and only if $\mathscr{T}(M)$ is feasible. □

Example 6.27. Consider again the net system in Figure 6.1.

Let $w = \varepsilon$. It holds $\mathscr{M}(\varepsilon) = \{(M_0, \vec{0})\}$ thus by Proposition 6.24, $\Delta(\varepsilon, T_f^1) = \Delta(\varepsilon, T_f^2) = \{0, 1\}$. To completely determine the diagnosis states we need to verify if the integer constraints set defined in Proposition 6.26 admit solutions. This is not true in the case of the first class, while it is the case for the second class. Therefore we conclude that $\Delta(\varepsilon, T_f^1) = 0$ and $\Delta(\varepsilon, T_f^2) = 1$, that is in accordance with Example 6.23. ■

6.5 A general approach to diagnosis

On the basis of the results presented in the previous Section 6.4.2, if the T_u-induced net is acyclic, diagnosis may be carried out by simply looking at the set $\mathscr{M}(w)$ for any observed word w and, should the diagnosis state be either 0 or 1, by additionally evaluating if the corresponding integer constraint set (6.3) admits a solution.

The main steps of the procedure are summarized in the following algorithm.

Algorithm 6.28. ***[A general approach to diagnosis]***

Input*: a Petri net* $\langle N, M_0 \rangle$ *whose unobservable subnet is acyclic,*
the set of unobservable transitions T_u*,*
the fault classes $\{T_f^i\}_{i=1,\ldots,r}$*,*
the observed word w*.*
Output*: the diagnosis states.*
1. *Let* $w = \varepsilon$*.*
2. *Let* $\mathscr{M}(w) = \{(M_0, \vec{0})\}$*.*
3. *Wait until a new transition* t *fires.*
4. *Let* $w' = w$ *and* $w = w't$*.*
5. *Let* $\mathscr{M}(w) = \emptyset$*.* ***[Computation of*** $\mathscr{M}(w)$***]***
6. *For all* M' *such that* $(M', y') \in \mathscr{M}(w')$ *, do*
6.1. *for all* $e \in J_{\min}(M', t)$*, do*
6.1.1. *let* $M = M' + C_u \cdot e + C(\cdot, t)$*,*
6.1.1.1. *for all* y' *such that* $(M', y') \in \mathscr{M}(w')$*, do*
6.1.1.1.1. *let* $y = y' + e$*,*
6.1.1.1.2. *let* $\mathscr{M}(w) = \mathscr{M}(w) \cup \{(M, y)\}$*.*
7. *For all* $i = 1, \ldots, r$*, do* ***[Computation of the diagnosis states]***

7.1. *if* $\forall\ (M,y) \in \mathscr{M}(w)$ *and* $\forall\, t_f \in T_f^i$ *it holds* $y(t_f) = 0$, *do*
  7.1.1. *if* $\forall\ (M,y) \in \mathscr{M}(w)\ \mathscr{T}(M)$ *is not feasible, do*
    7.1.1.1. *let* $\Delta(w, T_f^i) = 0$,
  7.1.2. *else*
    7.1.2.1. *let* $\Delta(w, T_f^i) = 1$,
7.2. *if* $\exists\ (M,y) \in \mathscr{M}(w)$ *and* $(M',y') \in \mathscr{M}(w)$ *such that:*
  (i) $\exists\, t_f \in T_f^i$ *such that* $y(t_f) > 0$, *(ii)* $\forall\, t_f \in T_f^i$, $y'(t_f) = 0$, *do*
  7.2.1. *let* $\Delta(w, T_f^i) = 2$,
7.3. *if* $\forall\ (M,y) \in \mathscr{M}(w)\ \exists\, t_f \in T_f^i$ *such that* $y(t_f) > 0$, *do*
  7.3.1. *let* $\Delta(w, T_f^i) = 3$.
8. *Goto step 3.*

■

In simple words, steps 1 to 6 coincide with those of Algorithm 6.17, while in step 7 we compute the diagnosis states using Propositions 6.24 and 6.26.

6.6 Diagnosis of bounded systems

The diagnosis approach presented in the previous sections applies both to bounded and unbounded Petri nets having the unobservable subnet acyclic. In this section we focus on *bounded* Petri nets and propose an original technique to design an observer to be used to solve the problem of diagnosis. Note that in this case the problem can be associated to the one solved in the automata case. However, as will be discussed later, our approach has the main advantage to reduce the computational complexity using the notion of basis marking.

6.6.1 Basis reachability graph

The proposed observer is based on the construction of a deterministic graph, that we call *basis reachability graph* (BRG). As discussed later, the main advantage of using BRG is that it enables us to move off-line most of the computations.

Definition 6.29. *The BRG is a deterministic graph that has as many nodes as the number of possible basis markings.*

To each node is associated a different basis marking and a row vector with as many entries as the number of fault classes. The entries of this vector may only take binary values: 1 if $\mathscr{T}(M)$ *is feasible, where* M *is the the basis marking, 0 otherwise.*

Arcs are labeled with observable transitions and e-vectors. More precisely, an arc exists from node containing the basis marking M *to node containing the basis marking* M' *if and only if there exists an observable transition* t *for which an explanation exists at* M *and the firing of* t *and one of its minimal explanations leads to* M'. *The arc going from* M *to* M' *is labeled* (t,e), *where* $e \in Y_{\min}(M,t)$ *and* $M' = M + C_u \cdot e + C(\cdot,t)$. ■

Note that the number of nodes of the BRG is always finite by Proposition 6.19. Moreover, the row vector of binary values associated to the nodes of the BRG allows us to distinguish between the diagnosis state 1 or 0.

The following algorithm provides a systematic procedure to compute the BRG. Also in this case the Petri net used as input must have the T_u-induced subnet acyclic.

Algorithm 6.30. ***[Computation of the BRG]***

Input*: a Petri net $\langle N, M_0\rangle$ whose unobservable subnet is acyclic,*
the set of unobservable transitions T_u,
the fault classes $\{T_f^i\}_{i=1,\dots,r}$.
Output*: the BRG.*
1. *Label the initial node (M_0, x_0) where $\forall\, i = 1,\dots,r$,*

$$x_0(T_f^i) = \begin{cases} 1 & \text{if } \mathscr{T}(M_0) \text{ is feasible,} \\ 0 & \text{otherwise.} \end{cases}$$

Assign no tag to it.
2. *While nodes with no tag exist select a node with no tag and do*
2.1. *let M be the marking in the node,*
2.2. *for all t such that $Y_{\min}(M,t) \neq \emptyset$, do*
2.2.1. *for all $e \in Y_{\min}(M,t)$, do*
2.2.1.1. *let $M' = M + C_u \cdot e + C(\cdot, t)$,*
2.2.1.2. *if $\nexists$ a node with M', do*
2.2.1.2.1. *add a new node to the graph containing the couple (M', x') where $\forall\, i = 1,\dots,r$,*

$$x'(T_f^i) = \begin{cases} 1 & \text{if } \mathscr{T}(M') \text{ is feasible,} \\ 0 & \text{otherwise.} \end{cases}$$

2.3. *tag the node "old".*
3. *Remove all tags.*

■

The algorithm constructs the BRG starting from the initial node to which it corresponds the initial marking and the row vector of binaries defining which classes of faults may occur at M_0. Now, we consider all observable transitions for which a minimal explanation at M_0 exists. For any of these transitions $t \in T_o$ we compute the marking resulting from firing t at $M_0 + C_u \cdot e$, for any $e \in J_{\min}(M_0, t)$. If a marking not contained in the previous nodes is obtained, a new node is added to the graph. The arc going from the initial node to the new node is labeled (t, e). The procedure is iterated until all basis markings have been considered. Note that the notion of basis marking allows us to represent the state space in a compact manner. Thus, for any input Petri net, we need to enumerate only a subset of the reachability space.

The following example clarifies this.

Example 6.31. In Figure 6.3 we have reported the BRG of the net system in Figure 6.1. The notation used in Figure 6.3 is detailed in Tables 7.1 and 7.2.

M_0	[	1	0	0	0	0	0	0	0	0	0	0	0	1	1	$]^T$
M_1	[	0	2	0	0	0	0	1	0	0	0	0	0	1	1	$]^T$
M_2	[	0	1	0	0	0	0	0	0	0	0	0	1	1	1	$]^T$
M_3	[	0	0	0	0	0	0	1	0	0	0	0	1	0	1	$]^T$
M_4	[	0	1	0	0	0	0	1	0	0	0	0	1	0	0	$]^T$
M_5	[	0	2	0	0	0	0	0	0	0	0	0	1	1	0	$]^T$
M_6	[	0	0	0	0	0	0	0	0	0	0	0	2	0	0	$]^T$

Table 6.1: The basis markings of the BRG in Figure 6.3.

	ε_3	ε_4	ε_5	ε_6	ε_7	ε_8	ε_9	ε_{10}	ε_{11}	ε_{12}	ε_{13}
e_1	1	1	1	1	1	1	1	1	0	0	0
e_2	2	1	1	1	0	1	1	1	1	0	0
e_3	1	0	1	1	0	1	1	1	1	0	1
e_4	0	0	1	1	1	1	1	1	0	0	1

Table 6.2: The e-vectors of the BRG in Figure 6.3 (for clearness of presentation they are given in tabular form rather than in vectorial form).

Each node of the graph contains a different basis marking and a row vector that has two entries as the number of fault classes. As an example, the vector [0 1] is associated to M_0 because $\mathscr{T}(M_0)$ is feasible for the first fault class and unfeasible for the second fault class.

Then, there is an arc labeled $(t_1, \vec{0})$ from M_0 to M_1 because t_1 is enabled at M_0 and its firing leads to M_1. Note that in such a case $J_{\min}(M_0, t_1) = \{\vec{0}\}$.

Furthermore, there are four arcs exiting from node M_1, all labeled t_2 and containing minimal explanations e_1, e_2, e_3 and e_4, respectively, and leading to nodes containing markings M_2, M_3, M_4 and M_5, respectively.

Note that, by looking at the BRG we can also read all sequences of observable words, that is finite in the case at hand, and equal to $\{\varepsilon, t_1, t_1 t_2, t_1 t_2 t_2\}$.

Moreover note that for all the j-vectors in Table 7.2 the component associated to ε_{12} is equal to 0. This means that ε_{12} cannot be never detected. ■

6.6.2 Diagnosis using BRG

The following algorithm summarizes the main steps of the on-line diagnosis carried out by looking at the BRG. The Petri net used as input must have the T_u-induced subnet acyclic.

Algorithm 6.32. ***[Diagnosis using the BRG]***

Input*: a Petri net $\langle N, M_0 \rangle$ whose unobservable subnet is acyclic,*

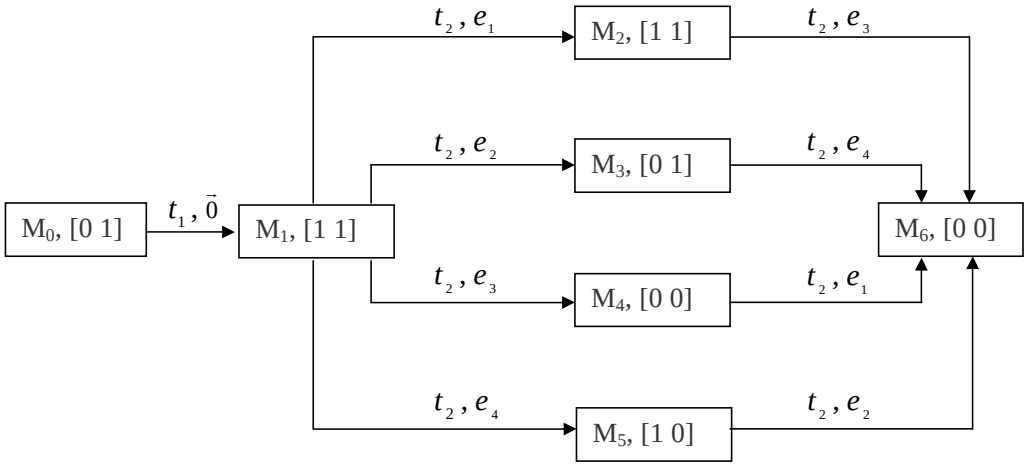


Figure 6.3: The BRG of the net in Figure 6.1.

the set of unobservable transitions T_u,
the fault classes $\{T_f^i\}_{i=1,\ldots,r}$,
the BRG.

***Output**: the diagnosis states.*

1. *Let* $w = \varepsilon$.
2. *Let* $\mathscr{M}(w) = \{(M_0, \vec{0})\}$.
3. *Wait until a new transition t fires.*
4. *Let* $w' = w$ *and* $w = w't$.
5. *Let* $\mathscr{M}(w) = \emptyset$, ***[Computation of*** $\mathscr{M}(w)$***]***
6. *For all nodes containing* M' *such that* $(M', y') \in \mathscr{M}(w')$, *do*
  6.1. *for all arcs exiting from the node containing* M', *do*
    6.1.1. *let* $M = M' + C_u \cdot e + C(\cdot, t)$,
    6.1.2. *for all* y' *such that* $(M', y') \in \mathscr{M}(w')$, *do*
      6.1.2.1. *let* $y = y' + e$,
      6.1.2.2. *let* $\mathscr{M}(w) = \mathscr{M}(w) \cup \{(M, y)\}$,
7. *for all* $i = 1, \ldots, r$, *do* ***[Computation of the diagnosis state]***
  7.1. *if* $\forall\ (M, y) \in \mathscr{M}(w)$ *and* $\forall\, t_f \in T_f^i$ *it holds* $y(t_f) = 0$, *do*
    7.1.1. *if* $\forall\ (M, y) \in \mathscr{M}(w)$ *it holds* $x(i) = 0$,
      where x is the binary vector in the node M, do
      7.1.1.1. *let* $\Delta(w, T_f^i) = 0$,
    7.1.2. *else*
      7.1.2.1. *let* $\Delta(w, T_f^i) = 1$,
  7.2. *if* $\exists\ (M, y) \in \mathscr{M}(w)$ *and* $(M', y') \in \mathscr{M}(w)$ *such that:*
    (i) $\exists t_f \in T_f^i$ *such that* $y(t_f) > 0$, *(ii)* $\forall\, t_f \in T_f^i$, $y'(t_f) = 0$, *do*
    7.2.1. *let* $\Delta(w, T_f^i) = 2$,
  7.3. *if* $\forall\ (M, y) \in \mathscr{M}(w)\ \exists t_f \in T_f^i$ *such that* $y(t_f) > 0$, *do*
    7.3.1. *let* $\Delta(w, T_f^i) = 3$.
8. *Goto step 3.*

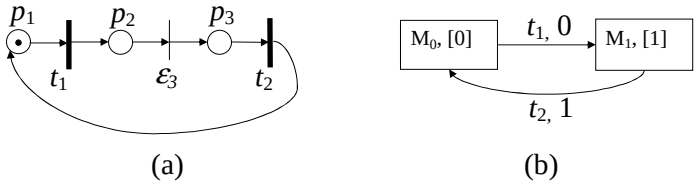


Figure 6.4: The Petri net of the Example 6.34 (a) and its BRG (b).

■

Steps 1 to 6 of Algorithm 6.32 enable us to compute the set $\mathscr{M}(w)$. When no event is observed, namely $w = \varepsilon$, then $\mathscr{M}(w) = \{(M_0, \vec{0})\}$. Now, assume that a transition t is observed. We include in the set $\mathscr{M}(t)$ all couples (M, y) such that an arc labeled t exits from the initial node and ends in a node containing the basis marking M. The corresponding value of y is equal to the e-vector in the arc going from M_0 to M, being $\vec{0}$ the j-vector relative to M_0. In general, if w' is the actual observation, and a new transition t fires, we consider all couples $(M', y') \in \mathscr{M}(w')$ and all nodes that can be reached from M' with an arc labeled t. Let M be the basis marking of the generic resulting node. We include in $\mathscr{M}(w) = \mathscr{M}(w't)$ all couples (M, y), where for any M, y is equal to the sum of y' plus the e-vector labeling the arc from M' to M.
Step 7 of Algorithm 6.32 computes the diagnosis state. Let us consider the generic ith fault class. If $\forall (M, y) \in \mathscr{M}(w)$ and $\forall\, t_f \in T_f^i$ it holds $y(t_f) = 0$, we have to check the ith entry of all the binary row vectors associated to the basis markings M, such that $(M, y) \in \mathscr{M}(w)$. If these entries are all equal to 0, we set $\Delta(w, T_f^i) = 0$, otherwise we set $\Delta(w, T_f^i) = 1$. Finally, steps 7.1 and 7.3 are exactly the same of Algorithm 6.28.

Example 6.33. Consider the BRG in Figure 6.3 relative to the net system in Figure 6.1, where $T_f^1 = \{\varepsilon_{11}, \varepsilon_{12}\}$ and $T_f^2 = \{\varepsilon_{13}\}$.

Let $w = \varepsilon$. By looking at the BRG we establish that $\Delta(\varepsilon, T_f^1) = 0$ and $\Delta(\varepsilon, T_f^2) = 1$, because the first entry of the row vector in the node M_0 is 0, while its second entry is equal to 1.

Now, let us consider $w = t_1 t_2$. In such a case $\mathscr{M}(w) = \{(M_2, y_1), (M_3, y_2), (M_4, y_3), (M_5, y_4)\}$, where

$$y_1 = \vec{0} + e_1 = e_1,$$
$$y_2 = \vec{0} + e_2 = e_2,$$
$$y_3 = \vec{0} + e_3 = e_3,$$
$$y_4 = \vec{0} + e_4 = e_4.$$

It holds $\Delta(t_1 t_2, T_f^1) = 2$ being $y_2(\varepsilon_{11}) = y_3(\varepsilon_{11}) = 1$ and $y_1(\varepsilon_{11}) = y_4(\varepsilon_{11}) = y_j(\varepsilon_{12}) = 0$ for $j = 1, 4$. Analogously, $\Delta(t_1 t_2, T_f^2) = 2$ being $y_3(\varepsilon_{13}) = y_4(\varepsilon_{13}) = 1$ and $y_1(\varepsilon_{13}) = y_2(\varepsilon_{13}) = 0$.

Finally, for $w = t_1 t_2 t_2$ it holds $\Delta(t_1 t_2 t_2, T_f^i) = 3$ for $i = 1, 2$. In fact $\mathscr{M}(w) = \{(M_6, y_5), (M_6, y_6)\}$, where $y_5 = y_1 + e_3 = y_3 + e_1$, $y_6 = y_2 + e_4 = y_4 + e_2$, and $y_5(\varepsilon_{11}) = y_6(\varepsilon_{11}) = 1$, $y_5(\varepsilon_{13}) = y_6(\varepsilon_{13}) = 1$.

■

In the previous example we considered a net that does not contain repetitive sequences and the corresponding BRG is acyclic. In such a case, we could also determine off-line the j-vector associated to each basis marking.

However, our procedure applies to the more general case of bounded Petri nets with repetitive sequences, to which a cyclic BRG corresponds. For this class of nets we need to compute the j-vector of a basis marking on-line as shown in the following example.

Example 6.34. Consider the bounded Petri net shown in Figure 6.4(a), where $T_o = \{t_1, t_2\}$ and $T_u = \{\varepsilon_3\}$. We assume that the only fault that can occur is $T_1^f = \{\varepsilon_3\}$. This net contains the repetitive sequence $t_1\varepsilon_3 t_2$ that can fire infinitely often, hence its BRG is cyclic as shown in Figure 6.34(b), where the basis markings are $M_0 = [1\ 0\ 0]^T$ and $M_1 = [0\ 1\ 0]^T$.

It is easy to verify that in this case we cannot associate off-line j-vectors to basis markings. In fact when the observed word is $w = \varepsilon$ the j-vector associated to node $M_0, [0]$ is $[0]$, hence the diagnosis state is $\Delta(\varepsilon, T_1^f) = 0$. On the contrary, after $w = t_1 t_2$ fires we reach the same basis marking $M_0, [0]$ but now its j-vector is $[1]$, and the diagnosis state is changed to $\Delta(t_1 t_2, T_1^f) = 3$.

■

6.7 Remark

In this section we briefly remark on the hypothesis of acyclicity made in this chapter.

This assumption, that is analogous to the classical hypothesis of acyclicity of the subgraph in the theory of automata, allows us to:

- study the reachability of the unobservable subnet with the state equation;
- devise an easy algorithm for the computation of the firing vectors relative to justifications.

In particular, the first item implies that we can distinguish between the diagnosis states 0 and 1 in an efficient way (solving an integer programming problem). However, the hypothesis of acyclicity could also be removed.

In this case

- Algorithm 6.9 for the computation of $Y_{\min}(M, t)$,
- Proposition 6.14 and Algorithm 6.17 for the computation of basis markings and j-vectors,
- Theorem 6.20, Proposition 6.26,
- Algorithms 6.28, 6.30, 6.32,

do not apply anymore.

Chapter 7

Diagnosis of discrete event systems using labeled Petri nets

Summary

In this chapter we focus on the diagnosis of labeled PNs, i.e., nets where two or more transitions may share the same label. In particular we present an approach for diagnosis. The proposed procedure is based on results on labeled PNs and allows us to also consider events that are indistinguishable, namely events that produce an output signal that is observable, but that is common to other events. Four diagnosis states are defined, each one corresponding to a different degree of alarm. A procedure is given to compute the actual diagnosis state given the current observation. We also show that, in the case of bounded net systems, the most burdensome part of the procedure can be moved off-line defining a particular graph, that we call *Basis Reachability Graph*.

7.1 Consistent markings and sequences

In this section we introduce some notations and definitions that we will use in the rest of the chapter.

A *labeling function* $\varphi : T \to L \cup \{\varepsilon\}$ assigns to each transition $t \in T$ either a symbol from a given alphabet L or the empty string ε.

We denote as T_u the set of transitions whose label is ε, i.e., $T_u = \{t \in T \mid L(t) = \varepsilon\}$. Transitions in T_u are called *unobservable* or *silent*.

We denote as T_o the set of transitions labeled with a symbol in L. Transitions in T_o are called *observable* because when they fire their label can be observed. Note that in this chapter we assume that the same label $l \in L$ can be associated to more than one transition. In particular, two transitions $t_1, t_2 \in T_o$ are called *indistinguishable* if they share the same label, i.e., $\varphi(t_1) = \varphi(t_2)$. The set of transitions sharing the same label l are denoted as T_l.

In the following we denote as C_u (C_o) the restriction of the incidence matrix to T_u (T_o) and denote as n_u and n_o, respectively, the cardinality of the above sets.

Moreover, given a sequence $\sigma \in T^*$, $P_u(\sigma)$, resp., $P_o(\sigma)$, given in Definition 6.1, denotes the projection of σ over T_u, resp., T_o.

We denote as w the word of events associated to the sequence σ, i.e., $w = P_o(\sigma)$. Note that the length of a sequence σ (denoted $|\sigma|$) is always greater than or equal to the length of the corresponding word w (denoted $|w|$). In fact, if σ contains k' transitions in T_u then $|\sigma| = k' + |w|$.

Definition 7.1. *Let $\langle N, M_0\rangle$ be a labeled net system with labeling function $\varphi : T \to L \cup \{\varepsilon\}$, where $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$. Let $w \in L^*$ be an observed word. We define*

$$\mathscr{S}(w) = \{\sigma \in L(N, M_0) \mid P_o(\sigma) = w\}$$

the set of firing sequences consistent *with $w \in L^*$, and*

$$\mathscr{C}(w) = \{M \in R(N, M_0) \mid \exists \sigma \in T^* : P_o(\sigma) = w \wedge M_0[\sigma\rangle M\}$$

the set of markings consistent *with $w \in L^*$.* ■

In plain words, given an observation w, $\mathscr{S}(w)$ is the set of sequences that may have fired, while $\mathscr{C}(w)$ is the set of markings in which the system may actually be.

Example 7.2. Let us consider the PN in Figure 7.1. Let us assume $T_o = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ and $T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}\}$, where for a better understanding unobservable transitions have been denoted ε_i rather than t_i. The labeling function is defined as follows: $\varphi(t_1) = a$, $\varphi(t_2) = \varphi(t_3) = b$, $\varphi(t_4) = \varphi(t_5) = c$, $\varphi(t_6) = \varphi(t_7) = d$.

First let us consider $w = acd$. The set of firing sequences that are consistent with w is $\mathscr{S}(w) = \{t_1 t_5 t_6, t_1 t_5 \varepsilon_{12} \varepsilon_{13} t_7\}$, and the set of markings consistent with w is $\mathscr{C}(w) = \{[0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T\}$. Thus two different firing sequences may have fired (the second one also involving silent transitions), but they both lead to the same marking.

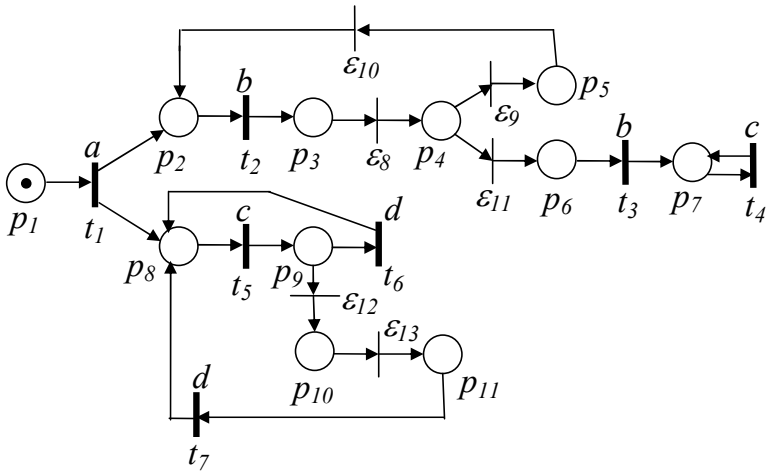


Figure 7.1: A PN system with faults ε_{11} and ε_{12}.

On the contrary, different markings can be reached if we consider $w = ab$. In particular, the set of firing sequences that are consistent with w is $\mathscr{S}(w) = \{t_1t_2, t_1t_2\varepsilon_8, t_1t_2\varepsilon_8\varepsilon_9, t_1t_2\varepsilon_8\varepsilon_9\varepsilon_{10}, t_1t_2\varepsilon_8\varepsilon_{11}\}$, and the set of markings consistent with w is $\mathscr{C}(w) = \{[0\,0\,1\,0\,0\,0\,0\,1\,0\,0\,0]^T, [0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0]^T, [0\,0\,0\,0\,1\,0\ 0\,1\,0\,0\,0]^T, [0\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0]^T, [0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0]^T\}$. ■

7.2 Minimal explanations and minimal e-vectors

In Chapter 6 we have introduced the definitions of minimal explanation and minimal e-vectors for free-label Petri nets (Definition 6.6). In the case of labeled PNs what we observe are symbols in L. Thus, it is useful to compute the following sets.

Definition 7.3. *Given a marking M and an observation $l \in L$, we define the set of* minimal explanations of l at M *as*

$$\hat{\Sigma}_{\min}(M,l) = \cup_{t\in T_l} \cup_{\sigma\in\Sigma_{\min}(M,t)} \{(t,\sigma)\},$$

i.e., the set of pairs (transition labeled l; corresponding minimal explanation), and we define the set of minimal e-vectors of l at M *as*

$$\hat{Y}_{\min}(M,l) = \cup_{t\in T_l} \cup_{e\in Y_{\min}(M,t)} \{(t,e)\},$$

i.e., the set of pairs (transition labeled l; corresponding minimal e-vector). ■

Obviously, $\hat{\Sigma}_{\min}(M,l)$ and $\hat{Y}_{\min}(M,l)$ are a generalization of the sets of minimal explanations and minimal e-vectors introduced for unlabeled PNs with unobservable transitions. Moreover, in the above sets $\hat{\Sigma}_{\min}(M,l)$ and $\hat{Y}_{\min}(M,l)$ different sequences σ and different e-vectors e, respectively, are associated in general to the same $t \in T_l$.

Example 7.4. Let us consider the PN in Figure 7.1 previously introduced in Example 7.2.

Let us consider $M = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$ and $l = b$. The set of minimal explanations of b at M is $\hat{\Sigma}_{\min}(M,b) = \{(t_2,\varepsilon_8\varepsilon_9\varepsilon_{10}),(t_3,\varepsilon_8\varepsilon_{11})\}$. The set of minimal e-vectors is $\hat{Y}_{\min}(M,b) = \{(t_2,[1\ 1\ 1\ 0\ 0\ 0]),(t_3,[1\ 0\ 0\ 1\ 0\ 0])\}$. ■

7.3 Basis markings and j-vectors

In Chapter 6 the notions of *basis markings* and *j-vectors* have been defined for unlabeled PNs where some transitions are unobservable.

Here we basically use the same definitions, but with a slight but crucial difference: in the previous chapter the unique form of nondeterminism is the presence of unobservable transitions, in this chapter we have two forms of nondeterminism due to the presence of unobservable and indistinguishable transitions. In fact, in the case of labeled PNs the observation w is a sequence of labels, namely $w \in L^*$. In general several sequences $\sigma_o \in T_o^*$ may correspond to the same w, i.e., there are several sequences of observable transitions such that $P_o(\sigma_o) = w$ that may have actually fired. Moreover, in general, to any of such sequences σ_o a different sequence of unobservable transitions interleaved with it is necessary to make it firable at the initial marking. Thus, again we need to define sets of couples. In particular, we introduce the following definition of couples (sequence of transitions in T_o labeled w; corresponding *justification*).

Definition 7.5. *Let $\langle N, M_0\rangle$ be a net system with labeling function $\varphi : T \to L \cup \{\varepsilon\}$, where $N = (P,T,Pre,Post)$ and $T = T_o \cup T_u$. Let $w \in L^*$ be a given observation. We define*

$$\begin{aligned}\hat{\mathscr{J}}(w) \quad &= \{\,(\sigma_o,\sigma_u),\ \sigma_o \in T_o^*,\ P_o(\sigma_o) = w,\ \sigma_u \in T_u^* \mid [\exists \sigma \in \mathscr{S}(w)\ :\ \sigma_o = P_o(\sigma),\ \sigma_u = P_u(\sigma)] \wedge \\ &\quad \left[\nexists \sigma' \in \mathscr{S}(w) :\ \sigma_o = P_o(\sigma'),\ \sigma'_u = P_u(\sigma') \wedge\ \pi(\sigma'_u) \lneq \pi(\sigma_u)\right]\}\end{aligned}$$

the set of couples (sequence $\sigma_o \in T_o^*$ with $P_o(\sigma_o) = w$; corresponding justification *of w).*

Moreover, we define

$$\hat{Y}_{\min}(M_0,w) = \{(\sigma_o,y),\ \sigma_o \in T_o^*,\ P_o(\sigma_o) = w,\ y \in \mathbb{N}^{n_u} \mid \exists (\sigma_o,\sigma_u) \in \hat{\mathscr{J}}(w) : \pi(\sigma_u) = y\}$$

the set of couples (sequence $\sigma_o \in T_o^*$ with $P_o(\sigma_o) = w$; corresponding j-vector*).* ■

In simple words, $\hat{\mathscr{J}}(w)$ is the set of pairs whose first element is the sequence $\sigma_o \in T_o^*$ labeled w and whose second element is the corresponding sequence of unobservable transitions interleaved with σ_o whose firing enables σ_o and whose firing vector is minimal. The firing vectors of these sequences are called *j-vectors*.

Definition 7.6. *Let $\langle N, M_0\rangle$ be a net system with labeling function $\varphi : T \to L \cup \{\varepsilon\}$, where $N = (P,T,Pre,Post)$ and $T = T_o \cup T_u$. Let w be a given observation and $(\sigma_o,\sigma_u) \in \hat{\mathscr{J}}(w)$ be a generic couple (sequence of observable transitions labeled w; corresponding minimal justification). The marking*

$$M_b = M_0 + C_u \cdot y + C_o \cdot y', \qquad y = \pi(\sigma_u),\ \ y' = \pi(\sigma_o),$$

i.e., the marking reached firing σ_o interleaved with the minimal justification σ_u, is called basis marking *and y is called its* j-vector *(or* justification-vector*).* ■

Obviously, because in general more than one justification exists for a word w (the set $\hat{\mathscr{J}}(w)$ is generally not a singleton), the basis marking may be not unique as well.

Proposition 7.7. *Given a net system $\langle N, M_0\rangle$ with labeling function $\varphi : T \to L \cup \{\varepsilon\}$, where $N = (P, T, Pre, Post)$, $T = T_o \cup T_u$ and whose T_u-induced subnet is acyclic. Let $w = w'l$ be a given observation.*

The set $\hat{J}_{\min}(M_0, wl)$ is defined as:

$$\hat{J}_{\min}(M_0, wl) \subseteq \{(\sigma_o, y) \mid \sigma_o = \sigma'_0 t \wedge y = y' + e : \quad (\sigma'_o, y') \in \hat{J}_{\min}(M_0, w), (t, e) \in \hat{J}_{\min}(M'_b, l) \text{ and } \varphi(t) = l\},$$

where $M'_b = M_0 + C_u \cdot y' + C_o \cdot \sigma'_o$.

Proof: It trivially follows from Definitions 7.3, 7.5 and 7.6 and from the fact that, as shown in Theorem 6.20, in Petri nets where the unobservable subnet is acyclic basis markings completely characterized the set of consistent markings. □

Let us finally introduce the following definition.

Definition 7.8. *Let $\langle N, M_0\rangle$ be a net system with labeling function $\varphi : T \to L \cup \{\varepsilon\}$, where $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$. Let $w \in L^*$ be an observed word. We define*

$$\mathscr{M}(w) = \{(M, y) \mid (\exists \sigma \in \mathscr{S}(w) : M_0[\sigma\rangle M) \wedge (\exists (\sigma_o, \sigma_u) \in \hat{\mathscr{J}}(w) : \sigma_o = P_o(\sigma), \sigma_u = P_u(\sigma), \ y = \pi(\sigma_u))\}$$

the set of couples (basis marking; relative j-vector) that are consistent *with $w \in L^*$.* ■

Note that the set $\mathscr{M}(w)$ does not keep into account the sequences of observable transitions that may have actually fired. It only keeps track of the basis markings that can be reached and of the sequences of unobservable transitions that have fired to reach them. Indeed, this is the information really significant when performing diagnosis.

Example 7.9. Let us consider the PN in Figure 7.1 previously introduced in Example 7.2.

Let us assume $w = acd$. The set of justifications is $\hat{\mathscr{J}}(w) = \{(t_1 t_5 t_6, \varepsilon), (t_1 t_5 t_7, \varepsilon_{12}\varepsilon_{13})\}$ and the set of j-vectors is $\hat{J}_{\min}(M_0, w) = \{(t_1 t_5 t_6, \vec{0}), (t_1 t_5 t_7, [0\ 0\ 0\ 0\ 1\ 1]^T)\}$. The above j-vectors lead to the same basis marking $M_b = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$ thus $\mathscr{M}(w) = \{(M_b, \vec{0}), (M_b, [0\ 0\ 0\ 0\ 1\ 1]^T)\}$.

Now, let us consider $w = ab$. In this case $\hat{\mathscr{J}}(w) = \{(t_1 t_2, \varepsilon)\}$, $\hat{J}_{\min}(M_0, w) = \{(t_1 t_2, \vec{0})\}$ and the basis marking is the same as in the previous case, namely $M_b = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$, thus $\mathscr{M}(w) = \{(M_b, \vec{0})\}$. ■

The set $\mathscr{M}(w)$ can be easily constructed using the following algorithm. Note that, the Petri net used as input must have the unobservable subnet acyclic, since this assumption is necessary for the computation of the minimal explanations.

Algorithm 7.10. *[Computation of the basis markings and j-vectors]*

1. *Let* $w = \varepsilon$.
2. *Let* $\mathscr{M}(w) = \{(M_0, \vec{0})\}$.
3. *Wait until a new label* l *is observed.*
4. *Let* $w' = w$ *and* $w = w'l$.
5. *Let* $\mathscr{M}(w) = \emptyset$.
6. *For all* M' *such that* $(M', y') \in \mathscr{M}(w')$ *, do*
  6.1. *for all* $t \in T_l$*, do*
    6.1.1. *for all* $e \in Y_{\min}(M', t)$*, do*
    6.1.1.1. *let* $M = M' + C_u \cdot e + C(\cdot, t)$,
    6.1.1.2. *for all* y' *such that* $(M', y') \in \mathscr{M}(w')$*, do*
      6.1.2.1. *let* $y = y' + e$,
      6.1.2.2. *let* $\mathscr{M}(w) = \mathscr{M}(w) \cup \{(M, y)\}$.
7. *Remove from* $\mathscr{M}(w)$ *any pair* (M, y) *for which there exists another pair* (M', y') *such that* y *covers* y'.
8. *Goto step 3.*

■

In simple words, the above algorithm can be explained as follows. We assume that a certain word w (that is equal to the empty string at the initial step) has been observed. Then, a new observable t fires and we observe its label $\varphi(t)$ (e.g., l). We consider all basis markings at the observation w' before the firing of t, and we select among them those that may have allowed the firing of at least one transition $t \in T_l$, also taking into account that this may have required the firing of appropriate sequences of unobservable transitions. In particular, we focus on the minimal explanations, and thus on the corresponding minimal e-vectors (step 6.1.1). Finally, we update the set $\mathscr{M}(w)$ including all couples of new basis markings and j-vectors, taking into account that for each basis marking at w' it may correspond more than one j-vector.

Following definition 6.18 we denote $\mathscr{M}_{basis}(w)$ the set of basis markings at w and $\mathscr{M}_{basis}$ the set of all basis markings for any observation w. Moreover, following Corollary 6.20, given a Petri net whose unobservable subnet is acyclic, for any $w \in T_o^*$, the set of consistent markings can be computed using the set of basis marking at w:

$$\mathscr{C}(w) = \{M \in \mathbb{N}^m \mid M = M_b + C_u \cdot y : y \geq \vec{0} \text{ and } M_b \in \mathscr{M}_{basis}(w)\}.$$

7.4 Diagnosis using Petri nets

Assume that the set of unobservable transitions is partitioned in two subsets, namely $T_u = T_f \cup T_{reg}$ where T_f includes all fault transitions (modeling anomalous or fault behavior), while T_{reg} includes all transitions relative to unobservable but regular events. The set T_f is further partitioned into r different subsets T_f^i, where $i = 1, \ldots, r$, that model the different fault classes.

The following definition introduces the notion of *diagnoser* for labeled PNs.

Definition 7.11. *A* diagnoser *is a function* $\Delta : L^* \times \{T_f^1, T_f^2, \ldots, T_f^r\} \to \{0,1,2,3\}$ *that associates to each observation* $w \in L^*$ *and to each fault class* T_f^i, $i = 1, \ldots, r$, *a* diagnosis state.

- $\Delta(w, T_f^i) = 0$ *if for all* $\sigma \in \mathscr{S}(w)$ *and for all* $t_f \in T_f^i$ *it holds* $t_f \notin \sigma$.

  In such a case the i*th fault cannot have occurred, because none of the firing sequences consistent with the observation contains fault transitions of class* i.

- $\Delta(w, T_f^i) = 1$ *if:*

  (i) there exist $\sigma \in \mathscr{S}(w)$ *and* $t_f \in T_f^i$ *such that* $t_f \in \sigma$ *but*

  (ii) for all $(\sigma_o, \sigma_u) \in \hat{\mathscr{L}}(w)$ *and for all* $t_f \in T_f^i$ *it holds that* $t_f \notin \sigma_u$.

  In such a case a fault transition of class i *may have occurred but is not contained in any justification of* w.

- $\Delta(w, T_f^i) = 2$ *if there exist* $(\sigma_o, \sigma_u), (\sigma'_o, \sigma'_u) \in \hat{\mathscr{L}}(w)$ *such that*

  (i) there exists $t_f \in T_f^i$ *such that* $t_f \in \sigma_u$*;*

  (ii) for all $t_f \in T_f^i$, $t_f \notin \sigma'_u$.

  In such a case a fault transition of class i *is contained in one (but not in all) justification of* w.

- $\Delta(w, T_f^i) = 3$ *if for all* $\sigma \in \mathscr{S}(w)$ *there exists* $t_f \in T_f^i$ *such that* $t_f \in \sigma$.

  In such a case the i*th fault must have occurred, because all firable sequences consistent with the observation contain at least one fault in* T_f^i. ■

Example 7.12. Let us consider the PN in Figure 7.1 previously introduced in Example 7.2.

Let $T_f = \{\varepsilon_{11}, \varepsilon_{12}\}$. Assume that the two fault transitions belong to different fault classes, i.e., $T_f^1 = \{\varepsilon_{11}\}$ and $T_f^2 = \{\varepsilon_{12}\}$.

Let us observe $w = acd$. Then $\Delta(w, T_f^1) = 0$ and $\Delta(w, T_f^2) = 2$, being $\hat{\mathscr{L}}(w) = \{(t_1 t_5 t_6, \varepsilon), (t_1 t_5 t_7, \varepsilon_{12}\varepsilon_{13})\}$ and $\mathscr{S}(w) = \{t_1 t_5 t_6, t_1 t_5 \varepsilon_{12}\varepsilon_{13} t_7\}$.

Now, let us consider $w = ab$. In this case $\Delta(w, T_f^1) = 1$ and $\Delta(w, T_f^2) = 0$, being $\hat{\mathscr{L}}(w) = \{(t_1 t_2, \varepsilon)\}$ and $\mathscr{S}(w) = \{t_1 t_2, t_1 t_2 \varepsilon_8, t_1 t_2 \varepsilon_8 \varepsilon_9, t_1 t_2 \varepsilon_8 \varepsilon_9 \varepsilon_{10}, t_1 t_2 \varepsilon_8 \varepsilon_{11}\}$. ■

The following two results proved in Chapter 6 for unlabeled PNs still hold in the case of labeled PNs.

Proposition 7.13. *Consider an observed word* $w \in L^*$.

- $\Delta(w, T_f^i) \in \{0,1\}$ *iff for all* $(M, y) \in \mathscr{M}(w)$ *and for all* $t_f \in T_f^i$ *it holds* $y(t_f) = 0$.

- $\Delta(w, T_f^i) = 2$ *iff there exist* $(M, y) \in \mathscr{M}(w)$ *and* $(M', y') \in \mathscr{M}(w)$ *such that:*
  (i) there exists $t_f \in T_f^i$ *such that* $y(t_f) > 0$,
  (ii) for all $t_f \in T_f^i$, $y'(t_f) = 0$.
- $\Delta(w, T_f^i) = 3$ *iff for all* $(M, y) \in \mathscr{M}(w)$ *there exists* $t_f \in T_f^i$ *such that* $y(t_f) > 0$.

The following proposition shows how to distinguish among states 0 and 1.

Proposition 7.14. *For a PN whose unobservable subnet is acyclic, let* $w \in L^*$ *be an observed word such that for all* $(M, y) \in \mathscr{M}(w)$ *it holds* $y(t_f) = 0 \ \forall \ t_f \in T_f^i$.

Let us consider the constraint set

$$\mathscr{T}(M) \quad = \quad \begin{cases} M + C_u \cdot z \geq \vec{0}, \\ \sum\limits_{t_f \in T_f^i} z(t_f) > 0, \\ z \in \mathbb{N}^{n_u}. \end{cases} \tag{7.1}$$

- $\Delta(w, T_f^i) = 0$ *if* $\forall$ $(M, y) \in \mathscr{M}(w)$ *the constraint set* (7.1) *is not feasible.*
- $\Delta(w, T_f^i) = 1$ *if* $\exists$ $(M, y) \in \mathscr{M}(w)$ *such that the constraint set* (7.1) *is feasible.*

On the basis of the above two results, if the T_u-induced net is acyclic, diagnosis may be carried out by simply looking at the set $\mathscr{M}(w)$ for any observed word w and, should the diagnosis state be either 0 or 1, by additionally evaluating if the corresponding integer constraint set (7.1) admits a solution.

Example 7.15. Let us consider the PN in Figure 7.1 where $T_f^1 = \{\varepsilon_{11}\}$ and $T_f^2 = \{\varepsilon_{12}\}$.

Let $w = acd$. It is $\mathscr{M}(w) = \{(M_b, \vec{0}), (M_b, [0\ 0\ 0\ 0\ 1\ 1]^T)\}$, where $M_b = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$ has been computed in Example 7.9. It is $\Delta(w, T_f^1) = 0$ being $\mathscr{T}(M_b)$ not feasible.

Let $w = ab$. In this case $\mathscr{M}(w) = \{(M_b, \vec{0})\}$, where $M_b = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$ as in the previous case. Being $\mathscr{T}(M_b)$ feasible only for the fault class T_f^1 it holds $\Delta(w, T_f^1) = 1$ and $\Delta(w, T_f^2) = 0$.

■

7.5 Basis Reachability Graph

In Chapter 6 we have shown that in the case of bounded PNs a useful tool to perform diagnosis is the *Basis Reachability Graph* (BRG). In this section we show how the BRG can still be defined in the case of arbitrary labeled PNs.

The BRG is a deterministic graph that has as many nodes as the number of possible basis markings. To each node is associated a different basis marking M and a row vector with as

many entries as the number of fault classes. The entries of this vector may only take binary values: 1 if $\mathscr{T}(M)$ is feasible, 0 otherwise.

Arcs are labeled with observable events in L and e-vectors. More precisely, an arc exists from a node containing the basis marking M to a node containing the basis marking M' if and only if there exists a transition t for which an explanation exists at M and the firing of t and one of its minimal explanations leads to M'. The arc going from M to M' is labeled $(\varphi(t), e)$, where $e \in Y_{\min}(M,t)$ and $M' = M + C_u \cdot e + C(\cdot,t)$.

Note that the number of nodes of the BRG is always finite being the set of basis markings a subset of the set of reachable markings, that is finite being the net bounded. Moreover, the row vector of binary values associated to the nodes of the BRG allows us to distinguish between the diagnosis state 1 or 0.

The main steps for the computation of the BRG in the case of labeled PNs are summarized in the following algorithm.

Algorithm 7.16. *[Computation of the BRG]*

1. *Label the initial node* (M_0, x_0) *where* $\forall i = 1,\ldots,r$,
$$x_0(T_f^i) = \begin{cases} 1 & \text{if } \mathscr{T}(M_0) \text{ is feasible,} \\ 0 & \text{otherwise.} \end{cases}$$
Assign no tag to it.
2. *While nodes with no tag exist, select a node with no tag and do*
2.1. *let* M *be the marking in the node* (M,x),
2.2. *for all* $l \in L$
2.2.1. *for all* $t : L(t) = l \wedge Y_{\min}(M,t) \neq \emptyset$, *do*
for all $e \in Y_{\min}(M,t)$, *do*
let $M' = M + C_u \cdot e + C(\cdot,t)$,
if $\nexists$ *a node with the first element of the couple equal to* M', *do*
add a new node to the graph containing (M', x') *where* $\forall i = 1,\ldots,r$,
$$x'(T_f^i) = \begin{cases} 1 & \text{if } \mathscr{T}(M') \text{ is feasible,} \\ 0 & \text{otherwise.} \end{cases}$$
and arc (l,e) *from* (M,x) *to* (M',x')
else
add arc (l,e) *from* (M,x) *to* (M',x') *if it does not exist yet*
2.3. *tag the node "old".*
3. *Remove all tags.*

■

The algorithm constructs the BRG starting from the initial node to which it corresponds the initial marking and a binary vector defining which classes of faults may occur at M_0. Now, we consider all the labels $l \in L$ such that there exists a transition t with $L(t) = l$ for which a minimal explanation at M_0 exists. For any of these transitions we compute the marking resulting from firing t at $M_0 + C_u \cdot e$, for any $e \in Y_{\min}(M_0,t)$. If a couple (marking, binary vector) not contained in the previous nodes is obtained, a new node is added to the graph.

M_0	[	1	0	0	0	0	0	0	0	0	0	$]^T$
M_1	[	0	1	0	0	0	0	0	1	0	0	$]^T$
M_2	[	0	1	0	0	0	0	0	0	1	0	$]^T$
M_3	[	0	0	1	0	0	0	0	1	0	0	$]^T$
M_4	[	0	0	1	0	0	0	0	0	1	0	$]^T$
M_5	[	0	0	0	0	0	0	1	1	0	0	$]^T$
M_6	[	0	0	0	0	0	0	1	0	1	0	$]^T$

Table 7.1: The basis markings of the BRG in Figure 7.2.

	ε_8	ε_9	ε_{10}	ε_{11}	ε_{12}	ε_{13}
e_1	0	0	0	0	1	1
e_2	1	1	1	0	0	0
e_3	1	0	0	1	0	0

Table 7.2: The e-vectors of the BRG in Figure 7.2 (given in tabular form).

The arc going from the initial node to the new node is labeled (l, e). The procedure is iterated until all basis markings have been considered.

An example clarifies the previous concepts.

Example 7.17. Let us consider the PN in Figure 7.1, where $T_o = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$, $T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}\}$, $T_f^1 = \{\varepsilon_{11}\}$ and $T_f^2 = \{\varepsilon_{12}\}$. The labeling function is defined as follows: $\varphi(t_1) = a$, $\varphi(t_2) = \varphi(t_3) = b$, $\varphi(t_4) = \varphi(t_5) = c$, $\varphi(t_6) = \varphi(t_7) = d$.

The BRG relative to the net system in Figure 7.1 is shown in Figure 7.2. The notation used in Figure 7.2 is detailed in Tables 7.1 and 7.2.

Each node contains a different basis marking and a binary row vector of dimension two, being two the number of fault classes.

As an example, the binary vector [0 0] is associated to M_0 because $\mathscr{T}(M_0)$ is not feasible for both fault classes. From node M_0 to node M_1 there is one arc labeled a and with the null vector as minimal explanation. The node containing the basis marking M_2 has binary vector [0 1], because $\mathscr{T}(M_2)$ is feasible only for T_f^2. Node $(M_2, [0\ 1])$ has two output arcs both labeled with d and both directed to node $(M_1, [0\ 0])$ with two different minimal explanations $\vec{0}$ and e_1, respectively, plus another output arc $(b, \vec{0})$ directed to node $(M_4, [1\ 1])$. ■

The following algorithm summarizes the main steps of the on-line diagnosis carried out by looking at the BRG.

Algorithm 7.18. *[Diagnosis using the BRG]*

1. *Let* $w = \varepsilon$.

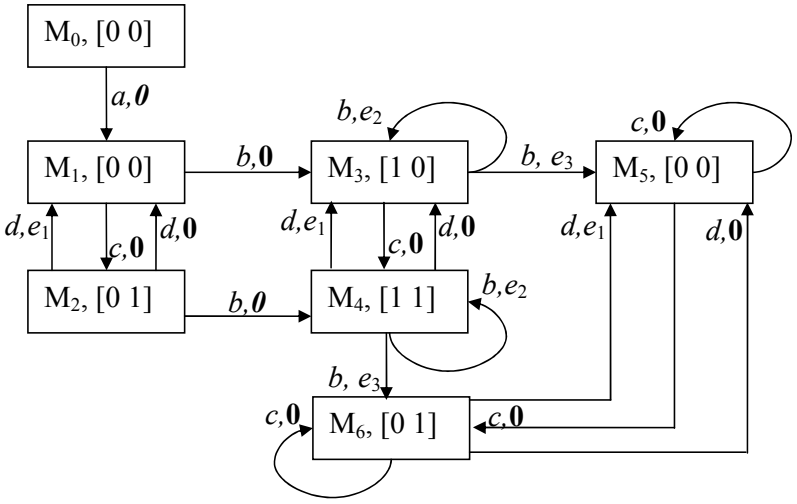


Figure 7.2: The BRG of the PN in Figure 7.1.

2. *Let* $\mathscr{M}(w) = \{(M_0, \vec{0})\}$.
3. *Wait until a new observable transition fires. Let* l *be the observed event.*
4. *Let* $w' = w$ *and* $w = w'l$.
5. *Let* $\mathscr{M}(w) = \emptyset$, *[Computation of* $\mathscr{M}(w)$*]*
6. *For all nodes containing* M' *:* $(M', y') \in \mathscr{M}(w')$, *do*
  6.1. *for all arcs exiting from the node with* M', *do*
    6.1.1. *let* M *be the marking of the output node and* e *be the minimal e-vector on the edge from* M' *to* M,
    6.1.2. *for all* y' *such that* $(M', y') \in \mathscr{M}(w')$, *do*
      6.1.2.1. *let* $y = y' + e$,
      6.1.2.2. *let* $\mathscr{M}(w) = \mathscr{M}(w) \cup \{(M, y)\}$,
7. *for all* $i = 1, \ldots, r$, *do* *[Computation of the diagnosis state]*
  7.1. *if* $\forall\ (M, y) \in \mathscr{M}(w) \wedge \forall\, t_f \in T_f^i$ *it is* $y(t_f) = 0$, *do*
    7.1.1. *if* $\forall\ (M, y) \in \mathscr{M}(w)$ *it holds* $x(i) = 0$, *where* x *is the binary vector in node* M, *do*
      7.1.1.1. *let* $\Delta(w, T_f^i) = 0$,
    7.1.2. *else*
      7.1.2.1. *let* $\Delta(w, T_f^i) = 1$,
  7.2. *if* $\exists\ (M, y) \in \mathscr{M}(w)$ *and* $(M', y') \in \mathscr{M}(w)$ *s.t.:*
    (i) $\exists t_f \in T_f^i$ *such that* $y(t_f) > 0$,
    (ii) $\forall\, t_f \in T_f^i$, $y'(t_f) = 0$, *do*
    7.2.1. *let* $\Delta(w, T_f^i) = 2$,
  7.3. *if* $\forall\ (M, y) \in \mathscr{M}(w)\ \exists t_f \in T_f^i : y(t_f) > 0$, *do*
    7.3.1. *let* $\Delta(w, T_f^i) = 3$.
8. *Goto step 3.*

■

Example 7.19. Let us consider the PN in Figure 7.1 and its BRG in Figure 7.2.

Let $w = \varepsilon$. By looking at the BRG we establish that $\Delta(\varepsilon, T_f^1) = \Delta(\varepsilon, T_f^2) = 0$ being both entries of the row vector associated to M_0 equal to 0.

Now, let us consider $w = ab$. In such a case $\mathscr{M}(w) = \{(M_3, \vec{0})\}$. It holds $\Delta(ab, T_f^1) = 1$ and $\Delta(ab, T_f^2) = 0$ being the row vector in the node equal to [1 0].

Finally, for $w = abbc$ it holds $\Delta(abbc, T_f^1) = 2$ and $\Delta(abbc, T_f^2) = 1$. In fact $\mathscr{M}(w) = \{(M_4, y_1), (M_5, y_2), (M_5, y_3)\}$, where $y_1 = e_2$ and $y_2 = y_3 = e_3$, and the row vectors associated to M_4, M_5 and M_6 are respectively [1 1], [0 0] and [0 1]. ■

Chapter 8

Diagnosability of bounded Petri nets

Summary

In this chapter we present an approach to solve the problem of diagnosability of bounded Petri net systems. In particular, we first give necessary and sufficient conditions for diagnosability. Then, we present a method to test diagnosability that is based on the analysis of two graphs that depend on the structure of the net, including the fault models, and the initial marking. The first graph is called *basis reachability diagnoser*, the second one is called *modified basis reachability graph*. At the end of the chapter a comparison between our diagnosis procedure and Diagnoser approach, presented in Chapter 4, is made.

8.1 Problem Statement

Assume that the set of transitions is partitioned as $T = T_o \cup T_u$, where T_o is the set of observable transitions, and T_u is the set of unobservable transitions.

When an observable transition fires we observe its label, thus our observations consist in sequences of symbols in the alphabet L.

The set of unobservable transitions is partitioned in two subsets, namely $T_u = T_f \cup T_{reg}$ where T_f includes all fault transitions (modeling anomalous or fault behavior), while T_{reg} includes all transitions relative to unobservable but regular events. The set T_f is further partitioned into r different subsets T_f^i, where $i = 1, \ldots, r$, that model the different fault classes.

In this chapter we deal with the problem of diagnosability of a bounded Petri net system.

Definition 8.1. *A Petri net system $\langle N, M_0 \rangle$ having no deadlock after the occurrence of transition $t_f \in T_f^i$, for $i = 1, \ldots, r$, is said* diagnosable with respect to *(wrt)* the fault class T_f^i *if there do not exist two sequences σ_1 and σ_2 in T^* satisfying the following conditions:*

- $P_o(\sigma_1) = P_o(\sigma_2)$,
- $\forall\, t_f \in T_f^i, t_f \notin \sigma_1$,
- $\exists$ *at least one $t_f \in T_f^i$ such that $t_f \in \sigma_2$,*
- *σ_2 can be made arbitrarily long after a fault $t_f \in T_f^i$.* ■

The previous definition of diagnosability of Petri nets is inspired by the definition of diagnosability for languages introduced in [29].

Definition 8.2. *A Petri net system $\langle N, M_0 \rangle$ is said* diagnosable *if it is diagnosable wrt all fault classes.* ■

Note that the diagnosability of a system does not imply that we are able to distinguish among transitions in the same class. It simply implies that if one or more transitions in a given fault class have fired, then after a finite number of observations we are able to establish that at least one transition of that class has fired.

In this chapter we investigate the problem of providing necessary and sufficient conditions for diagnosability. In particular, we consider labeled Petri net systems under the following assumptions.

A1) The net system $\langle N, M_0 \rangle$ is *bounded* and *does not deadlock after the firing of any fault transition.*

A2) The T_u-induced net is *acyclic.*

A3) The labeling function $\varphi : T_o \to L$ may associate the same label to different transitions.

A4) The structure of N is known as well as the initial marking M_0.

Note that, the necessity of assumption (A2) is discussed at the end of Chapter 6.

8.2 Modified Basis Reachability Graph

The Basis reachability Graph (BRG), introduced in Chapter 6 for unlabeled Petri nets and successively in Chapter 7 for labeled Petri nets, needs to be modified if we want to use it as an auxiliary tool to establish if the system is diagnosable. In fact, BRG is an automaton whose set of events coincides with the set of observable transitions. As it will be shown in Section 8.5, in certain cases we need to know if the fault occurs which states are reached, and this information is not included in the BRG. To this aim we define a new graph, that we call *Modified Basis Reachability Graph* (MBRG) that contains this information.

The MBRG is a deterministic graph whose nodes contain two elements $(M,\ x)$: $M \in \mathbb{N}^m$ is a marking defined as below, and x is row vector in $\{0,1\}^r$ where $x(i) = 1$ if $\mathscr{T}(M)$ in (7.1) is feasible wrt the ith class, $x(i) = 0$ otherwise.

Markings M in the nodes are defined as basis markings computed assuming that all fault transitions are observable. This means that minimal explanations are restricted to transitions in T_{reg}.

In the following we denote as $Y_{\min}^{mod}(M,t)$ the set of minimal e-vectors restricted to T_{reg}, and C_{reg} the restriction of the incidence matrix to T_{reg}.

Arcs may be labeled in two different ways depending on the associated event.

In the case of events corresponding to the firing of transitions in T_o, the label contains three informations summarized as $(l(t), e)$, where $l \in L$ is the observed label, t is the transition labeled l whose firing at the input node is enabled by a sequence of regular transitions with firing vector $e \in Y_{\min}^{mod}(M,t)$, and that leads to the marking in the output node.

In the case of events corresponding to the firing of fault transitions the label only contains two informations summarized as (t_f, e), where $t_f \in T_f$ is the fault transition whose firing at the input node is enabled by a sequence with firing vector $e \in Y_{\min}^{mod}(M,t)$, and that leads to the marking in the output node.

A formal algorithm for the construction of the MBRG can be written as follows.

Algorithm 8.3. *[Computation of the MBRG]*

1. *Label the initial node* (M_0, x_0) *where* $\forall\, i = 1,\ldots,r$,

$$x_0(T_f^i) = \begin{cases} 1 & \text{if } \mathscr{T}(M_0) \text{ is feasible,} \\ 0 & \text{otherwise.} \end{cases}$$

Assign no tag to it.

2. *While nodes with no tag exist*

1 *select a node with no tag,*

2.2 *let* (M,x) *be the selected node,*

2.3 *for all* $l \in L$
2.3.1 *for all* $t : L(t) = l \wedge Y^{mod}_{\min}(M,t) \neq \emptyset$, *do*
• *for all* $e \in Y^{mod}_{\min}(M,t)$, *do*
• *let* $M' = M + C_{reg} \cdot e + C(\cdot, t)$,
• *if* $\nexists$ *already a node with* M', *do*
• *add a new node to the graph containing the couple* (M', x') *where* $\forall\, i = 1, \ldots, r$,

$$x'(T^i_f) = \begin{cases} 1 & \text{if } \mathcal{T}(M') \text{ is feasible,} \\ 0 & \text{otherwise.} \end{cases}$$

• *add arc* $(l(t), e)$ *from node* (M, x) *to node* (M', x')
2.4 *for all* $i = 1, \ldots, r$: $x(T^i_f) = 1$
2.4.1 *for all* $t_f \in T^i_f : Y^{mod}_{\min}(M, t_f) \neq \emptyset$, *do*
• *for all* $e \in Y^{mod}_{\min}(M, t_f)$, *do*
• *let* $M' = M + C_{reg} \cdot e + C(\cdot, t_f)$,
• *if* $\nexists$ *already a node with* M', *do*
• *add a new node to the graph containing the couple* (M', x') *where* $\forall\, i = 1, \ldots, r$,

$$x'(T^i_f) = \begin{cases} 1 & \text{if } \mathcal{T}(M') \text{ is feasible,} \\ 0 & \text{otherwise.} \end{cases}$$

• *add arc* (t_f, e) *from node* (M, x) *to node* (M', x')
2.5 *tag the node* (M, x) *"old".*
3. *Remove all tags.*

■

The algorithm constructs the MBRG starting from the initial node to which it corresponds the initial marking and a binary vector defining which classes of faults may occur at M_0. Now, we consider all labels $l \in L$ (step 2.3) and all fault classes $i = 1, \ldots, r$ (step 2.4) such that there exists a transition t with $L(t) = l$ or a fault transition $t_f \in T^i_f$ for which a minimal explanation at M_0 exists. For any of such transitions, that can be either $t \in T_o$ or $t_f \in T^i_f$, we compute the marking M' resulting from its firing at $M_0 + C_u \cdot e$ ($e \in Y^{mod}_{\min}(M_0, t)$ or $e \in Y^{mod}_{\min}(M_0, t_f)$, respectively). If a new couple (marking, binary vector) is obtained, a new node is added to the graph, containing the resulting marking M' and the corresponding vector x'. The arc going from the initial node to the new node is either labeled $(l(t), e)$ or (t_f, e), depending on the considered event. The procedure is iterated until all nodes have been examined.

Note that if the net is bounded the procedure terminates in a finite number of steps because the number of nodes is upper limited by the cardinality of the set $R(N, M_0)$.

Example 8.4. Let us consider the PN in Figure 7.1, where $T_o = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$, $T_u = \{\varepsilon_8, \varepsilon_9, \varepsilon_{10}, \varepsilon_{11}, \varepsilon_{12}, \varepsilon_{13}\}$, $T^1_f = \{\varepsilon_{11}\}$ and $T^2_f = \{\varepsilon_{12}\}$. The labeling function is defined as follows: $\varphi(t_1) = a$, $\varphi(t_2) = \varphi(t_3) = b$, $\varphi(t_4) = \varphi(t_5) = c$, $\varphi(t_6) = \varphi(t_7) = d$.

The MBRG relative to the net system in Figure 7.1 is shown in Figure 8.1. The notation used in Figure 8.1 is detailed in Tables 8.1 and 8.2.

M_0	[	1	0	0	0	0	0	0	0	0	0	$]^T$
M_1	[	0	1	0	0	0	0	0	1	0	0	$]^T$
M_2	[	0	1	0	0	0	0	0	0	1	0	$]^T$
M_3	[	0	0	1	0	0	0	0	1	0	0	$]^T$
M_4	[	0	0	1	0	0	0	0	0	1	0	$]^T$
M_5	[	0	0	0	0	0	0	1	1	0	0	$]^T$
M_6	[	0	0	0	0	0	0	1	0	1	0	$]^T$
M_7	[	0	1	0	0	0	0	0	0	0	1	$]^T$
M_8	[	0	0	0	0	0	1	0	1	0	0	$]^T$
M_9	[	0	0	0	0	0	1	0	0	1	0	$]^T$
M_{10}	[	0	0	1	0	0	0	0	0	0	1	$]^T$
M_{11}	[	0	0	0	0	0	1	0	0	0	1	$]^T$
M_{12}	[	0	0	0	0	0	0	1	0	0	1	$]^T$

Table 8.1: The basis markings of the MBRG in Figure 8.1.

	ε_8	ε_9	ε_{10}	ε_{13}
e_1	1	1	1	0
e_2	1	0	0	0
e_3	0	0	0	1

Table 8.2: The modified minimal e-vectors of the MBRG in Figure 8.1 (given in tabular form).

Each node contains a different marking and a vector with two entries (because there are two fault classes). As an example, vector [0 0] is associated to M_0 because $\mathscr{T}(M_0)$ is not feasible for both fault classes. On the contrary, vector [0 1] is associated to M_2 because $\mathscr{T}(M_2)$ is not feasible for the first fault class but is feasible for the second fault class. In fact, fault transition ε_{12} is enabled at M_2.

Arcs are labeled either by (label (relative transition), corresponding modified minimal e-vector) (see e.g. $(c(t_5),\vec{0})$ from M_3 to M_4), or by (unobservable transition, corresponding modified minimal e-vector) (see e.g. $(\varepsilon_{11},\vec{e}_2)$ from M_3 to M_8).

Finally, let us observe that not all the markings in the MBRG are basis markings. Precisely, markings from M_0 to M_6 are basis markings, while markings from M_7 to M_{12} are markings reached firing a fault transition. This shows that the computational complexity required to solve the problem of diagnosability is greater than the complexity required to perform diagnosis. Note however, that the number of markings in the MBRG is equal to the number of consistent markings only in the worst case, but in general is smaller, as in this example. ■

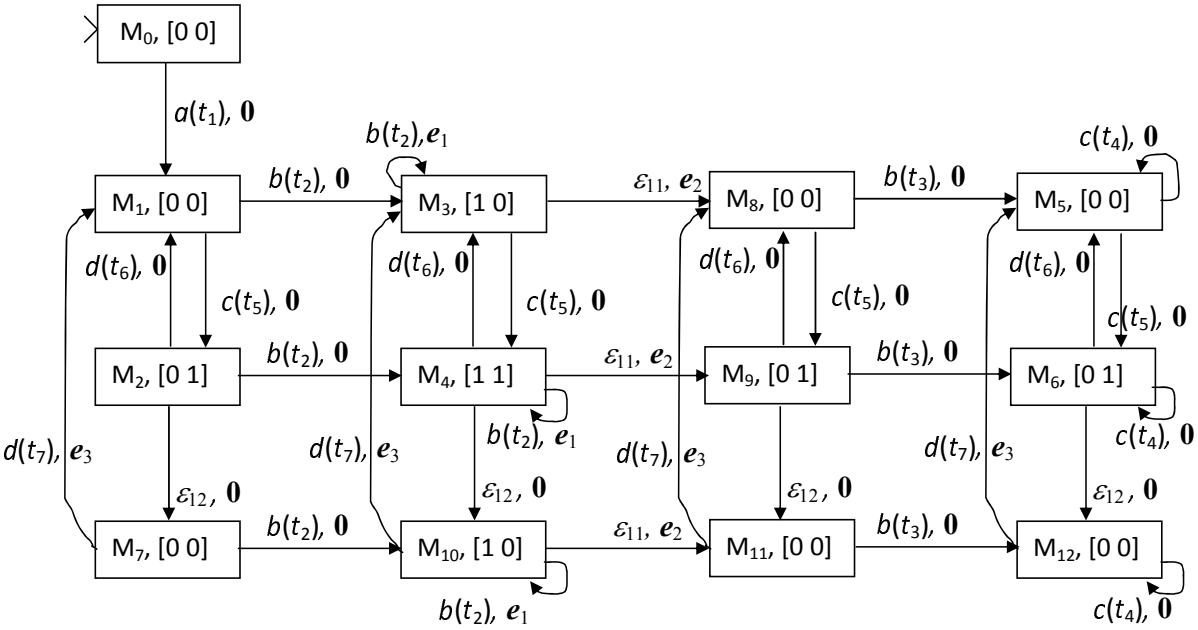


Figure 8.1: The MBRG of the Petri net in Figure 7.1.

8.3 Basis Reachability Diagnoser

In this section we define a diagnoser called *Basis Reachability Diagnoser* (BRD). It is a nondeterministic graph that, used in addition with the MBRG, allows us to state necessary and sufficient conditions for diagnosability.

Definition 8.5. *The BRD is a nondeterministic graph where each node contains the following items:*

- *one or more triples* (M, x, h)*, where:*
  - M *is a basis marking;*
  - $x \in \{0,1\}^{|T_f|}$ *is a row vector whose* i*th entry is equal to* 1 *if* $\mathcal{T}(M)$ *is feasible wrt the* i*th class, and is equal to* 0 *otherwise;*
  - $h \in \{N,F\}^{|T_f|}$ *is a row vector whose* i*th entry is equal to* N *if reaching* M *from* M_0 *no fault in* T_f^i *has occurred, and is equal to* F *otherwise;*
- r *tags* Δ_i, $i = 1, \ldots, r$*, that represent the diagnosis state of the node wrt the* r *fault classes.*

Finally, arcs are labeled with a symbol in L. ■

The BRD can be easily computed starting from the MBRG. In particular, the values of M and x are readable from the MBRG by only looking at the nodes containing basis markings.

The values of h can be deduced by looking at the path(s) from M_0 to the corresponding value of M (denoted as $M_0 \rightsquigarrow M$). If there exists a path $M_0 \rightsquigarrow M$ containing fault transitions in

the ith class, then to the couple M, x it is associated a value of $h(i) = F$. If there exists a path $M_0 \rightsquigarrow M$ containing no fault transition in the ith class, then to the couple M, x it is associated a value of $h(i) = N$. Note that, since in general there may exist more than one path going from M_0 to M, one containing a fault in T_f^i and another not, then the couple M, x may appear twice in the same node, both with $h(i) = F$ and with $h(i) = N$.

The diagnosis state for each fault class is trivially obtained by definition just looking at the last two entries of all triples in the node.

The following algorithm summarizes the main steps for the construction of the BRD. Note that to simplify the notation, we assume that each class only includes one fault transition, thus $|T_f| = r$. The extension to the more general case is trivial and is not reported here.

Algorithm 8.6. *[Computation of the BRD]*

1. *Label the initial node* $d_0 = (M_0, x_0, h_0)$, $h_0 = N^r$.
For $i = 1, \ldots, r$, *if* $x_0(i) = 0$ *then* $\Delta_i = 0$, *else* $\Delta_i = 1$.
Assign no tag to it.

2. *While nodes with no tag exist*

2.1 *select a node* d *with no tag and do*

2.2 *for all* $l \in L$

2.2.1 *for all* $M \in d$: $Y_{\min}(M, t) \neq \emptyset$ *for some transition* $t : L(t) = l$

- *for all triples with marking* M *in* d
  - *let* $\tilde{d} = \emptyset$
  - *for all output arcs of* (M, x) *in the MBRG labeled* l, *do*
    - *let* (M', x') *be the output node in the MBRG,*
    - *let*

$$\begin{cases} h'(i) = N & \text{if } h(i) = N \\ h'(i) = F & \text{if } h(i) = F \end{cases}$$

  - *let* $\tilde{d} = \tilde{d} \cup \{(M', x', h')\}$
  - *for all output paths of* (M, x) *in the MBRG labeled* $\sigma_f\, l$ *such that* $\pi(\sigma_f) \in Y_{\min}(M, t)$ *and* $L(t) = l$,
    - *let* (M', x') *be the final node in the MBRG,*
    - *let*

$$\begin{cases} h'(i) = N & \text{if } h(i) = N \wedge t_{f_i} \notin M \rightsquigarrow M' \\ h'(i) = F & \text{if } h(i) = F \\ h'(i) = F & \text{if } h(i) = N \wedge t_{f_i} \in M \rightsquigarrow M' \end{cases}$$

  - *let* $\tilde{d} = \tilde{d} \cup \{(M', x', h')\}$
  - *if* $\forall\ M' \in \tilde{d}$ *it is* $h'(i) = N$ *and* $x'(i) = 0$, *then*
    - *let* $\Delta_i = 0$
  - *else if* $\forall M' \in \tilde{d}$ *it is* $h'(i) = N$ *and* $x'(i) = 1$, *then*
    - *let* $\Delta_i = 1$
  - *else if* $\exists (M', x', h') \in \tilde{d} : h'(i) = N$ *and* $\exists (M'', x'', h'') \in \tilde{d} : h''(i) = F$, *then*
    - *let* $\Delta_i = 2$
  - *else if* $\forall\ M' \in \tilde{d}$ *it is* $h'(i) = F$, *then*
    - *let* $\Delta_i = 3$

2.2.2 *if* $\nexists$ *a node* $\bar{d} = \tilde{d}$ *in the graph then*

- *add a new node* $\tilde{d}$ *to the graph*

2.2.3 *add arc l from d to $\tilde{d}$*
2.3 *tag d* old.
2.4 *Goto step 2.1.*
3. *Remove all tags.*

■

The algorithm constructs the BRD starting from the initial node to which it corresponds a triple (M_0, x_0, h_0), where M_0 and x_0 are the components of the initial node of the MBRG and $h_0 = N^r$. Its diagnosis state Δ_i is set to zero if no fault transition in T_f^i may have occurred from the initial marking, namely if the entry of x_0 associated to the only (for assumption) fault transition $t_{f_i} \in T_f^i$ is null, otherwise Δ_i is set to one.

Starting from the initial node and looking at the MBRG we focus on the set of basis markings that are reachable firing transitions with label l at M_0, either immediately or after the firing of one or more fault transitions.

The new node will be composed by all triples (M', x', h') such that the couple (M', x') is reached in the MBRG either firing a transition labeled l at M_0, or firing a minimal explanation containing one or more fault transitions and then the considered label l; h' is computed considering h_0 and all paths from M_0 to M' ($M_0 \rightsquigarrow M'$) in the MBRG.

Finally, for each node the diagnosis state Δ_i depends on the ith entry of the two vectors x and h of all the markings appearing the node.

The procedure is iterated until all nodes have been explored.

Example 8.7. In Figure 8.2 is reported the BRD of the Petri net in Figure 7.1.

The initial node contains the triple (M_0, [0 0], [N N]) and its diagnosis states are $\Delta_1 = \Delta_2 = 0$ being $\vec{x}_0 = [0\ 0]$. From this node a is enabled and it leads to node (M_1, [0 0], [N N]), where $M_1 = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$. Also for this node diagnosis states are $\Delta_1 = \Delta_2 = 0$.

Now, let us consider $w = abb$. In this case we reach node containing the two triples (M_3, [1 0], [N N]) and (M_5, [0 0], [F N]), where $M_3 = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$ and $M_5 = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0]^T$. In fact, could have fired either observable transitions $t_1 t_2 t_2$ or $t_1 t_2 t_3$. Diagnosis states are $\Delta_1 = 2$, being $h_3(1) = N$ and $h_5(1) = F$, and $\Delta_2 = 0$, since $h_3(2) = h_5(2) = N$ and $x_3(2) = x_5(2) = 0$ (where $x_j(i)$ and $h_j(i)$ indicate the entries of vectors $\vec{x}, \vec{h}$ associated to marking M_j wrt to the fault class i).

Finally, let us consider $w = abbcc$. In this case we reach node containing the two triples (M_5, [0 0], [F N]) and (M_6, [0 1], [F N]), where $M_6 = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0]^T$. Diagnosis states are $\Delta_1 = 3$, being $h_5(1) = h_6(1) = F$, and $\Delta_2 = 1$, since $h_5(2) = h_6(2) = N$ and $x_6(2) = 1$.

Note that in the nodes of the BRD there are only basis markings. ■

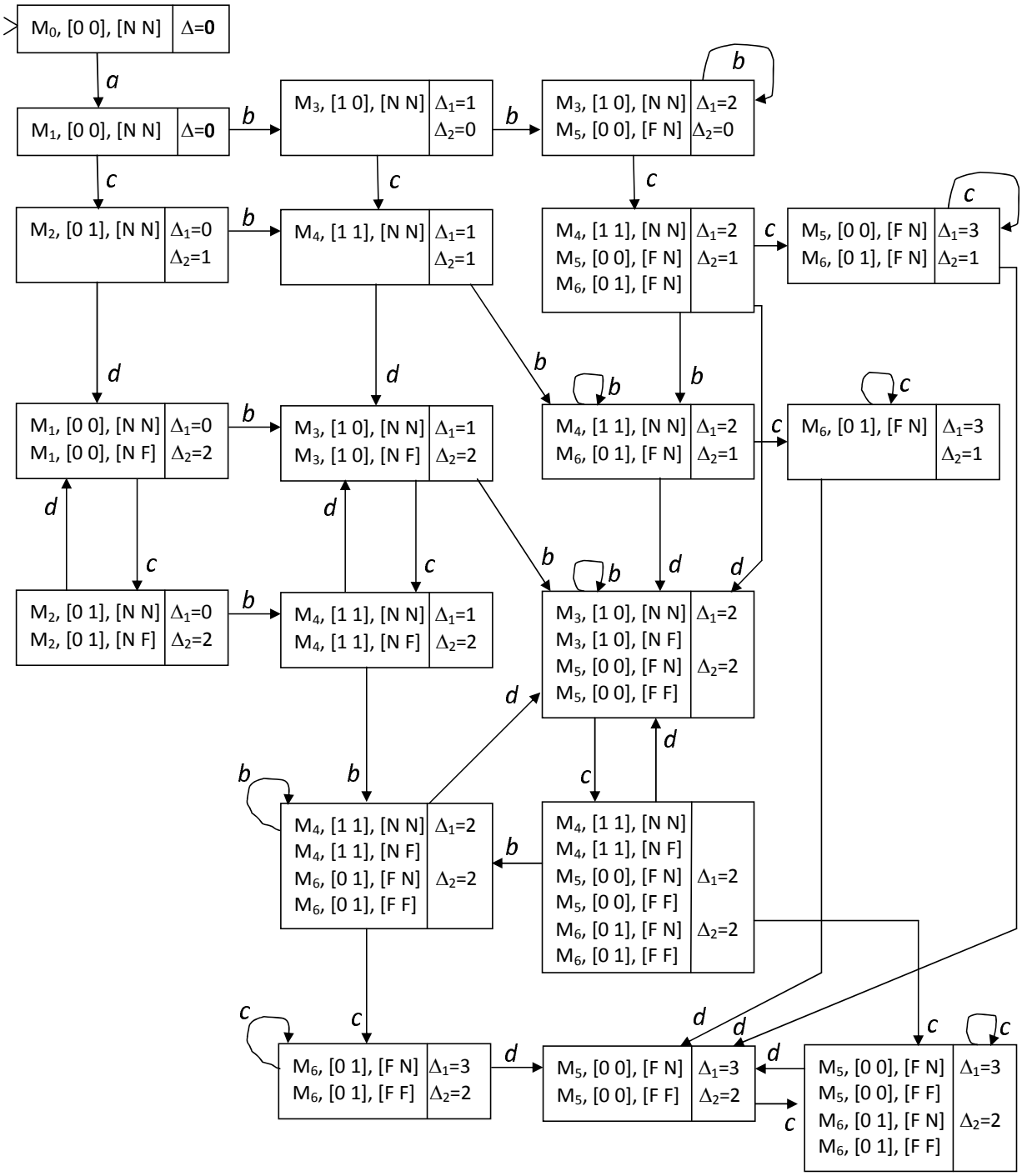


Figure 8.2: The BRD of the Petri net in Figure 7.1.

8.4 Necessary and sufficient conditions for diagnosability

In this section we provide necessary and sufficient conditions for diagnosability based on the notions of uncertain and indeterminate cycles. These conditions can be verified using the BRD in conjunction with the MBRG. In particular, first we have to check if the BRD contains an uncertain cycle, namely a potential indeterminate cycle, and then using the MBRG verify if that cycle is indeterminate or not.

Definition 8.8. *Let γ be a cycle in the BRD with observable projection $\rho \in L^*$ and let $p \in L^*$ be a path from the initial node to any node of the cycle. The cycle γ is* uncertain *wrt a fault class T_f^i if it only includes states with $\Delta_i = 2$, or $\Delta_i = 1$, or $\Delta_i = 1$ and $\Delta_i = 2$.* ■

Definition 8.9. *Let γ be an uncertain cycle in the BRD with observable projection $\rho \in L^*$ and let $p \in L^*$ be a path from the initial node to any node of the cycle. The cycle γ is* indeterminate *wrt a fault class T_f^i if in the MBRG there exist two cycles γ_1 and γ_2 satisfying the following three conditions:*

(i) their observable projection is equal to ρ;

(ii) there exist two paths p_1 and p_2 with observable projection p, that from the initial node in the MBRG enable γ_1 and γ_2;

(iii) Both γ_2 and p_2 do not contain a fault in T_f^i, while either γ_1 or p_1 or both contain a fault in T_f^i. ■

The following example well clarifies the above definition.

Example 8.10. Let us consider the BRD in Figure 8.2 that corresponds to the Petri net in Figure 7.1.

In Figures 8.3 and 8.4 are reported the uncertain cycles for the first and the second fault class, respectively.

Let us consider two uncertain cycles for the first fault class. First let us $\gamma = [(M_3, [1\ 0], [N\ N]), (M_5, [0\ 0], [F\ N])] \underrightarrow{b} [(M_3, [1\ 0], [N\ N]), (M_5, [0\ 0], [F\ N])]$, for which $\rho = b$ and $p = (M_0, [0\ 0], [N\ N]) \underrightarrow{a} (M_1, [0\ 0], [N\ N]) \underrightarrow{b} (M_3, [1\ 0], [N\ N]) \underrightarrow{b}$. Looking at the MBRG in Figure 8.1, it is easy to see that conditions of Definition 8.9 are not satisfied. In fact, does not exist a path p_1, containing fault ε_{11}, having the same observable projection of p and that enable a cycle γ_1 such that $P_o(\gamma_1) = \rho$. Therefore this cycle is not indeterminate.

Let us consider now the uncertain cycle $[(M_1, [0\ 0], [N\ N]), (M_1, [0\ 0], [N\ F])] \underrightarrow{c} [(M_2, [0\ 1], [N\ N]), (M_2, [0\ 1], [N\ F])] \underrightarrow{d} [(M_1, [0\ 0], [N\ N]), (M_1, [0\ 0], [N\ F])]$, that has $\rho = cd$ and $p = (M_0, [0\ 0], [N\ N]) \underrightarrow{a} (M_1, [0\ 0], [N\ N]) \underrightarrow{c} (M_2, [0\ 1], [N\ N]) \underrightarrow{d}$. In this case the three conditions of Definition 8.9 are satisfied, therefore the cycle ρ is indeterminate. In fact, in the MBRG there exist two cycle $\gamma_1 = (M_1, [0\ 0]) \underrightarrow{c(t_5)} (M_2, [0\ 1]) \underrightarrow{\varepsilon_{12}} (M_7, [0\ 0]) \underrightarrow{d(t_7)} (M_1, [0\ 0])$ and $\gamma_2 = (M_1, [0\ 0]) \underrightarrow{c(t_5)} (M_2, [0\ 1]) \underrightarrow{d(t_6)} (M_1, [0\ 0])$ having the same observable projection ρ and there exist two

paths $p_1 = p_2 = (M_0, [0\ 0])\underrightarrow{a(t_1)}(M_1, [0\ 0])\underrightarrow{c(t_5)}(M_2, [0\ 1])\underrightarrow{d(t_6)}$ having the same observable projection of p and that from the initial node enable γ_1 and γ_2. Finally both p_2 and γ_2 do not contain fault transition ε_{11}, while γ_1 contains ε_{11}.

Finally, let us consider an uncertain cycle for the second fault class composed by states having $\Delta_2 = 1$. The considered cycle in the BRD is $\gamma = (M_6, [0\ 1], [F\ N])\ \underrightarrow{c}\ (M_6, [0\ 1], [F\ N])$, for which $\rho = c$ and $p = (M_0, [0\ 0], [N\ N])\ \underrightarrow{a}\ (M_1, [0\ 0], [N\ N])\ \underrightarrow{b}\ (M_3, [1\ 0], [N\ N])\ \underrightarrow{c}\ (M_4, [1\ 1],$ $[N\ N])\ \underrightarrow{b}\ [(M_4, [1\ 1], [N\ N]), (M_6, [0\ 1], [F\ N])]\ \underrightarrow{c}$. Looking at the MBRG in Figure 8.1 we can see that this cycle is indeterminate since conditions of Definition 8.9 are satisfied. In fact, in the MBRG there exist two cycle $\gamma_1 = (M_6, [0\ 1])\underrightarrow{c(t_4)}\ (M_6, [0\ 1])$ and $\gamma_2 = (M_{12}, [0\ 0])\underrightarrow{c(t_4)}(M_{12}, [0\ 0])$ having the same observable projection ρ and there exist two paths $p_1 = (M_0, [0\ 0])\underrightarrow{a(t_1)}(M_1,$ $[0\ 0])\underrightarrow{b(t_2)}(M_3, [1\ 0])\ \underrightarrow{c(t_5)}(M_4, [1\ 1])\ \underrightarrow{\varepsilon_{11}}(M_9, [0\ 1])\underrightarrow{b(t_3)}(M_6, [0\ 1])\underrightarrow{c(t_4)}$ and $p_2 = (M_0, [0\ 0])\underrightarrow{a(t_1)}$ $(M_1, [0\ 0])\underrightarrow{b(t_2)}\ (M_3, [1\ 0])\underrightarrow{c(t_5)}\ (M_4, [1\ 1])\ \underrightarrow{\varepsilon_{11}}(M_9, [0\ 1])\ \underrightarrow{b(t_3)}(M_6, [0\ 1])\underrightarrow{\varepsilon_{12}}(M_{12}, [0\ 0])\underrightarrow{c(t_4)}$ having the same observable projection of p and that from the initial node enable γ_1 and γ_2. Finally both p_2 and γ_2 do not contain fault transition ε_{12}, while p_1 contains ε_{12}. ■

The following theorem allow us to test diagnosability looking for indeterminate cycles in the BRD.

Theorem 8.11. *A net system $\langle N, M_0\rangle$ satisfying assumptions (A1) to (A4) is* diagnosable *wrt the fault class T_f^i iff its BRD has no cycle that is indeterminate wrt T_f^i.*

Proof. We prove the if and only if statements separately.

(If) Assume by contradiction that an indeterminate cycle labeled ρ exists in the BRD. Moreover, we assume that in the MBRG there exist two cycles γ_1 and γ_2 satisfying conditions (i) to (iii) in Definition 8.9. This obviously implies that there exist two sequences relative to $p_1\gamma_1$ and $p_2\gamma_2$ having the same observable projection, one containing a fault in the ith class and the other one not, that can be made arbitrary long, because γ_1 and γ_2 can be repeated an arbitrary large number of times. Thus, by Definition 8.1 the system is not diagnosable wrt to the ith class.

(Only if) Assume that the BRD has no cycle that is indeterminate wrt T_f^i. By Definition 8.1 the other sequences that may potentially lead to a violation of the diagnosability property because they have the same observable projection and can be made arbitrary long, are those corresponding to cycles with one of the following features: (1) they include at least one node with $\Delta_i = 0$; (2) they only include nodes with $\Delta_i = 3$; (3) they include nodes with $\Delta_i = 1$ and/or $\Delta_i = 2$ but they are not indeterminate.

Case (1) means that after a finite number of observed events (at most equal to the number of events of the cycle in the BRD) it is possible to be sure that no fault has occurred, thus the third item of Definition 8.1 may never happen.

Case (2) means that a fault has occurred for sure, thus the second item of Definition 8.1 may never hold.

Case (3) means that there do not exist two sequences σ_1 and σ_2 having the same observable projection where σ_2 can be made arbitrary long, namely there do not exist two sequences

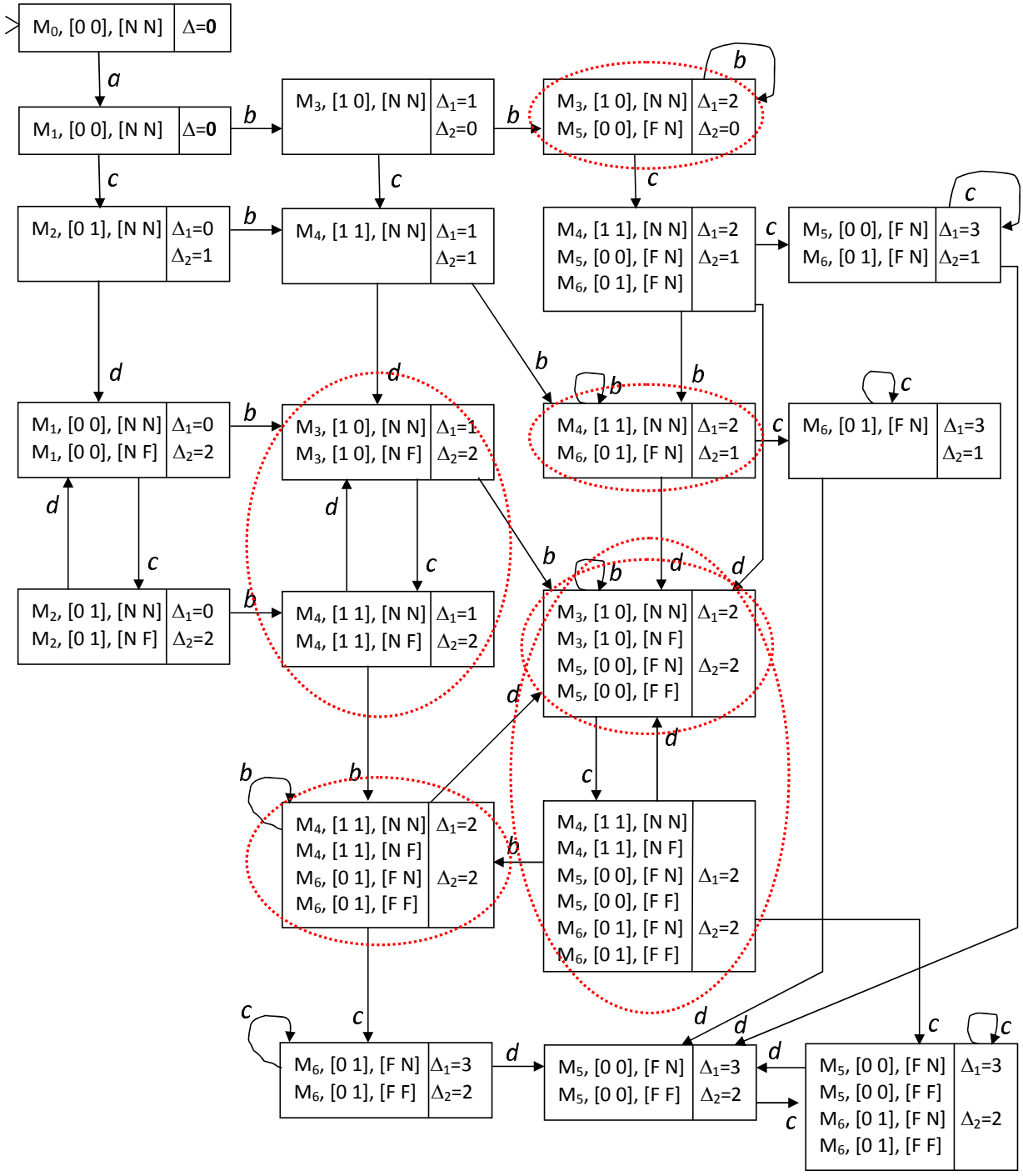


Figure 8.3: The BRD of the Petri net in Figure 7.1 with red circles for the uncertain cycles of the first fault class.

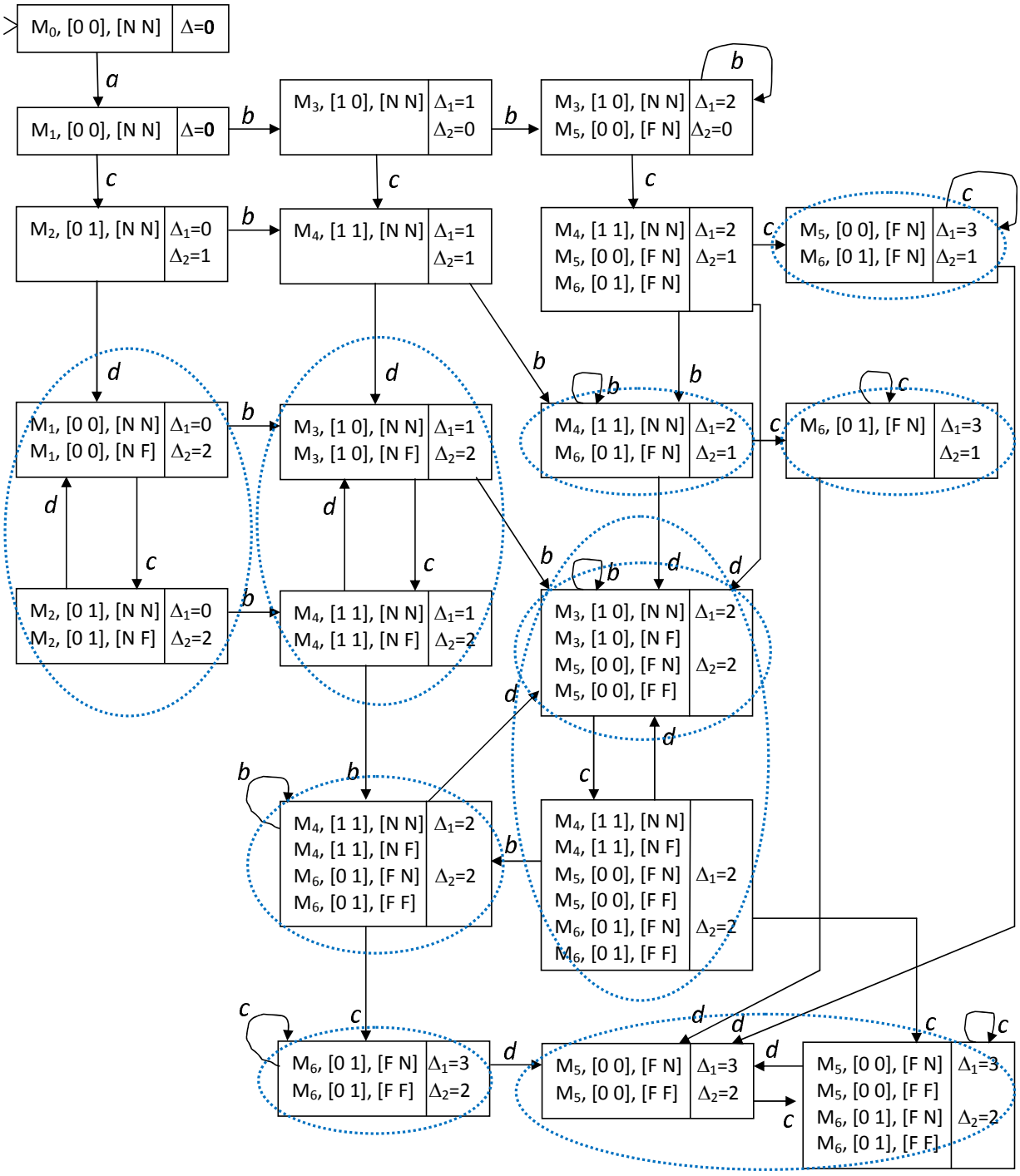


Figure 8.4: The BRD of the Petri net in Figure 7.1 with blue circles for the uncertain cycles of the second fault class.

satisfying the conditions in Definition 8.1. □

Corollary 8.12. *A net system $\langle N, M_0 \rangle$ satisfying assumptions (A1) to (A4) is* diagnosable *iff its BRD has no cycle that is indeterminate wrt any fault classes.* ■

The following example shows how test diagnosability of the system.

Example 8.13. Let us consider the Petri net system in Figure 7.1 whose BRD is given in Figure 8.2. From the analysis on the indeterminate cycles reported in Example 8.10 we can conclude that the system is not diagnosable wrt both fault classes.

Note that, as soon as you find an indeterminate cycle for a fault class you can conclude that the system is not diagnosable wrt that fault class. On the contrary, to establish if a system is diagnosable wrt to a fault class it is necessary to examine all uncertain cycles for that fault class and show that none is indeterminate. ■

8.5 Necessity of the MBRG

As already discussed, the BRG does not give us all information necessary to establish if a system is diagnosable or not. In this section we show with two examples why the construction of the MBRG is necessary.

Let us consider the Petri net in Figure 8.5.(a), where the set of observable transitions is $T_o = \{t_1, t_2, t_3, t_4\}$ and the set of unobservable transitions, that coincides with the set of fault transitions, is $T_u = T_f = \{f\}$. Thus there is just one fault class. The labeling function is $\mathscr{L}(t_1) = a$, $\mathscr{L}(t_2) = b$, $\mathscr{L}(t_3) = c$ and $\mathscr{L}(t_4) = d$. The BRD shown in Figure 8.5.(d) has a cycle of states having diagnosis state $\Delta = 1$. To understand if the system is diagnosable we need to know if after the fault f occurs, the behavior of the net system changes or not. The problem is that the BRG, shown in Figure 8.5.(b), shows that the fault may have occurred starting from both states having diagnosis state $\Delta = 1$ in the BRD, but it does not specify if the fault occurs which states are reached. In this case the net system is diagnosable, since as shown by the MBRG (shown in Figure 8.5.(c)) if the fault f occurs the behavior of the net changes, i.e., if the f occurs either the label c or the label d is observed.

Now, let us consider the Petri net in Figure 8.6.(a), where the set of observable transitions is $T_o = \{t_1, t_2\}$ and the unique unobservable and fault transition is f, namely $T_u = T_f = \{f\}$. The labeling function is $\mathscr{L}(t_1) = a$, $\mathscr{L}(t_2) = b$. The BRD shown in Figure 8.6.(d) has a cycle of states having diagnosis state $\Delta = 1$. Also in this case the analysis of the BRG, shown in Figure 8.6.(b), is not sufficient to understand what happens after the fault occurs. On the contrary, using the MBRG (shown in Figure 8.6.(c)) it can be noticed that the system shows the same behavior before the fault occurs and after the fault has occurred. Then, the system is not diagnosable.

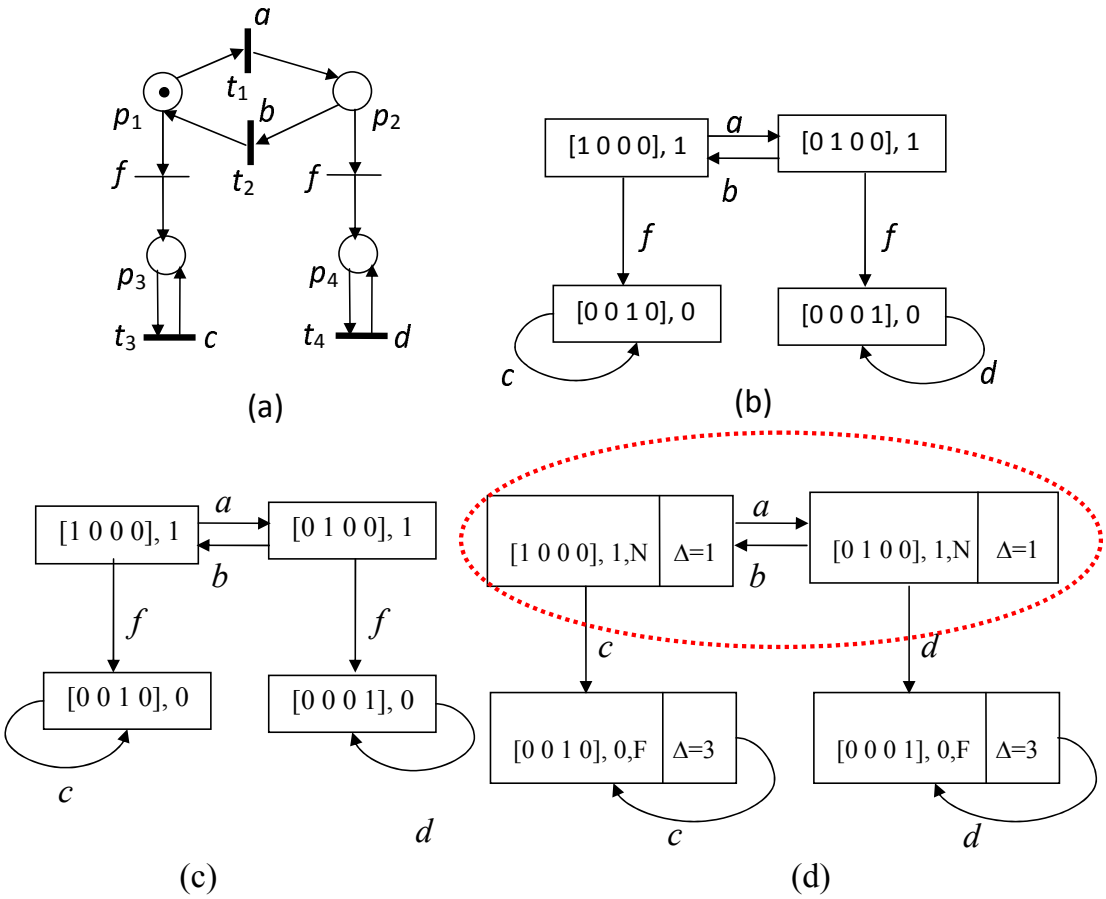


Figure 8.5: (a) An example of PN, (b) its BRG, (c) it MBRG, (d) its BRD.

8.6 Remark

In this chapter we presented an approach to solve the problem of diagnosability of bounded Petri nets based on the concept of *basis marking*, that allows us to represent the reachability space in a compact manner. We first give a necessary and sufficient condition for diagnosability. Then, we provide a method to test the diagnosability that is based on the analysis of a diagnoser that we call *basis reachability diagnoser*, in conjunction with another graph (that is used for the construction of the diagnoser) called *modified basis reachability graph*. The computational complexity to compute diagnosability is greater than the the complexity to perform diagnosis. In fact, in this case we need to take into account not only basis markings, but also all markings reachable firing a fault transition. Note however, that the number of markings in the MBRG is equal to the number of consistent markings only in the worst case, but in general is smaller.

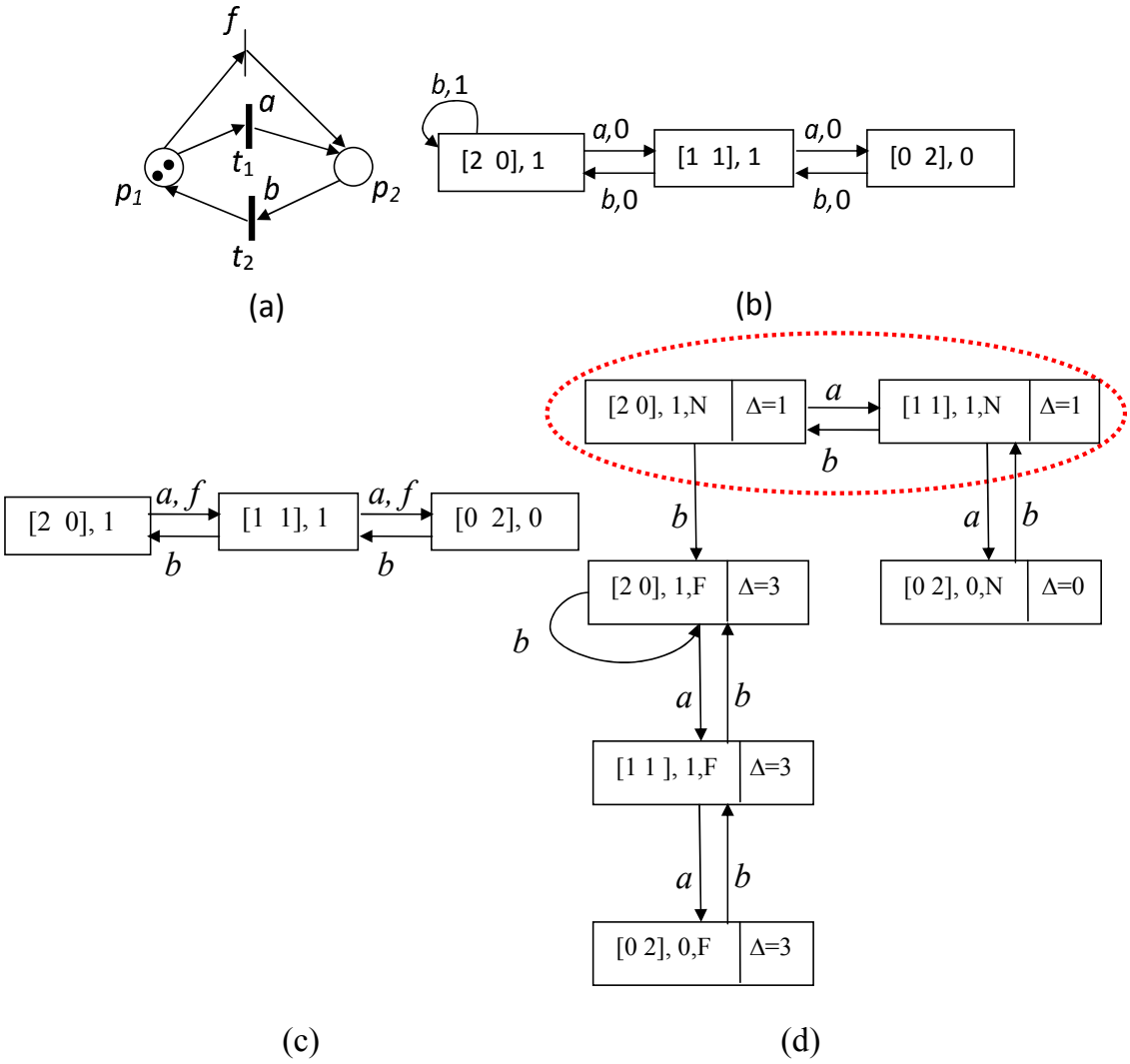


Figure 8.6: (a) An example of PN, (b) its BRG, (c) it MBRG, (d) its BRD.

8.7 A comparison between Diagnoser Approach and our diagnosis approach

The diagnosis approach presented in Chapter 4, that is based on automata, and our diagnosis approach, described in Chapters 6, 7, 8, that is based on Petri nets, have many analogies that can be summarized as follow.

- In the first case the set of fault events is a subset of the set of unobservable events. Analogously, in the Petri net case the set of fault transitions is a subset of the set of unobservable transitions.
- The automata approach can be applied to arbitrarily labeled nondeterministic finite state automaton (NFSA). Analogously, the Petri net approach can be applied to arbitrarily labeled Petri nets, as shown in Chapter 7.
- Both approaches allow to check diagnosability of the given system.

The advantages of our approach wrt Diagnoser approach are due to the fact that we use Petri nets. In fact, automata provide a general framework for establishing fundamental properties of DES. They are not convenient or intuitive models for practical systems, however, because of the large number of states that have to be introduced to represent several interacting subsystems. Moreover, the lack of structure in automata models limits the possibilities for developing computationally efficient algorithms for diagnosis. Petri nets have been proposed as an alternative modeling formalism for DES to exploit purported advantages Petri nets offer over automata models. Petri net models are generally more compact and more powerful than automata models.

In particular, the main advantage of our approach consists in the fact that we only need to enumerate basis markings, that are a subset of the set of reachable markings. This advantage become substantial when we deal with systems having a big number of unobservable transitions in series (as will be shown in Chapter 9). As an example, for the Petri net in Figure 6.1, the set of basis markings for the observed word $w = t_1 t_2$ has cardinality equal to 4 while the set of consistent markings has cardinality equal to 68, as shown in Chapter 6.

Moreover, we can also deal with unbounded systems, namely systems with an infinite state space. On the contrary, the automata approach is based on NFSA, thus it can only deal with systems with finite state space.

In spite of all that, Diagnoser approach is more suitable if we deal with systems that do not have an extended state space. In fact, in this case we do not have to introduce the heavy notation of our approach.

Another disadvantage of our approach wrt Diagnoser approach, is that it requires a stronger assumption on the structural conditions of the unobservable systems. In fact, automata approach only requires that no cycle of unobservable events may happen after the occurrence of fault events, while our approach requires that unobservable subnet is acyclic.

A comparison between the two diagnostic procedures from a computational point of view is presented in Chapter 9.

Chapter 9

A Comparison Between Two Diagnostic Tools Based on Automata and Petri Nets

Summary

In this chapter we consider two diagnosis procedures for discrete event systems based respectively on automata and Petri nets. We apply them to a diagnosis benchmark and compare them in terms of computational complexity. As a result we conclude that the Petri net approach presents significant advantages in terms of computational complexity.

Note that this chapter is based on paper [54] that has been written before we developed the procedure to perform diagnosability of Petri nets, described in Chapter 8. Then the comparison has been done between the automata tool, that allows to perform on-line diagnosis and diagnosability, and Petri nets tool, that only allows to perform on-line diagnosis. This fact has to be considered in the simulations results that reveal a significant advantage in terms of computational complexity using the Petri nets tool. Our future work will be that of developing a tool for the diagnosability of a Petri net and providing a new comparison between this tool and automata tool.

9.1 Diagnosis

In this section we briefly recall the diagnosis approach using automata introduced in Chapter 4. Then, we introduce the diagnosis approach using Petri nets presented in Chapter 6. Finally, we compare the two diagnosis procedures.

Note that this chapter is based on paper [54] that has been written before we developed the procedure to perform diagnosability of Petri nets, described in Chapter 8. Then the comparison has been done between the automata tool, that allows to perform on-line diagnosis and diagnosability, and Petri nets tool, that only allows to perform on-line diagnosis. This fact has to be considered in the simulations results that reveal a significant advantage in terms of computational complexity using the Petri nets tool. Our future work will be that of developing a tool for the diagnosability of a Petri net and providing a new comparison between this tool and automata tool.

9.1.1 Diagnosis using automata

The system to be diagnosed is modeled by a NFSA. Faults are modeled by unobservable events, but there may also exist other events that represent legal behaviors that are unobservable as well. Thus we assume that the set of events can be partitioned as $E = E_o \cup E_u$, where E_o is the set of observable events, and E_u is the set of unobservable events. The set of fault events is denoted E_f and it holds $E_f \subseteq E_u$. The set E_f can be further partitioned into r different subsets E_f^i, where $i = 1, \ldots, r$, that model the different fault classes.

Given a NFSA $G = (X, E, \Delta, x_0)$ it is always possible to define a DFSA, called *observer* of G, $Obs(G)$. The projection on E_o of the language generated by G coincides with the language generated by the observer $Obs(G)$, namely with $\mathscr{L}(G)$. Each state of $Obs(G)$ is composed by a subset of X. In particular, the initial state of the observer $x_{0_{obs}}$ includes x_0 and all the states of G that are reachable starting from x_0 executing one or more unobservable events in G. Analogously, a generic state reachable from $x_{0_{obs}}$ with an observable sequence of events $w \in E_o^*$ is composed by the set of states that are reachable in G from the states in $x_{0_{obs}}$ executing w interleaved with zero or more unobservable events.

The *diagnoser* $Diag(G)$ is a DFSA as well and it can be built with the same procedure used for the observer. The states of $Diag(G)$ are composed by ordered sets of $r+1$ entries, $\{x, l_1, l_2, \ldots, l_r\}$, where $x \in X$ is a state of G and $l_i \in \{F_i, N_i\}$ is the label associated to the ith fault class. In particular, $l_i = F_i$ if x has been reached firing at least one fault event of class i, otherwise $l_i = N_i$.

Note that the state of a diagnoser may contain two different sets relative to the same state $x \in X$ and the same fault class, namely $\{x, l_1, \ldots, l_i, \ldots, l_r\}$ and $\{x, l_1, \ldots, l_i', \ldots, l_r\}$ where $l_i = F_i$ and $l_i' = N_i$. This happens when the same state x can be reached in G executing words that contain fault events in the ith class and words not containing them.

The states of $Diag(G)$ can be classified as follows.

- *Positive state to the ith class*: all entries relative to the ith class are equal to F_i. This means that if the word that leads to such a state is observed, then we can be sure that

a fault event in the ith class has occurred.

- *Negative state to the ith class*: all entries relative to the ith class are equal to N_i. This means that if the word that leads to such a state is observed, then we can be sure that no fault event in the ith class has occurred.
- *Uncertain state to the ith class*: it includes both sets labeled with N_i and sets labeled with F_i. In such a case a fault event may either have occurred or not.

The diagnoser can be used to solve two different types of problems.

- *Diagnosis*: given a sequence of observable events w it allows to determine, for each class, if a fault in that class has occurred for sure or not, or it may have occurred, but we cannot be sure of this.

  This is equivalent to establish if the state of the diagnoser that is reached executing w is respectively, positive, negative or uncertain to each class.
- *Diagnosability*: it allows to determine if a system is *diagnosable*, i.e., if it is possible to reconstruct the occurrence of fault events observing words of finite length (see Definition 4.6).

9.1.2 Diagnosis using Petri nets

Faults are modeled by unobservable transitions, but there may also exist other transitions that represent legal behaviors that are unobservable as well. Thus we assume that the set of transitions can be partitioned as $T = T_o \cup T_u$, where T_o is the set of observable transitions, and T_u is the set of unobservable transitions. The set of fault transitions is denoted T_f and it holds $T_f \subseteq T_u$. The set T_f can be partitioned into r different subsets T_f^i, where $i = 1, \ldots, r$, that model the different fault classes.

The diagnosis with Petri nets is based on two main notions: *j-vector (or justification-vector)* and *basis marking*.

Given an observation $w \in T_o^*$, we denote as $\mathscr{J}(w)$ the *set of justifications*, i.e., the set of *minimal* sequences of unobservable transitions interleaved with w and whose firing enables w. We denote as *j-vectors* the firing vectors relative to the justifications in $\mathscr{J}(w)$.

Finally, the *set of basis markings at w* $\mathscr{M}_{basis}(w)$ is the set of markings reached from M_0 firing w interleaved with a justification $\sigma_u \in \mathscr{J}(w)$. The generic marking in $\mathscr{M}_{basis}(w)$ is denoted M_b.

The diagnoser is a function $\Delta : T_o^* \times \{T_f^1, \ldots, T_f^r\} \longrightarrow \{0,1,2,3\}$ that associates to each observation w and to each fault class T_{f_i}, $i = 1, \ldots, r$, a *diagnosis state*.

- 0: the ith fault cannot have occurred because none of the firing sequences consistent with the observation contains fault transitions of class i.

- 1: a fault transition of class i may have fired but is not contained in any justification of w.
- 2: a fault transition of class i is contained in one (but not in all) justifications of w.
- 3: the ith fault must have occurred, because all firable sequences consistent with the observation contain at least one fault transition of class i.

In Chapter 6 we shown that the procedure to evaluate the diagnosis states may be carried out by simply performing matrix multiplications and evaluating the feasibility of certain integer constraint sets. Such a procedure may be applied to all net systems whose unobservable subnet is acyclic.

Clearly, the most burdensome part of the proposed procedure consists in evaluating the feasibility of a finite number of integer constraint sets. However, as shown in Chapter 6, in the case of bounded net systems this computation may be moved off-line. An oriented graph, that we call *basis reachability graph* (BRG) may be constructed off-line, that summarizes all the information required for diagnosis. Then, given any observable word w, for any fault class T_f^i, we may evaluate the corresponding diagnosis state by simply looking at the BRG.

9.2 The considered benchmark

The benchmark describes a family of manufacturing systems characterized by three parameters: n, m and k.

- n is the number of production lines.
- m is the number of units of the final product that can be simultaneously produced. Each unit of product is composed of n parts.
- k is the number of operations that each part must undergo in each line.

To obtain one unit of final product n orders are sent, one to each line; this is represented by observable event t_s. Each line will produce a part (all parts are identical) and put it in its final buffer. An assembly station will take one part from each buffer (observable event t_e) to produce the final product.

The part in line i ($i = 1,\ldots,n$) undergoes a series of k operations, represented by unobservable events $\varepsilon_{i,1},\varepsilon_{i,2},\cdots,\varepsilon_{i,k}$.

After this series of operations two events are possible: either the part is regularly put in the final buffer of the line, or a fault may occur.

- Putting the part in the final buffer of line 1 corresponds to unobservable event $\varepsilon_{1,k+1}$, while putting the part in the final buffer of line i ($i = 2,\ldots,n$) corresponds to observable event $t_{i,k+1}$.

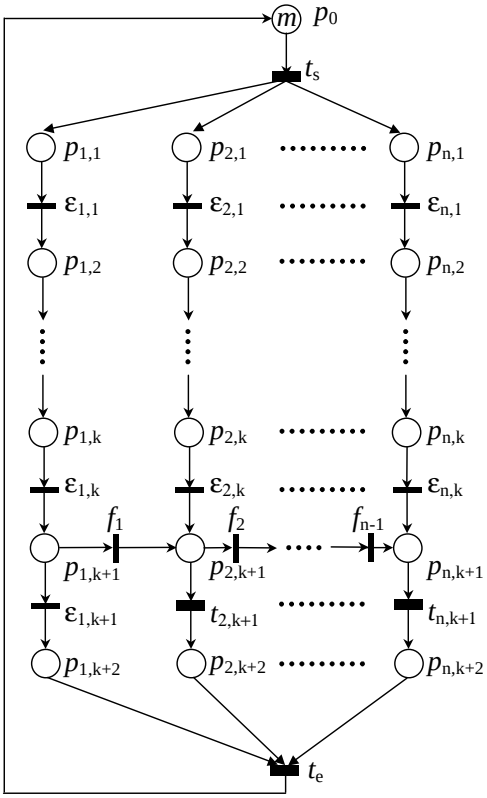


Figure 9.1: The considered benchmark.

- There are $n-1$ faults, represented by unobservable events f_i ($i = 1,\ldots,n-1$). Fault f_i moves a part from line i to line $i+1$. Note that on line i ($i = 1,\ldots,n-1$) the fault may only occur when the part has finished processing and is ready to be put in its final buffer; the part goes to the same processing stage in line $i+1$.

A Petri net model of this system is shown in Figure 9.1, where thick transitions represent observable event and thin transitions represent unobservable events.

9.3 Numerical simulations

In this section we compare the tool we developed in Matlab [72], that implements the Petri net procedure, and the UMDES library [53] that implements the automata procedure. In particular, such a comparison is carried out on the benchmark introduced in Section 9.2, whose Petri net model is sketched in Figure 9.1, while the automaton model corresponds to the reachability graph of the Petri net system that is not reported here for sake of brevity.

Several numerical simulations have been run for different values of n, k and m, that are summarized in Tables 9.1, 9.2 and 9.3.

Note that for sake of simplicity we assumed that all faults belong to the same class.

All simulations have been run on a PC Intel with a clock of 1.80 GHz.

n	k	$\|R\|$	t_R [sec]	$\|BRG\|$	t_{BRG} [sec]	$\|Obs\|$	$t_{Obs}[sec]$	$\|Diag\|$	t_{Diag} [sec]
2	1	15	0.011	5	< 0.03	4	0.053	12	0.048
2	2	24	0.019	5	< 0.03	15	0.060	39	0.049
2	3	35	0.026	5	0.032	22	0.059	58	0.050
2	4	48	0.038	5	0.032	34	0.060	88	0.053
3	1	80	0.074	17	0.12	51	0.061	151	0.056
3	2	159	0.234	17	0.145	228	0.063	698	0.084
3	3	274	0.693	17	0.15	494	0.073	1480	0.155
3	4	431	1.85	17	0.15	794	0.12	2405	0.31
4	1	495	2.52	69	0.51	4172	0.619	14009	16.66
4	2	1200	19.04	69	0.57	32132	319.6	126041	4874
4	3	2415	95.33	69	0.67	120083	6169.4	*o.t.*	*o.t.*
4	4	4320	368.8	69	0.766	*o.t.*	*o.t.*	*n.c.*	*n.c.*
5	1	3295	203.6	305	4.82	*o.m.*	*o.m.*	*n.c.*	*n.c.*
5	2	*o.t.*	*o.t.*	305	5.06	*n.c.*	*n.c.*	*n.c.*	*n.c.*
5	3	*o.t.*	*o.t.*	305	6.37	*n.c.*	*n.c.*	*n.c.*	*n.c.*
5	4	*o.t.*	*o.t.*	305	6.6	*n.c.*	*n.c.*	*n.c.*	*n.c.*

Table 9.1: Numerical results in the case of $m = 1$.

n	k	$\|R\|$	t_R [sec]	$\|BRG\|$	t_{BRG} [sec]	$\|Obs\|$	$t_{Obs}[sec]$	$\|Diag\|$	t_{Diag} [sec]
2	1	96	0.094	17	0.078	349	0.125	3864	1.764
2	2	237	0.521	17	0.078	9849	9.754	*o.m.*	*o.m.*
3	1	1484	27.19	140	1.17	*o.t.*	*o.t.*	*n.c.*	*n.c.*
3	2	5949	648.8	140	1.031	*o.t.*	*o.t.*	*n.c.*	*n.c.*
4	1	28203	20622	1433	86	*o.t.*	*o.t.*	*n.c.*	*n.c.*
4	2	*o.t.*	*o.t.*	1433	86.4	*n.c.*	*n.c.*	*n.c.*	*n.c.*

Table 9.2: Numerical results in the case of $m = 2$.

n	k	$\|R\|$	t_R [sec]	$\|BRG\|$	t_{BRG} [sec]	$\|Obs\|$	$t_{Obs}[sec]$	$\|Diag\|$	t_{Diag} [sec]
2	1	377	1.221	39	0.25	*o.t.*	*o.t.*	*n.c.*	*n.c.*
2	2	1293	19.038	39	0.25	*o.t.*	*o.t.*	*n.c.*	*n.c.*
3	1	12048	2978	553	7.65	*o.t.*	*o.t.*	*n.c.*	*n.c.*
3	2	*o.t.*	*o.t.*	553	7.55	*n.c.*	*n.c.*	*n.c.*	*n.c.*
4	1	*o.t.*	*o.t.*	9835	2392.3	*n.c.*	*n.c.*	*n.c.*	*n.c.*
4	2	*o.t.*	*o.t.*	9835	2395.3	*n.c.*	*n.c.*	*n.c.*	*n.c.*

Table 9.3: Numerical results in the case of $m = 3$.

- Columns 1 and 2 show the values of n and k.
- Column 3 shows the number of nodes $|R|$ of the reachability graph, and thus the num-

ber of states of the NFSA modeling the system.

- Column 4 shows the time t_R in seconds we spent to compute the reachability graph using a function we developed in Matlab.
- Column 5 shows the number of nodes $|BRG|$ of the BRG.
- Column 6 shows the time t_{BRG} in seconds we spent to compute the BRG using a function we developed in Matlab [72].
- Column 7 shows the number of nodes $|Obs|$ of the observer.
- Column 8 shows the time t_{Obs} in seconds we spent to compute the observer using the UMDES library [53].
- Column 9 shows the number of nodes $|Diag|$ of the diagnoser.
- Column 10 shows the time t_{Diag} in seconds we spent to compute the diagnoser using the UMDES library [53].

Note that, since in some cases the times to run simulations are very short, in order to minimize the variance of such times — related to the concurrency of the processes executed by the processor — the average time over several simulations has been computed. In particular, the first 9 rows of Table 9.1 and the first row of Table 9.2 show the average time over 100 simulations. The 10th row of Table 9.1 shows the average time over 20 simulations. In all the other cases only one simulation has been run.

Some boxes of the above tables contain non numerical values.

- *o.t.* (*out of time*): denotes that the corresponding value has not been computed within 6 hours;
- *n.c.* (*not computable*): denotes that the corresponding value cannot be computed: e.g., if the observer is *o.t.* the corresponding diagnoser cannot be evaluated;
- *o.m.* (*out of memory*): denotes that the corresponding value has not been computed because the virtual memory of the calculator has run out.

Tables 9.1, 9.2 and 9.3 show that the time spent to compute the reachability graph, the observer and the diagnoser highly increases with the dimension of the net, namely with n and k, and with the number of products m. In same cases it has been even impossible to compute them.

On the contrary, the time spent to compute the BRG is always reasonable even for high values of n, k and m.

We can observe that there is not a clear relationship between the number of nodes of the reachability graph and the number of nodes of the observer. As an example, let us consider the two cases: $n = 3$, $k = 3$, $m = 1$ and $n = 2$, $k = 2$, $m = 2$. In the first case the number

of nodes of the reachability graph is $|R| = 274$ and the number of nodes of the observer is $|Obs| = 494$. In the second case $|R| = 237 < 274$ and $|Obs| = 9849 >> 494$.

We can also observe that, as expected, the number of states of the diagnoser $|Diag|$ is greater than the number of states of the observer $|Obs|$. Thus, it is always possible to construct the observer if it is possible to construct the diagnoser, but not viz (e.g., $n = 4$, $k = 3$, $m = 1$). However, there is not a clear relationship between the complexity in evaluating them. As an example let us consider the two cases: $n = 4$, $k = 2$, $m = 1$ and $n = 2$, $k = 2$, $m = 2$. In the first case $|Obs| = 32132$ and it has been possible to also compute the diagnoser. In the second case $|Obs| = 9849 << 32132$ and it has not been possible to compute the diagnoser within 6 hours.

Tables 9.1, 9.2 and 9.3 also show that the number of nodes of the BRG only depends on n and m, while it is invariant with respect to k. Only the contrary, $|R|$, $|Obs|$ and $|Diag|$ also highly increases with k.

The relationship among the time to compute the reachability graph, the BRG, the observer and the diagnoser, respectively, and the values of n and k is better highlighted in Figure 9.2 in the case of $m = 1$.

Looking at Figure 9.2 it can be noticed that while t_{BRG} slowly increases with n and k, t_R, t_{Obs} and t_{Diag} highly depend on these parameters. Moreover, while the BRG is computable for all considered values of n and k, the reachability graph is only computable for $n \leq 4$ if $k \geq 2$; the observer is only computable for $n \leq 3$ if $k = 4$, and for $n \leq 4$ if $k = 1,2,3$; finally, the diagnoser is only computable for $n \leq 3$ if $k = 3,4$, and for $n \leq 4$ if $k = 1,2$.

On the basis of the above simulations we can conclude that from a computational point of view, the Petri net tool is better than the automata tool. In particular, in several cases we realized that the diagnoser cannot be built at all (at least in a reasonable time), while the BRG can always be computed in a sufficiently small time.

This is not surprising: in fact thanks to the basis markings the reachability space can be described in a compact manner. On the contrary the automata approach is based on an exhaustive enumeration of all reachable states. This advantage is particularly evident in the case of concurrent systems [15].

On the other hand, as already highlighted above, the Petri net approach considered in this chapter is only applicable when performing on-line diagnosis, while the automata approach also provides necessary and sufficient conditions for diagnosability.

We can finally observe that the small number of simulations we have been able to carry on using automata does not allow us to evaluate how the computational complexity is related to n and k in the automata approach. However, we can conjecture an exponential dependence on n since the computational time varies from few seconds to "*out of time*" slightly increasing n.

Evaluating the dependence on k is even more difficult: for $m = 1$ and $n = 2$ the times to compute the diagnoser are very small and comparable for all values of k; for $m = 1$ and $n = 3$ the time is still reasonable for all values of k, even if it increases approximately in a linear way with k; for $m = 1$ and $n = 4$ it becomes "*out of time*" for $k = 3$.

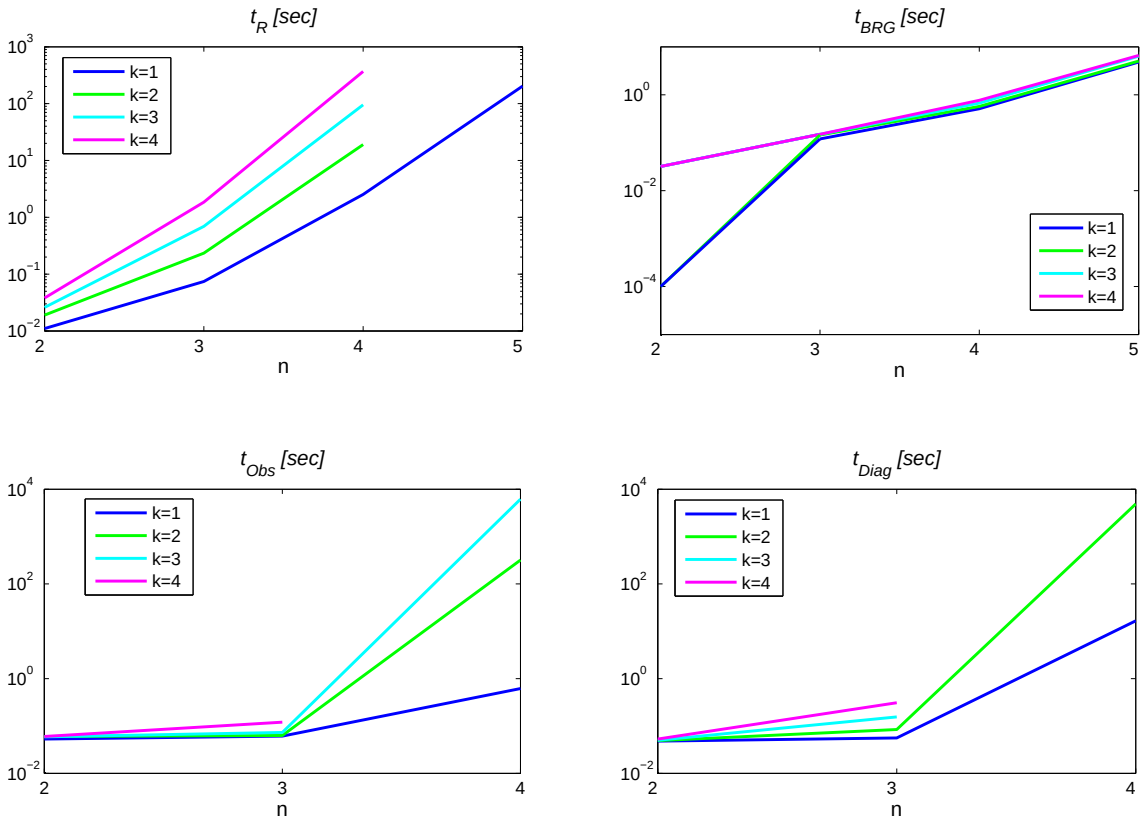


Figure 9.2: The computational times t_R, t_{BRG}, t_{Obs} and t_{Diag} with respect to n and k, in the case of $m = 1$.

Discussion

We have compared the complexity of solving the WODES benchmark problem with two different tools, one using the BRG and the other one using the diagnoser. The diagnoser has size which is exponential in the number of system states, whereas the size of the BRG is linear in the size of the state space, being at most equal to the size of the reachability graph. Thus, it was expected that the second tool should have better performance.

Note however, that it does not necessarily follow that Petri nets diagnosis approaches are better than automata based approaches. We remind that there also exist other automata based approaches that do not require an exhaustive enumeration of the diagnoser states. As an example, in [94], a subgenerator of the Reachability Transistion System (RTS) reachable from the initial state, is used in a similar fashion as the BRG. Since the system states that are only on unobservable trajectories do not show up in the reachable subgenerator of RTS, the size of the reachable subgenerator of RTS is also smaller than the original system. It would be interesting to test the WODES benchmark problem on a tool based on this approach and compare with the results obtained using our tool.

Part III

Identification

Chapter 10

Identification of Petri nets from knowledge of their language

Summary

In this chapter we deal with the problem of identifying a Petri net system, given a finite language generated by it. First we consider the problem of identifying a free labeled Petri net system, i.e., all transition labels are distinct. The set of transitions and the number of places is assumed to be known, while the net structure and the initial marking are computed solving an integer programming problem. Then we extend this approach in several ways introducing additional information about the model (structural constraints, conservative components, stationary sequences) or about its initial marking. We also treat the problem of synthesizing a bounded net system starting from an automaton that generates its language. Moreover, we show how the approach can also be generalized to the case of labeled Petri nets, where two or more transitions may share the same label. In particular, in this case we impose that the resulting net system is deterministic. In both cases the identification problem can still be solved via an integer programming problem. Finally, we analyze the complexity of such an approach in terms of computational time required to get an admissible solution, that may also be optimal according to a given performance criterion

10.1 Basic identification procedure for free labeled Petri nets

In this section we describe the identification procedure for free labeled Petri nets. As mentioned in Subsection 3.5 for this type of Petri nets we assume $E = T$ without any loss of generality.

The problem we consider in this section can be formally stated as follows.

Problem 10.1. *Let $\mathscr{L} \subset T^*$ be a finite prefix-closed language (see Appendix A), and*

$$k = \max_{\sigma \in \mathscr{L}} |\sigma|$$

be the length of the longest string in $\mathscr{L}$. Choosing a set of places P of cardinality m we want to identify the structure of a net $N = (P, T, Pre, Post)$ and an initial marking M_0 such that

$$L_k(N, M_0) = \mathscr{L}.$$

We also assume that a nonnegative integer K is given such that the following condition[1] *holds:*

$$\max_i M_0(p_i) + k \cdot \max_{i,j} Post(i, j) \leq K.$$

The unknowns we want to determine are the elements of the two matrices $Pre,\ Post \in \mathbb{N}^{m \times n}$ and the elements of the vector $M_0 \in \mathbb{N}^m$. ■

Let us consider the following definitions that we will be useful in the identification procedure.

Definition 10.2. *Given two pairs (σ, t) and (σ', t') we say that*

$$(\sigma, t) \equiv (\sigma', t')$$

if $\pi(\sigma) = \pi(\sigma')$ and $t = t'$. ■

It is easy to verify that the relation introduced in Definition 10.2 is an equivalence relation (see Appendix B). Thus two pairs (σ, t) and (σ', t') having σ and σ' with the same firing vector and where t and t' are equal, belong to the same equivalence class.

Definition 10.3. *Let $\mathscr{L} \in T^*$ be a finite prefix-closed language and let $k \in \mathbb{N}$ be defined as in Problem 10.1.*

We define the following sets

$$\mathscr{E}' = \{(\sigma, t_j) \mid \sigma \in \mathscr{L}, |\sigma| < k, \sigma t_j \in \mathscr{L}\},$$

[1] This assumption is purely technical, as mentioned in Remark 10.5, and since K can be chosen arbitrarily large does not pose any practical limitation.

$$\mathscr{E} = \mathscr{E}'|_{\equiv}, \tag{10.1}$$

$$\mathscr{D}' = \{(\sigma, t_j) \mid \sigma \in \mathscr{L}, |\sigma| < k, \sigma t_j \notin \mathscr{L}\},$$

$$\mathscr{D} = \mathscr{D}'|_{\equiv}, \tag{10.2}$$

where $\mathscr{E}$ and $\mathscr{D}$ are the sets containing only one element of each equivalent class for the $\equiv$ relation in Definition 10.2. ■

A solution to the above identification problem can be computed thanks to the following theorem, that provides a linear algebraic characterization of the place/transition nets with m places and n transitions such that $L_k(N, M_0) = \mathscr{L}$.

Theorem 10.4. *A net system $\langle N, M_0\rangle$ is a solution of the identification problem (10.1) if and only if it satisfies the following set of linear algebraic constraints*

$$\mathscr{G}_m(\mathscr{E}, \mathscr{D}) \triangleq \begin{cases} M_0 + Post \cdot y - Pre \cdot (y + \vec{t}_j) \geq \vec{0} & \forall (\sigma, t_j) \in \mathscr{E} \quad (a) \\ -K S_{\sigma,j} + M_0 + Post \cdot y - Pre \cdot (y + \vec{t}_j) \leq -\vec{1}_m & \forall (\sigma, t_j) \in \mathscr{D} \quad (b) \\ \vec{1}^{\,T} S_{\sigma,j} \leq m - 1 & \forall (\sigma, t_j) \in \mathscr{D} \quad (c) \\ M_0 \in \mathbb{N}^m & (d) \\ Pre, Post \in \mathbb{N}^{m \times n} & (e) \\ S_{\sigma,j} \in \{0,1\}^m & (f) \end{cases} \tag{10.3}$$

where $y = \pi(\sigma)$ and $\mathscr{E}, \mathscr{D}$ are given in Eq. (10.1)*, Eq.* (10.2)*, respectively.*

Proof.

- Assume that $\sigma t_j \in \mathscr{L}$, where $\sigma \in T^*$ and $t_j \in T$. Then transition t_j is enabled from the marking $M_\sigma = M_0 + (Post - Pre) \cdot y$ and the following relation must hold

  $$M_\sigma \geq Pre(\cdot, t_j).$$

  This relation can be rewritten as

  $$M_0 + Post \cdot y - Pre \cdot (y + \vec{t}_j) \geq \vec{0}_m. \tag{10.4}$$

- Assume that $\sigma \in \mathscr{L}$ and $\sigma t_j \notin \mathscr{L}$, where $\sigma \in T^*$ and $t_j \in T$. Then transition t_j is not enabled from the marking

  $$M_\sigma = M_0 + (Post - Pre) \cdot y,$$

  that is for at least one place p_i it must hold

  $$M_\sigma(p_i) < Pre(p_i, t_j).$$

We first observe that that each component of M_σ is less than or equal to K, as defined in Problem 10.1. In fact it holds:

$$\begin{aligned} K &\geq \max_i M_0(p_i) + k \cdot \max_{i,j} Post(i,j) \\ &\geq \max_i M_0(p_i) + |\sigma| \cdot \max_{i,j} Post(i,j) \\ &\geq \max_i M_\sigma(p_i). \end{aligned} \tag{10.5}$$

We now define a vector

$$S_{\sigma,j} = \begin{bmatrix} s_1 \\ \vdots \\ s_m \end{bmatrix} \in \{0,1\}^m,$$

such that for all $i = 1, \ldots, m$ it holds

$$s_i = 0 \quad \Longrightarrow \quad M_\sigma(p_i) < Pre(p_i, t_j).$$

The i-th component of $S_{\sigma,j}$ (for $i = 1, \ldots, m$) must satisfy the equation

$$-Ks_i + M_\sigma(p_i) - Pre(p_i, t_j) < 0, \tag{10.6}$$

so that if $s_i = 0$ it must hold $M_\sigma(p_i) - Pre(p_i, t_j) < 0$, while if $s_i = 1$ equation (10.6) is trivially verified thanks to equation (10.5). In vector form (and taking into account that all variables are integers) equation (10.6) rewrites:

$$-KS_{\sigma,j} + M_0 + Post \cdot y - Pre \cdot (y + \vec{t}_j) \leq -\vec{1}_m. \tag{10.7}$$

Finally, there exists at least a place that disables t_j if

$$\sum_{i=1}^{m} s_i \leq m - 1, \tag{10.8}$$

so that at least one s_i is null. In vector form this equation rewrites

$$\vec{1}^{\,T} S_{\sigma,j} \leq m - 1. \tag{10.9}$$

□

Remark 10.5. *Let us briefly comment about the constant K defined in Problem 10.1.*

In Theorem 10.4, for any pair $(\sigma, t_j) \in \mathscr{D}$ one has to look for at least one place that disables t_j after σ. This or *condition, as discussed in Appendix C, may be rewritten in a simpler form if an upper bound on the absolute value of the variables involved in the constraints is given.*

In practice, in Problem 10.1 it is sufficient to pick K very large. We also mention that many software tools allow the definition of an arbitrary large constant.

In general the solution of the set (10.3) is not unique, thus there exists more than one Petri net system $\langle N, M_0 \rangle$ such that $L_k(N, M_0) = \mathscr{L}$. To select one among these Petri net systems we choose a given performance index and solving an appropriate IPP (Integer Programming Problem) we determine a Petri net system that minimizes the considered performance index[2]. In particular, if $f(M_0, Pre, Post)$ is the considered performance index, an identification problem can be formally stated as follows.

Problem 10.6. *Let us consider the identification problem (10.1) and let $f(M_0, Pre, Post)$ be a given performance index. The solution to the identification problem (10.1) that minimizes $f(M_0, Pre, Post)$ can be computed by solving the following IPP*

$$\begin{cases} \min & f(M_0, Pre, Post) \\ s.t. & \mathscr{G}_m(\mathscr{E}, \mathscr{D}). \end{cases} \tag{10.10}$$

■

Of particular interest are those objective functions that are linear in the unknowns, so that the problem to solve is a linear integer programming problem. As example of a linear objective function, assume we want to determine a Petri net system that minimizes the weighted sum of the tokens in the initial marking and of the arc weights. The general case is:

$$f(M_0, Pre, Post) = \sum_{i=1}^{m} \left(a_i \cdot M_0(p_i) + \left(\sum_{j=1}^{n} b_{i,j} \cdot Pre(p_i, t_j) + c_{i,j} \cdot Post(p_i, t_j) \right) \right), \tag{10.11}$$

where $a_i, b_{i,j}$ and $c_{i,j}$ are given coefficients.

A typical choice, that we follow in the rest of the paper, is that of choosing all coefficients equal to 1. In this case (11.6) can be rewritten:

$$f(M_0, Pre, Post) = \vec{1}_m^T \cdot M_0 + \vec{1}_m^T \cdot (Pre + Post) \cdot \vec{1}_n.$$

Example 10.7. Let $\mathscr{L} = \{\varepsilon, t_1, t_1 t_1, t_1 t_2, t_1 t_1 t_2, t_1 t_2 t_1\}$ and $m = 2$, thus $k = 3$. Assume that we want to determine the Petri net system that minimizes the sum of initial tokens and all arcs such that $L_3(N, M_0) = \mathscr{L}$. This requires the solution of an IPP of the form (10.10) where

$$\mathscr{E} = \mathscr{E}' = \{(\varepsilon, t_1), (t_1, t_1), (t_1, t_2), (t_1 t_2, t_1), (t_1 t_1, t_2)\}$$

and

$$\mathscr{D} = \mathscr{D}' = \{(\varepsilon, t_2), (t_1 t_2, t_2), (t_1 t_1, t_1)\}.$$

The procedure identifies a net system with

$$Pre = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Post = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad M_0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

namely the net system in Figure 10.1.a. ■

[2] Clearly, also in this case the solution may be not unique.

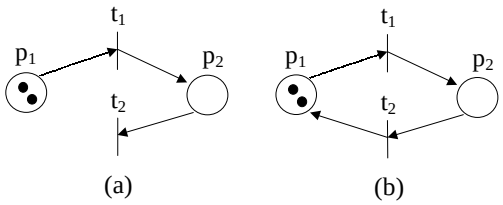


Figure 10.1: (a) The Petri net system of Example 10.7; (b) the Petri net of the same example when the additional constraint $m_1 + m_2 = \text{const}$ is added.

10.2 Extended identification procedure for free labeled Petri nets

In many cases the available information on the net to identify is not limited to samples of its language. As an example, it may be known that the net has a particular structure, or some partial information on the initial marking (in terms of available resources) may be given.

In this section it is shown how this additional information can easily be incorporated in the identification procedure previously described.

10.2.1 Structural constraints

P-vectors

Assume that some places of the net are known to belong to a conservative component, i.e., the weighted sum of their tokens in the component remains constant during any evolution. This is equivalent to say that some P-invariants for the net are known (see Definition 3.3).

More generally the knowledge of any P-vector may be taken into account adding to Problem 10.10 a suitable set of constraints.

- Assume $\vec{x} \in \mathbb{R}^m$ is *P-invariant*. We need to add to Problem 10.10 the following constraint
$$\vec{x}^T(Post - Pre) = \vec{0}_n^T$$
that imposes $\vec{x}^T \cdot C = \vec{0}_n^T$.

- Assume $\vec{x} \in \mathbb{R}^m$ is *P-increasing*. We need to add to Problem 10.10 the following constraints
$$\begin{cases} \vec{x}^T(Post - Pre) \geq \vec{0}_n^T \\ \vec{x}^T(Post - Pre)\vec{1}_n \geq 1 \end{cases}$$
The first constraint imposes that $\vec{x}^T \cdot C \geq \vec{0}_n^T$ and the second one imposes that $\vec{x}^T \cdot C \neq \vec{0}_n^T$.

- Assume $\vec{x} \in \mathbb{R}^m$ is *P-decreasing*. We need to add to Problem 10.10 the following constraints

$$\begin{cases} \vec{x}^T(Post - Pre) \leq \vec{0}_n^T \\ \vec{x}^T(Post - Pre)\vec{1}_n \leq -1 \end{cases}$$

The first constraint imposes that $\vec{x}^T \cdot C \leq \vec{0}_n^T$ and the second one imposes that $\vec{x}^T \cdot C \neq \vec{0}_n^T$.

T-vectors

Assume that a given firing sequence is known to be stationary, i.e., the number of the tokens of the net is not modified by the firing of this sequence. This is equivalent to say that some T-invariants for this net are known (see Definition 3.3).

More generally the knowledge of any T-vector may be taken into account adding to Problem 10.10 a suitable set of constraints.

- Assume $\vec{y} \in \mathbb{R}^n$ is *T-invariant*. We need to add to Problem 10.10 the following constraint

$$(Post - Pre)\vec{y} = \vec{0}_m$$

that imposes $C \cdot \vec{y} = \vec{0}_m$.

- Assume $\vec{y} \in \mathbb{R}^n$ is *T-increasing*. We need to add to Problem 10.10 the following constraints

$$\begin{cases} (Post - Pre) \cdot \vec{y} \geq \vec{0}_m \\ \vec{1}_m^T(Post - Pre)\vec{y} \geq 1 \end{cases}$$

The first constraint imposes that $C \cdot \vec{y} \geq \vec{0}_m^T$ and the second one imposes that $C \cdot \vec{y} \neq \vec{0}_m^T$.

- Assume $\vec{x} \in \mathbb{R}^m$ is *T-decreasing*. We need to add to Problem 10.10 the following constraints

$$\begin{cases} (Post - Pre) \cdot \vec{y} \leq \vec{0}_m \\ \vec{1}_m^T(Post - Pre)\vec{y} \leq -1 \end{cases}$$

The first constraint imposes that $C \cdot \vec{y} \leq \vec{0}_m^T$ and the second one imposes that $C \cdot \vec{y} \neq \vec{0}_m^T$.

Example 10.8. Let us consider again the case of Example 10.7 but assume the net is known to be conservative. In particular, the sum of the tokens in places p_1 and p_2 remains constant. To this aim we solve an IPP of the form (10.10) with the addition of a constraint of the form of (10.2.1), where $\vec{x} = [1\ 1]^T$. We identify a net system with

$$Pre = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad Post = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad M_0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

namely the net in Figure 10.1.b. ■

Net subclasses

In this subsection we consider the constraints that we need to add to Problem 10.10 to ensure that the identified net belongs to some particular subclasses of nets defined in Subsection 3.4.

- *Ordinary*:
$$Pre, Post \in \{0,1\}^{m\times n}.$$
- *Marked graph*:
$$\begin{cases} Pre\cdot\vec{1}_n = 1 \\ Post\cdot\vec{1}_n = 1. \end{cases}$$
- *State machine*:
$$\begin{cases} \vec{1}_m^T\cdot Pre = 1 \\ \vec{1}_m^T\cdot Post = 1. \end{cases}$$

All these results follow immediately from the definitions in Subsection 3.4.

Constraints on the initial marking

A type of general constraints that can be imposed on the markings of a Petri net is called GMEC (Generalized Mutual Exclusion Constraint) and can be represented by the couple $(\vec{w}, k)$, where $\vec{w} \in \mathbb{Z}^m$, $k \in \mathbb{Z}$. This constraint defines a set of legal markings:

$$\mathscr{M}(\vec{w}, k) = \{M \in \mathbb{N}^m \mid \vec{w}^T M \leq k\}.$$

If it is known that $M_0 \in \mathscr{M}(\vec{w}, k)$ then the constraint

$$\vec{w}^T M_0 \leq k,$$

should be added to Problem 10.10.

For example consider a Petri net with an initial marking that can not contain a number of tokens greater than 1 in places p_1 and p_2. In this case we need to impose as additional constraint

$$M(p_1) + M(p_2) \leq 1.$$

Structural decomposition

We can impose a *structural decomposition* of the net in a given number r of subnets. Let

$$P = P_1 \cup P_2 \cup \ldots \cup P_r$$

be a given partition of P. Assume that for all $t \in T$ we are given a set $\Pi(t) \subset \{1, \ldots, r\}$ such that $q \in \Pi(t)$ implies ${}^\bullet t^\bullet \cap P_q = \emptyset$. In plain words, $\Pi(t)$ represents the set of indices of P_q's such that t has no input/output arc from/to a place in P_q.

This can be imposed adding to Problem (10.10) the following set of linear constraints for all $t \in T$:

$$\sum_{q \in \Pi(t)} \sum_{p \in P_q} (Pre(p,t) + Post(p,t)) = 0.$$

10.2.2 Synthesis of bounded Petri net systems from regular languages

In this section we assume that the net system we want to synthesize is bounded, and thus its language is regular. The language is given in terms of a finite state automaton $G = (Q, T, \delta, q_0)$ where Q is the set of states, the alphabet T is the set of transitions of the net, $\delta : Q \times T \to Q$ is the transition function, and q_0 is the initial state.

We consider the following problem.

Problem 10.9. *Let $G = (Q, T, \delta, q_0)$ be a given finite state automaton. Given a set of places P of cardinality m and a nonnegative integer K, we want to identify the structure of a bounded net $N = (P, T, Pre, Post)$ and an initial marking M_0 such that $L(N, M_0) = \mathscr{L}(G)$, and*

$$\max_i M_0(p_i) + k \cdot \max_{i,j} Post(i,j) \leq K.$$

The unknowns we want to determine are the elements of the two matrices Pre, $Post \in \mathbb{N}^{m \times n}$ and the elements of the vector $M_0 \in \mathbb{N}^m$. ■

The identification procedure previously defined considers sequences of bounded length. An automaton is able to generate sequences of unbounded length every time that there is a cycle. Thus we have to distinguish between sequences that pass through cycles (that can be extended indefinitely) and sequences that do not pass through cycles (whose length is finite).

We say that a firing sequence $\sigma' \prec \sigma$ if σ' is a strict prefix of σ, i.e., if $\sigma = t_{\alpha_1} t_{\alpha_2} \dots t_{\alpha_k}$ and $\sigma' = t_{\alpha_1} t_{\alpha_2} \dots t_{\alpha_r}$ with $r < k$. In following we denote as

$$\Gamma(G) = \{\sigma \in T^* \mid \delta(q,\sigma) = q \ \wedge \ \forall \sigma', \sigma'' \prec \sigma \ \delta(q,\sigma') \neq \delta(q,\sigma'')\} \tag{10.12}$$

the set of elementary cycles of the automaton. We define the set of the firing vectors associated to the firing sequences in $\Gamma(G)$ as

$$Y(G) = \{\vec{\xi} \in \mathbb{N}^n \mid \exists\, \sigma \in \Gamma(G) \ : \ \vec{\xi} = \pi(\sigma)\}. \tag{10.13}$$

We define the set of sequences that are generated by the automaton without passing through a cycle as

$$\mathscr{L}_{ac}(G) = \{\sigma \in T^* \mid \forall\ u, v \preceq \sigma,\ u \neq v \Rightarrow\ \delta(q_0, u) \neq \delta(q_0, v)\} \subseteq \mathscr{L}(G), \tag{10.14}$$

where $\mathscr{L}(G)$ denotes the language generated by the automaton.

Finally we define the following sets

$$\mathscr{E}' = \{(\sigma, t) \mid \sigma \in \mathscr{L}_{ac}(G), \sigma t \in \mathscr{L}(G)\},$$

$$\mathscr{E} = \mathscr{E}'|_{\equiv}, \tag{10.15}$$

$$\mathscr{D}' = \{(\sigma, t) \mid \sigma \in \mathscr{L}_{ac}(G), \sigma t \notin \mathscr{L}\},$$

$$\mathscr{D} = \mathscr{D}'|_{\equiv}, \tag{10.16}$$

where $\mathscr{E}$ and $\mathscr{D}$ are the sets containing only one element of each equivalent class for the $\equiv$ relation in Definition 10.2.

Theorem 10.10. *A bounded net system $\langle N, M_0\rangle$ is a solution of the identification problem (10.9) if and only if it satisfies the following set of linear algebraic constraints*

$$\begin{cases} \mathscr{G}_m(\mathscr{E},\mathscr{D}) & (a) \\ (Post - Pre)\cdot\vec{\xi} = \vec{0} \quad \forall\, \vec{\xi} \in Y(G) & (b) \end{cases} \tag{10.17}$$

where $\mathscr{E}$ and $\mathscr{D}$ are given in (10.15), (10.16)*, respectively.*

Proof.

We just give a sketch of the proof. First consider a word $\sigma' = \sigma t$ where $\sigma \in \mathscr{L}_{ac}(G)$. Then $\mathscr{G}_m(\mathscr{E},\mathscr{D})$ contains enough constraints to ensure that σ and σ', or σ and not σ' are generated by any net solution of (18) according to the case $\sigma' \in \mathscr{L}(G)$ or $\sigma' \notin \mathscr{L}(G)$.

Consider next a word $\sigma' = \sigma t$ where $\sigma \in \mathscr{L}(G) \setminus \mathscr{L}_{ac}(G)$. Then $\sigma = \sigma_1 u \sigma_2$ where $u \neq \epsilon$ and $\delta(q_0, \sigma_1) = \delta(q_0, \sigma_1 u)$, hence $\sigma_1\sigma_2 \in \mathscr{L}(G)$, and the firing count vector of u is a T-invariant of any net solution of (18). In other words, such a net generates $\sigma\, t$ if and only if it generates $\sigma_1 \sigma_2\, t$, and since $\sigma_1 \sigma_2$ is strictly shorter than σ, the theorem follows by induction on words.

□

Example 10.11. Let us consider the finite state automaton G in Figure 10.2.a. It holds $Y(G) = \{[1\ 1]^T\}$ and $\mathscr{L}_{ac}(G) = \{\varepsilon, t_1, t_1 t_1\}$ thus $\mathscr{E} = \mathscr{E}' = \{(\varepsilon, t_1), (t_1, t_1), (t_1, t_2), (t_1 t_1, t_2)\}$ and $\mathscr{D} = \mathscr{D}' = \{(\varepsilon, t_2), (t_1 t_1, t_1)\}$. Now, assume that we want to determine the Petri net system that minimizes the sum of initial tokens and all arcs.

For $m = 1$ we get no feasible solution, while for $m = 2$ we find the net system in Figure 10.1.b whose reachability graph is shown in Figure 10.2.b. Note that in this particular case the reachability graph of the net is isomorphic to the given automaton G, but this is not necessarily guaranteed by our procedure. The problem of finding a net whose reachability graph is isomorphic to that of an automaton is addressed in the theory of regions as discussed in Chapter 5. ■

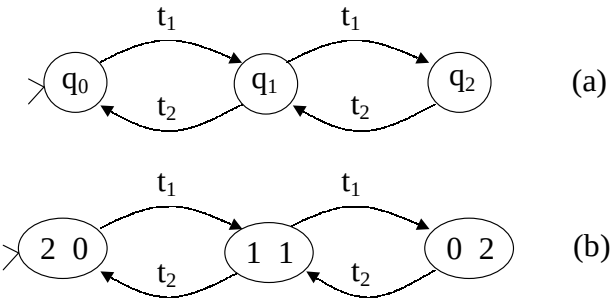


Figure 10.2: (a) The finite state automaton G of Example 10.11; (b) the reachability graph of the identified net system.

10.2.3 Optimizing the number of places

In the previous formulation we assumed that the number m of places is given. In this subsection we remove this assumption and consider the following identification problem.

Problem 10.12. *Let us consider an identification problem in the form (10.1) where m is only known to be less than or equal to a given value $\bar{m}$, and let $f(m, M_0, Pre, Post)$ be a given performance index. The solution of the identification problem that minimizes $f(m, M_0, Pre, Post)$ with the smallest number of places can be computed solving the following nonlinear IPP*

$$\begin{cases} \min\limits_{m \le \bar{m}} & \min & f(m, M_0, Pre, Post) \\ s.t. & & \mathscr{G}_m(\mathscr{E}, \mathscr{D}). \end{cases} \tag{10.18}$$

A trivial solution to the identification problem 10.12 consists in solving IPP of the form (10.10) for increasing values of m, until a feasible solution is obtained.

The following theorem provides an alternative approach to do this, that simply requires the solution of one IPP, while guaranteeing the optimality of the solution both in terms of minimum number of places and in terms of the chosen performance index.

Theorem 10.13. *Solving the identification problem 10.12 is equivalent to solving the following IPP:*

$$\begin{cases} \min & \overline{K} \cdot \vec{1}_{\bar{m}}^T \vec{z} + f(\bar{m}, M_0, Pre, Post) \\ s.t. & \mathscr{G}_{\bar{m}}(\mathscr{E}, \mathscr{D}) \\ & K \cdot \vec{z} - Pre \cdot \vec{1}_n - Post \cdot \vec{1}_n \ge \vec{0}_{\bar{m}} \\ & z_{i+1} \le z_i, \quad i = 1, \ldots, \bar{m} - 1 \\ & \vec{z} \in \{0,1\}^{\bar{m}} \end{cases} \tag{10.19}$$

for a sufficiently large constant $\overline{K}$ ($\overline{K}$ must be such that the minimization of the first term of the objective function has priority over the minimization of its second term).

In particular, let us denote as $\vec{z}^*$, $\bar{M}_0^*$, $\overline{Pre}^*$ and $\overline{Post}^*$ the solution of (10.19), and let m^* be the number of nonzero components of $\vec{z}^*$.

Let M_0^* be the vector obtained from $\bar{M}_0^*$ by only keeping its first m^* components. Analogously, let Pre^* and $Post^*$ be the matrices obtained from $\overline{Pre}^*$ and $\overline{Post}^*$, respectively, by only keeping their first m^* rows.

Then, m^*, M_0^*, Pre^*, $Post^*$ is a solution of the identification problem 10.12.

Proof.

Let us first observe that if $z_i = 1$, then the corresponding constraint

$$K - Pre(p_i, \cdot) \cdot \vec{1}_n - Post(p_i, \cdot) \cdot \vec{1}_n \geq 0$$

is trivially verified being K a very large constant.

On the contrary, if $z_i = 0$, the new constraint becomes

$$-Pre(p_i, \cdot) \cdot \vec{1}_n - Post(p_i, \cdot) \cdot \vec{1}_n \geq 0$$

whose only admissible solution is $Pre(p_i, \cdot) = Post(p_i, \cdot) = \vec{0}_n^T$. Place p_i is in this case redundant and can be removed without affecting the language of the net.

Since our main goal in (10.19) is that of minimizing $\vec{1}_m^T \vec{z}$, the optimal solution $\vec{z}^*$ will have as many zeros as possible, compatibly with the other constraints. Moreover, being $z_{i+1} \leq z_i, \quad i = 1, \ldots, \bar{m} - 1$, zero is assumed by the last components of $\vec{z}^*$. □

In the previous theorem the chosen performance index allows one to solve in one shot a two-criteria optimization problem using a procedure based on *global priorities* [14]. In this case we have a multi–objective performance in which the goals have different priorities. We first look for all solutions that optimize the first goal, i.e., the number of places, and then among them we look for those that optimize the second goal.

Example 10.14. Let

$$\mathscr{L} = \{\varepsilon, t_1, t_1 t_2, t_1 t_3, t_1 t_2 t_1, t_1 t_2 t_3, t_1 t_3 t_1, t_1 t_3 t_2\}$$

thus $k = 3$. Assume that we want to determine the Petri net system that minimizes the sum of initial tokens and all arcs such that $L_3(N, M_0) = \mathscr{L}$. This requires the solution of an IPP of the form (10.10) where

$$\mathscr{E} = \mathscr{E}' = \{(\varepsilon, t_1), (t_1, t_2), (t_1, t_3), (t_1 t_2, t_1), (t_1 t_2, t_3), (t_1 t_3, t_1), (t_1 t_3, t_2)\}$$

and

$$\mathscr{D} = \mathscr{D}'\{(\varepsilon, t_2), (\varepsilon, t_3), (t_1, t_1), (t_1 t_2, t_2), (t_1 t_3, t_3)\}.$$

We assume that the total number of places is bounded by $\bar{m} = 5$ and we choose the constant $K = \bar{m} \cdot n \cdot 10^4 = 15 \cdot 10^4$.

The procedure identifies a net system with $m = 3$ and

$$Pre = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad Post = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

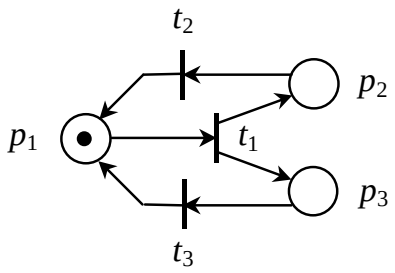


Figure 10.3: The Petri net system of Example 10.14.

$$M_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$

namely the net system in Figure 10.3. ■

10.3 λ-free labeled Petri nets

In this section we show how the above results can be extended to the case of λ-free labeled Petri nets.

We consider $\varphi : T \to E$ a labeling function over E and we denote the set of transitions that are labeled by symbol e as:

$$T_e = \{t \in T \mid \varphi(t) = e\} = \{t_1^e, \ldots, t_{n_e}^e\}, \quad e \in E$$

where $n_e = |T_e|$. Obviously it holds

$$T = \bigcup_{e \in E} T_e,$$

i.e., the the labeling equivalence induces a partition of T.

We say that an event e is *ambiguous* if $n_e > 1$, i.e., there exists more than one transition t such that $\varphi(t) = e$, otherwise we say that the event e is *not-ambiguous*. Analogously, we say that a transition t is *ambiguous* if its label is also associated to other transitions, otherwise a transition t is said to be *not-ambiguous*.

We denote $w = \varphi(\sigma)$ the word of events associated to sequence σ.

Given a labeled Petri net system $\langle N, M_0 \rangle$ we define its λ-free labeled language as the set of words in E^* generated from the initial marking M_0, namely,

$$L^E(N, M_0) = \{w \in E^* \mid \exists \sigma \in T^* : M_0[\sigma\rangle, \varphi(\sigma) = w\}.$$

We also denote $L_k^E(N, M_0)$ the set of words in $L^E(N, M_0)$ of length less than or equal to $k \in \mathbb{N}$, i.e.,

$$L_k^E(N, M_0) = \{w \in L^E(N, M_0) \mid |w| \leq k\}.$$

Problem 10.15. *Assume we are given a set of places $P = \{p_1, \ldots, p_m\}$ and a set of transitions $T = \{t_1, \ldots, t_n\}$. Let $\varphi : T \to E$ be a given labeling function over E whose equivalence classes T_e are known. Let $\mathscr{L} \subset E^*$ be a given finite prefix-closed language over E^*, and*

$$k = \max_{w \in \mathscr{L}} |w|$$

be the length of the longest word in $\mathscr{L}$.

We want to identify the structure of a deterministic[3] *net $N = (P, T, Pre, Post)$ labeled by φ and an initial marking M_0 such that*

$$L_k^E(N, M_0) = \mathscr{L}.$$

We also assume that a nonnegative integer K is given such that the following condition holds:

$$\max_i M_0(p_i) + k \cdot \max_{i,j} Post(i, j) \leq K.$$

The unknowns we want to determine are the elements of the two matrices

$$Pre = \{e_{i,j}\} \in \mathbb{N}^{m \times n} \quad and \quad Post = \{o_{i,j}\} \in \mathbb{N}^{m \times n}$$

and the elements of the vector

$$M_0 = \begin{bmatrix} m_{0,1} & m_{0,2} & \cdots & m_{0,m} \end{bmatrix}^T \in \mathbb{N}^m.$$

■

Let us consider the following definitions that we will use in the following.

Definition 10.16. *Given two pairs (w, e) and (w', e') we say that*

$$(w, e) \equiv (w', e')$$

if $\forall\ e \in E\ \ |w|_e = |w'|_e$ and $e = e'$. ■

It is easy to verify that the relation introduced in Definition 10.16 is an equivalence relation (see Appendix B). Thus two pairs (w, e) and (w', e') having the same cardinality of the event $e \in E$, for each event e of the alphabet, and where e and e' are equivalent, belong to the same equivalence class.

Definition 10.17. *Let $\mathscr{L} \in E^*$ be a finite prefix-closed language and let $k \in \mathbb{N}$ be defined as in Problem 10.15.*

We define the following sets

$$\mathscr{E} = \{(w, e) \mid w \in \mathscr{L}, |w| < k, we \in \mathscr{L}\},$$

[3] Determinism is a desirable property and we assume the net enjoys it. However, it may also be possible to solve this problem without assuming that the net be deterministic.

$$\mathscr{E} = \mathscr{E}'|_{\equiv}, \tag{10.20}$$

$$\mathscr{D}' = \mathscr{D} = \{(w,e) \mid w \in \mathscr{L}, |w| < k, we \notin \mathscr{L}\},$$

$$\mathscr{D} = \mathscr{D}'|_{\equiv}, \tag{10.21}$$

where $\mathscr{E}$ and $\mathscr{D}$ are the sets containing only one element of each equivalent class for the $\equiv$ relation in Definition 10.16. ■

The following theorem provides a linear algebraic characterization of the deterministic labeled Petri net systems with m places, n transitions and labeling function φ such that $L_k^E(N, M_0) = \mathscr{L}$.

Theorem 10.18. *A solution to the identification problem (10.15) satisfies the following set of linear algebraic constraints*

$$\mathscr{G}_m(\mathscr{E}, \mathscr{D}, \varphi) \triangleq$$

$$\begin{cases}
M_w - Pre(\cdot, t_1^e) \geq -z_1^{e,w} \cdot \vec{K} \\
\quad \vdots \\
M_w - Pre(\cdot, t_{n_e}^e) \geq -z_{n_e}^{e,w} \cdot \vec{K} \\
M_{we} - M_w - Post(\cdot, t_1^e) + Pre(\cdot, t_1^e) \leq z_1^{e,w} \cdot \vec{K} \\
M_{we} - M_w - Post(\cdot, t_1^e) + Pre(\cdot, t_1^e) \geq -z_1^{e,w} \cdot \vec{K} \\
\quad \vdots \\
M_{we} - M_w - Post(\cdot, t_{n_e}^e) + Pre(\cdot, t_{n_e}^e) \leq z_{n_e}^{e,w} \cdot \vec{K} \\
M_{we} - M_w - Post(\cdot, t_{n_e}^e) + Pre(\cdot, t_{n_e}^e) \geq -z_{n_e}^{e,w} \cdot \vec{K} \\
z_1^{e,w} + \ldots + z_{n_e}^{e,w} = n_e - 1 \\
z_1^{e,w}, \ldots, z_{n_e}^{e,w} \in \{0,1\} \\
\qquad\qquad \forall (w,e) \in \mathscr{E} & (a) \\
-K\bar{S}(w, t_j^e) + M_w - Pre(\cdot, t_j^e) \leq -\vec{1} \quad \forall (w,e) \in \mathscr{E} : |T_e| > 1,\ \forall t_j^e \in T_e & (b) \\
\vec{1}^T \bar{S}(w, t_j^e) \leq m - z_j^{e,w} \qquad \forall (w,e) \in \mathscr{E} : |T_e| > 1,\ \forall t_j^e \in T_e & (c) \\
-KS(w, t_j^e) + M_w - Pre(\cdot, t_j^e) \leq -\vec{1} \quad \forall (w,e) \in \mathscr{D}, \quad \forall t_j^e \in T_e & (d) \\
\vec{1}^T S(w, t_j^e) \leq m - 1 \qquad \forall (w,e) \in \mathscr{D}, \quad \forall t_j^e \in T_e & (e) \\
M_w \in \mathbb{N}^m, \quad \forall w \in \mathscr{L} & (f) \\
Pre, Post \in \mathbb{N}^{m \times n} & (g) \\
S(w, t_j^e) \in \{0,1\}^m & (h) \\
\bar{S}(w, t_j^e) \in \{0,1\}^m & (i)
\end{cases} \tag{10.22}$$

where $\mathscr{E}$ and $\mathscr{D}$ are defined in Eq. (10.20)*, Eq.* (10.21)*, respectively.*

Proof.

- Assume that $we \in \mathscr{L}$, where $w \in E^*$ and $e \in E$. Then *at least* one transition $t_j^e \in T_e$ should be enabled at M_w, or equivalently, for *at least* one $t_j^e \in T_e$ it should hold:

$$M_w \geq Pre(\cdot, t_j^e).$$

Thus, following again the procedure in Appendix C to convert the logical *or* operator in terms of linear constraints, we can write:

$$\begin{cases} M_w - Pre(\cdot, t_1^e) \geq -z_1^{e,w} \cdot \vec{K} \\ \vdots \\ M_w - Pre(\cdot, t_{n_e}^e) \geq -z_{n_e}^{e,w} \cdot \vec{K} \\ z_1^{e,w} + \ldots + z_{n_e}^{e,w} = n_e - 1 \\ z_1^{e,w}, \ldots, z_{n_e}^{e,w} \in \{0,1\} \end{cases}$$

If $z_j^{e,w} = 0$ it means that $t_j^e \in T_e$ may fire at M_w, and the marking M_{we} reached after its firing is

$$M_{we} = M_w + Post(\cdot, t_j^e) - Pre(\cdot, t_j^e).$$

that satisfies the following set of linear inequalities:

$$\begin{cases} M_{we} - M_w - Post(\cdot, t_1^e) + Pre(\cdot, t_1^e) \leq z_1^{e,w} \cdot \vec{K} \\ M_{we} - M_w - Post(\cdot, t_1^e) + Pre(\cdot, t_1^e) \geq -z_1^{e,w} \cdot \vec{K} \\ \vdots \\ M_{we} - M_w - Post(\cdot, t_{n_e}^e) + Pre(\cdot, t_{n_e}^e) \leq z_{n_e}^{e,w} \cdot \vec{K} \\ M_{we} - M_w - Post(\cdot, t_{n_e}^e) + Pre(\cdot, t_{n_e}^e) \geq -z_{n_e}^{e,w} \cdot \vec{K} \end{cases}$$

Now, if we want the net to be deterministic, we must impose that, whenever $|T_e| > 1$, only one transition $t_j^e \in T_e$ is enabled at M_w.

From the above constraints we know that transition $t_k^e \in T_e$ such that $z_k^{e,w} = 0$ is enabled at M_w. Thus, we need to impose additional constraints in order to be sure that, for all the other transitions t_j^e, $j \neq k$, for which $z_j^{e,w} = 1$, it holds that

$$M_w - Pre(\cdot, t_j^e) \ngeq \vec{0}.$$

In order to do this, for all $t_j^e \in T_e$ we introduce a vector of binary variables $\bar{S}(w, t_j^e)$ that satisfies the following set of linear inequalities:

$$\begin{cases} -K\bar{S}(w, t_j^e) + M_w - Pre(\cdot, t_j^e) \leq -\vec{1} \\ \vec{1}^T \bar{S}(w, t_j^e) \leq m - z_j^{e,w} \end{cases}$$

If $z_j^{e,w} = 0$, then all entries of $\bar{S}(w, t_j^e)$ may be unitary, thus adding no additional constraint (the corresponding inequality is trivially verified). On the contrary, if $z_j^{e,w} = 1$, then at least one entry of $\bar{S}(w, t_j^e)$ is null, thus making t_j^e not enabled at M_w. Being $z_1^{e,w} + \ldots + z_{n_e}^{e,w} = n_e - 1$, we can be sure that only one transition labeled e is enabled at M_w.

- Assume $w \in \mathscr{L}$ and $we \notin \mathscr{L}$. Then for all $t_j^e \in T_e$ the following set of linear constraints should be satisfied:

$$\begin{cases} -K \cdot S(w, t_j^e) + M_w - Pre(\cdot, t_j^e) \leq -\vec{1}_m \\ \vec{1} \cdot S(w, t_j^e) \leq m-1 \\ S(w, t_j^e) \in \{0,1\}^m. \end{cases}$$

□

As in the free labeled case, it may be possible to associate to our constraints a performance index to solve an integer programming problem and find, if there exists, the optimal solution.

Example 10.19. Let us now consider a numerical example taken from [79] where $m = n = 3$, $\varphi(t_1) = a$, $\varphi(t_2) = \varphi(t_3) = b$ and the net language is $\mathscr{L}' = \{a^r b^q,\ r \geq q \geq 0\}$.

Assume we want to minimize the sum of initial tokens and the sum of all arcs.

Let us first assume that $k = 3$, thus

$$\mathscr{L} = \{\varepsilon, a, aa, ab, aaa, aab\}.$$

This implies that

$$\mathscr{E} = \mathscr{E}' = \{(\varepsilon, a), (a, a), (a, b), (aa, a), (aa, b)\}$$

and

$$\mathscr{D} = \mathscr{D}' = \{(\varepsilon, b), (ab, a), (ab, b)\}.$$

The resulting net system is such that

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

$$Pre = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ Post = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

namely that represented in Figure 10.4.a.

Note that another optimal solution is given by the net in figure (b) if we remove the arc from t_2 to p_1 and the arc from p_3 to t_3.

Then, assume $k = 4$, thus

$$\mathscr{L} = \{\varepsilon, a, aa, ab, aaa, aab, aaaa, aaab, aabb\}.$$

This implies that

$$\mathscr{E} = \mathscr{E}' = \{(\varepsilon, a), (a, a), (a, b), (aa, a), (aa, b), (aaa, a), (aaa, b), (aab, b)\}$$

and

$$\mathscr{D} = \mathscr{D}' = \{(\varepsilon, b), (ab, a), (ab, b), (aab, a)\}.$$

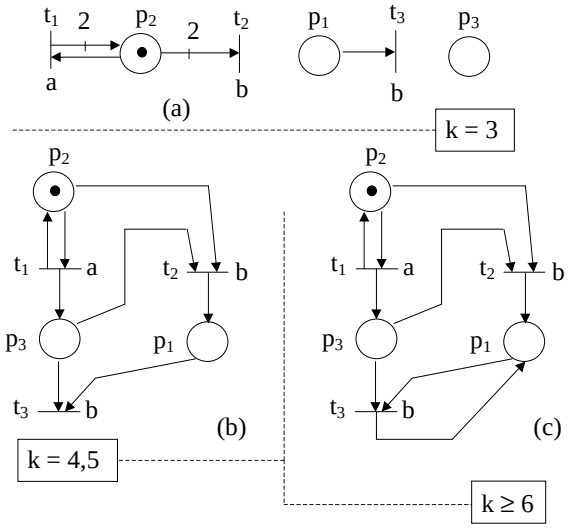


Figure 10.4: The results of Example 10.19.

The resulting net system is such that

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

$$Pre = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \; Post = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

namely that represented in Figure 10.4.b.

The same net system is also obtained if $k = 5$, while the net system in figure (c) is obtained if $k \geq 6$ (that coincides with the net in [79]). ■

Finally, we note that with the technique presented in the previous section we can also lift the requirement that the number of places is known.

It is also possible to deal with the case in which the cardinality of the set T_e for all $e \in E$ is not known a priori but only an upper bound on its value is known. This extension however is not straightforward and it will be left for future research.

10.4 Complexity of the identification procedure

In this section we discuss the complexity of the IPPs we must solve to identify a net. This complexity is given in terms of number of constraints and number of unknowns. Note however that it is well known that an IPP is an NP-hard problem itself.

10.4.1 Free labeled nets

Let n be the cardinality of T, k the length of the longest string in $\mathscr{L}$, and ν_r (for $r = 0,\ldots,k$) the number of pairs $(\sigma, t) \in \mathscr{E}$ such that $|\sigma t| = r$.

Then the constraint set (10.3) contains $\sum_{r=1}^{k} \nu_r$ constraints of type (a) and $\sum_{r=0}^{k-1}(n\nu_r - \nu_{r+1})$ constraints of type (b) and of type (c). The total number of scalar constraints is thus:

$$c_{\text{free}} = m\left(\sum_{r=1}^{k} \nu_r\right) + (m+1)\left(\sum_{r=0}^{k-1}(n\nu_r - \nu_{r+1})\right).$$

The total number of unknowns is

$$u_{\text{free}} = m + 2(m \times n) + m\left(\sum_{r=0}^{k-1}(n\nu_r - \nu_{r+1})\right).$$

Note that given a value of k and of n, it is possible to find a worst case bound for $\rho = \sum_{r=0}^{k-1}(n\nu_r - \nu_{r+1})$. In fact, it holds:

$$\rho = \textstyle\sum_{r=0}^{k-1}(n\nu_r - \nu_{r+1}) = n\nu_0 + (n-1)\left(\sum_{r=1}^{k-1}\nu_r\right) - \nu_k = n + (n-1)\left(\sum_{r=1}^{k-1}\nu_r\right) - \nu_k.$$

This expression is maximized if we assume $\nu_k = 0$ while all other ν_r must take the largest value, i.e., $\nu_r = n^r$. Hence we have

$$\rho \le n + (n-1)(n + \cdots + n^{k-1}) = n^k,$$

and the total number of unknowns in the worst case is

$$u_{\text{free}} = m + 2(m \times n) + m\, n^k = m(1 + 2n + n^k) = \mathscr{O}(m\, n^k),$$

i.e., it has exponential complexity with respect to k.

10.4.2 λ-free labeled Petri nets

Let $\tau = \max_{e\in E}|T_e|$, and as we have considered in the previous subsection, k be the length of the longest string in $\mathscr{L}$, and ν_r (for $r = 0,\ldots,k$) be the number of pairs $(w, e) \in \mathscr{E}$ such that $|we| = r$.

In the worst case the set (10.22) has

$$c_{\lambda\text{-free}} = [(4m+1)\tau + 1]\left(\sum_{r=1}^{k} \nu_r\right) + (m+1)\tau\left(\sum_{r=0}^{k-1}(n\nu_r - \nu_{r+1})\right)$$

constraints. Indeed, in such a case, we have $(3m\tau + 1)\left(\sum_{r=1}^{k}\nu_r\right)$ constraints of type (a), $(m+1)\tau\left(\sum_{r=1}^{k}\nu_r\right)$ constraints of type (b) plus (c), and $(m+1)\tau\left(\sum_{r=0}^{k-1}(n\nu_r - \nu_{r+1})\right)$ constraints of type (d) and (e).

Moreover, we have that the number of unknowns is

$$u_{\lambda\text{-free}} = m + 2mn + m\left(\sum_{r=1}^{k} \nu_r\right) + \tau\left(\sum_{r=1}^{k} \nu_r\right) + m\tau\left(\sum_{r=1}^{k} \nu_r\right) + m\tau\left(\sum_{r=0}^{k-1} (n\nu_r - \nu_{r+1})\right)$$

where each term corresponds, respectively, to: M_0; Pre and $Post$; M_w; the binary variables $z_j^{e,w}$; the binary vectors $\bar{S}(w, t_j^e)$; the binary vectors $S(w, t_j^e)$.

As shown in the previous subsection we can take:

$$\rho \leq n + (n-1)(n + \cdots + n^{k-1}) = n^k,$$

and then the total number of unknowns in the worst case is

$$u_{\lambda\text{-free}} = \mathcal{O}(m\tau n^k),$$

and keeping in mind that $\tau \leq n$ we can also write

$$u_{\lambda\text{-free}} = \mathcal{O}(mn^{k+1}).$$

Also in this case we have an exponential complexity with respect to k.

10.5 Numerical simulations

In this section we analyze the complexity of the identification approach here proposed in terms of computational time required to get an admissible solution, that may also be optimal according to a given performance criterion.

In particular, we want to investigate how the computational time depends on the cardinality of the set of finite length strings that describe the language, and on the chosen performance index.

To this aim we consider the language generated by a particular Petri net system that models a sender-receiver process. Different cases are examined with different number of places and transitions, and thus different languages generated.

As pointed out in the following of the section, the numerical simulations we carried out enabled us to conclude that the computational time becomes prohibitive for languages that are described by a large number of finite length strings, if we want to determine a solution that is *optimal* with respect to a given performance index. On the contrary, computational times are negligible if we limit to consider any *admissible* net system, e.g., the first admissible solution computed by CPLEX when solving the optimization problem. We believe that this is not a drawback of our procedure because in effect, when solving identification problems like this, the main requirement is that of determining an *admissible* solution, not necessarily an optimal one.

Let us consider the Petri net system in Figure 10.5 consisting of $2(q+1)$ places and $2q$ transitions. It models a sender-receiver process. In particular, places p_i, with $i = 1, \ldots q$, model

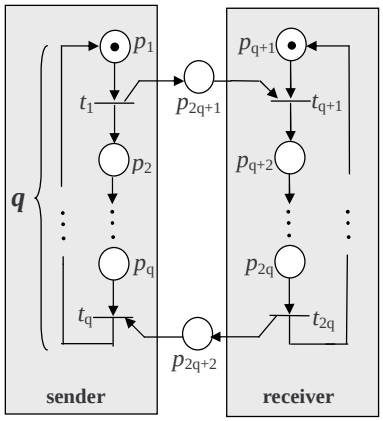


Figure 10.5: The sender-receiver process.

the sender process; places p_i, with $i = q+1,...,2q$, model the receiver process, and places p_i, with $i = 2q+1, 2q+2$, correspond to the communication channels between the two processes. When place p_1 is marked (as in Figure 10.5) the sender is ready to transmit a message. After the firing of t_1 the message is in the channel, ready to be received, provided that the receiver is also ready, namely that place p_{q+1} is marked. Now, transitions $t_2, \ldots, t_{q-1}, t_{q+1}, \ldots, t_{2q}$ can fire. In particular the firing of t_{2q} corresponds to the acknowledge from the receiver. Finally, the receipt from the sender is modeled by transition t_q.

In this section we present the results of various identification problems carried out considering different values of q, namely $q = 2, \ldots, 6$. In particular, our goal here is that of synthesizing a net system that generates the same language of the net system in Figure 10.5.

Note that for sake of brevity we do not report here the language $\mathscr{L} = L_k(N, M_0)$, but we limit to observe that $k = 2q$. Moreover, in order to obtain a net system that generates exactly the same language we also need to impose the minimal T-invariant $\vec{y} = \vec{1}_n$.

Note that we do not need to impose the P-invariants corresponding to the sender and the receiver. In fact, our requirement here is not that of obtaining exactly the net system in Figure 10.5, but a net system that generates the same language.

For any q we considered two different cases. In particular, using the notation of IPP (10.19), we assume:

(C1) $J = \bar{K} \cdot \vec{1}_{\bar{m}}^T \vec{z} + \vec{1}_{\bar{m}}^T \cdot M_0 + \vec{1}_{\bar{m}}^T \cdot (Pre + Post) \cdot \vec{1}_n$,

(C2) $J = \vec{1}_{\bar{m}}^T \cdot M_0$.

Therefore, in case (C1) we assume that the number of places is not known a priori, and our goal is that of determining among all the net systems that satisfy the given language specifications, that one who minimizes the number of places, the initial number of tokens and the arc weights.

In case (C2) we assign no weight on the number of places because we assume it is given. Our goal here is simply that of minimizing the number of tokens in the initial marking.

		First integer solution	Optimal solution (max 1 hour)	% after 1 hour
q=2	**C1**	0,03 sec (141)	0,03 sec (141)	0%
	C2	<0,01 sec (113)	<0,01 sec (113)	0%
q=3	**C1**	< 4 sec (598)		16,71%
	C2	<0,6 sec (607)	0,78 sec (2744)	0%
q=4	**C1**	< 29 sec (1706)		59,91%
	C2	<8 sec (2479)		50,00%
q=5	**C1**	<200 sec (3855)		75,05%
	C2	<20 sec (3625)		75,00%
q=6	**C1**	< 65 sec (6949)		97,61%
	C2	<20 sec (6067)		50,00%

Table 10.1: Numerical results

We carried out all numerical simulations using an appropriate tool we developed in MATLAB: given the language $\mathscr{L} = L_k(N, M_0)$, the structural constraints, and an upper bound on the number of places $\bar{m}$, it generates the set of constraints of IPP (10.19) in the syntax of CPLEX; then, given the desired performance index, the resulting IPP can be directly solved using ILOG CPLEX.

For both cases C1 and C2, and for any value of q, we limit the computational time to one hour. This constraint did not allow us to obtain the optimal solution in all cases examined.

Simulations have been run on a PC Athlon 64, 4000+ processor. Numerical results are summarized in Table 10.1.

In the first column we have reported the time (in seconds) and the number of iterations (between parenthesis) to find the first admissible solution.

In the second column we may have either an empty box or two numbers. The box is empty if CPLEX is not able to compute the optimal solution within an hour. If an optimal solution is determined within an hour, in the corresponding box we can read the time in seconds it took to compute it, and the number of iterations between parenthesis.

In the third column we reported a measure of the distance (as a percentage) between the optimal solution and the solution computed within an hour. If such a value is equal to 0% it means that the optimal solution has been obtained in the allowed time. As large is the percentage, as far the solution is from the optimum.

From Table 10.1 we can be easily observe that the optimal solution can be computed within an hour only for $q = 2$ (both in case C1 and C2), and for $q = 3$ (in case C2). In all the other cases the number of constraints was too high, and regardless of the considered performance

index, one hour was not enough to determine the optimal solution. In particular, the distance from the optimal solution in quite all cases examined also depend on the considered performance index (case C1 or C2).

Note however that this is not a serious limitation of our procedure because in general, when computing an identification problem, we are mainly interested in determining an admissible solution, rather than an optimal one. As it can be seen by looking at the first column of Table 10.1, the computational times are very very short also for large values of q if we consider an arbitrary solution, e.g., the first admissible one computed by CPLEX when solving an optimization problem.

Note that in case C1 we assumed that $\bar{m} = 2q + 2$ but we always found out a net with a minor number of places. This is not surprising because we are not imposing the P-invariants relative to the sender and the receiver, thus the structure of the net is different even if the language generated is the same.

Chapter 11

Identification of unbounded Petri nets from their coverability graph

Summary

In this chapter we solve the following problem: given an automaton that represents the coverability graph of a Petri net, determine a Petri net system whose coverability graph is isomorphic to the automaton.

The proposed approach requires solving an integer programming problem whose set of unknowns contains the elements of the pre and post incidence matrices and the initial marking of the net.

11.1 Coverability graph and properties

One technique used for the analysis of unbounded PNs is based on the construction of the coverability tree/graph (see also [69]) that provides a description in finite terms of the infinite reachability set. In particular, each node of the graph is labeled with an m dimensional row vector whose entries may either be integer number or may be equal to the special symbol ω, while arcs are elements in T. The symbol ω denotes that the marking of the corresponding place may grow indefinitely. Note that for all $n \in \mathbb{N}$ it holds $\omega > n$ and $\omega \pm n = \omega$.

Algorithm 11.1. ***Construction of the coverability tree for*** $\langle N, M_0 \rangle$**.**

1. *Label the root node q_0 with the initial marking M_0 and tag it "new".*
2. ***While** a node tagged "new" exists **do***
   a) *Select a node q tagged "new" and let M be its label.*
   b) ***For** all t enabled at M, i.e., such that $M \geq Pre(\cdot, t)$:*
      i. *Let $M' = M + C(\cdot, t)$ be the marking reached from M firing t.*
      ii. *Let $\bar{q}$ be the first node met on the backward path from q to q_0 whose label is $\bar{M} \lneq M'$. **If** such a node exists **then** for all $p \in P$ such that $M'(p) > \bar{M}(p)$ let $M'(p) = \omega$.*
      iii. *Add a new node q' and label it M'.*
      iv. *Add an arc labeled t from q to q'.*
      v. ***If** there exists already in the tree a node with label M', **then** tag node q' "duplicate", **else** tag it "new".*
   c) *Untag node q.*

■

From the coverability tree (CT) one can obtain the CG by fusing duplicate nodes with the untagged node with the same label: one can always convert a CT in a graph and viz. Note that, this algorithm can also be used to compute the reachability tree (and then the reachability graph) skipping the test at step 2.b.ii.

In the construction of the CT the existence of a sequence σ that leads from a marking $\bar{M}$ to a greater marking M' is identified at step 2.$(b).ii$. The components that by the repeated firing of such a sequence σ grow unbounded are denoted with a special symbol ω. Note that if $\bar{M}$ contains no ω components then σ is an increasing sequence. However, if $\bar{M}$ contains ω components we can only say that σ is increasing for all places p such that $\bar{M}(p) < \omega$: nothing can be said for the remaining places.

Definition 11.2. *Let us now consider a node q' labeled with a marking M' that has one or more components changed to ω at step* 2.$(b).ii$ *of Algorithm 11.1. With the notation used in step* 2.$(b).ii$*, we also denote $\bar{q}$ the node covered by q' and q the node father of q'.*

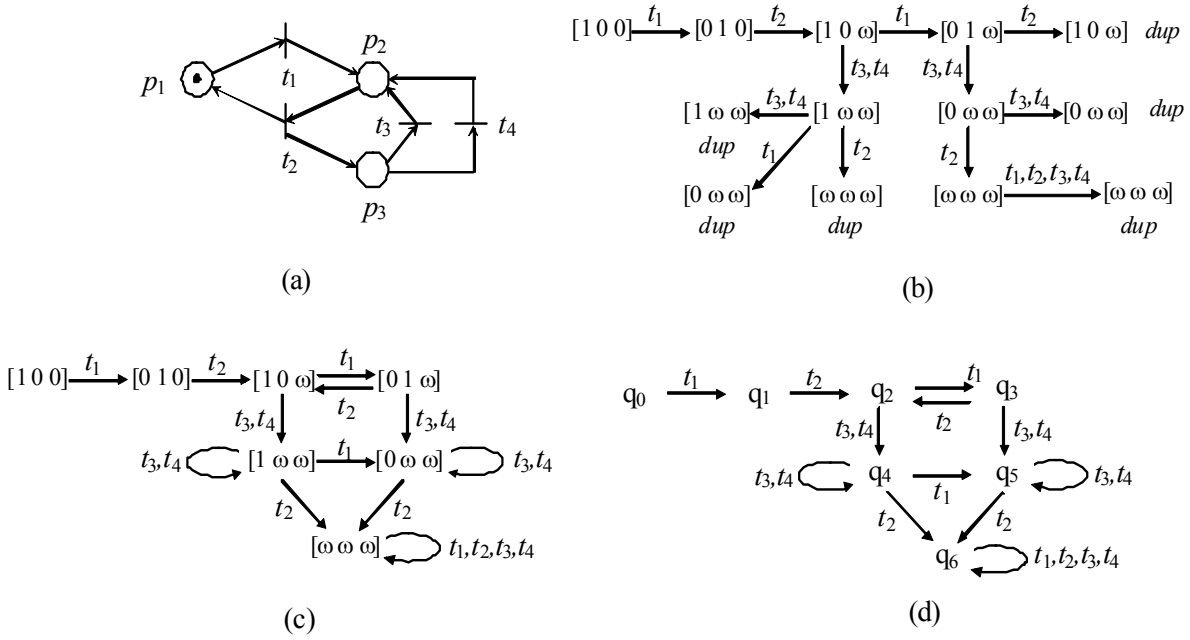


Figure 11.1: Net in Example 11.3.

- *Node q' is called an* ω-increasing node *and the corresponding marking is called an* ω-increasing marking.
- *The production associated to the path on the graph*

$$\pi : \bar{q} \xrightarrow{\sigma'} q \xrightarrow{t} q' \tag{11.1}$$

is called an ω-increasing production.

- *We also define for any production of the form in eq.* (11.1)*:* $v(\pi)$ the set of all nodes *of the production,* $s(\pi) = \bar{q}$ *the* start node *of the production,* $l(\pi) = q$ *the* last-but-one node, $e(\pi) = q'$ *the* end node*, and* $\ell(\pi) = \sigma' t$ *the corresponding sequence.* ■

Example 11.3. Let us consider the net in Figure 11.1.(a), whose CT is given in Figure 11.1.(b). The CG is shown in Figure 11.1.(c). This net and the successive ones do not have a particular physical meaning; we only use them to demonstrate properties of interest.

Sequence $t_1 t_2$ is a repetitive sequence that from $M_0 = [1\ 0\ 0]^T$ yields $[1\ 0\ 1]^T$ increasing the marking of place p_3: hence in the CG we have an ω-increasing production $\pi : q_0 t_1 q_1 t_2 q_2$ that from $s(\pi) = q_0$ leads to node $e(\pi) = q_2$ with label $M_2 = [1\ 0\ \omega]^T$.

Sequence t_3 from $[1\ 0\ \omega]^T$ yields $[1\ \omega\ \omega]^T$ increasing the marking of place p_2: hence in the CG we have an ω-increasing production $\pi : q_2 t_3 q_4$ that from $s(\pi) = q_2$ leads to node $e(\pi) = q_4$ with label $M_4 = [1\ \omega\ \omega]^T$ (in this case $l(\pi) = s(\pi)$). Note that although t_3 is an ω-increasing sequence, it is not a repetitive sequence because it decreases the marking of place p_3. ■

Now, let us show a sufficient condition for the ω-increasing productions.

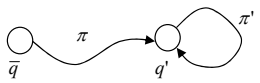


Figure 11.2: A sketch for Proposition 11.4.

Proposition 11.4. Let us consider a CG $\mathcal{G}$ of a Petri net and a node q' of $\mathcal{G}$. If there exists in the graph a node $\bar{q} \neq q'$ and a production π such that:

$$s(\pi) = \bar{q}, \quad e(\pi) = q'$$

and there exists another production π' such that:

$$s(\pi') = e(\pi') = q', \ \ell(\pi) = \ell(\pi') = \sigma, \ \nu(\pi) \cap \nu(\pi') = \{q'\}$$

then q' is an ω-increasing node and π is the corresponding ω-increasing production.

(see Fig. 11.2 where $\pi = \bar{q} \xrightarrow{\sigma} q'$ and $\pi' = q' \xrightarrow{\sigma} q'$).

Proof. *As explained in the construction of the CT, a node q' is labeled by an ω-increasing marking M' if and only if there exists a node $\bar{q}$ labeled by a marking $\bar{M}$ and a sequence σ such that $\bar{M}[\sigma\rangle\bar{M}'$ where $\bar{M} \lneq \bar{M}'$ and $M'(p) = \omega$ for all those places p such that $\bar{M}'(p) > \bar{M}(p)$, and $M'(p) = \bar{M}'(p)$ for all other places. Now since σ is enabled at $\bar{M}$ it is also enabled at the greater marking M'; however since its firing does not change the marking of a place p such that $M'(p) < \omega$, its firing from M' leads back to M'. Finally, we observe that π is $\omega-$increasing iff $e(\pi)$ (hence*[1] *all nodes in $\nu(\pi')$) contains at least one more $\omega-$component with respect to all nodes in $\nu(\pi) \setminus e(\pi)$; this implies that $\nu(\pi) \cap \nu(\pi') = e(\pi)$.* □

Proposition 11.4 gives us only sufficient conditions for the identification of ω-increasing sequences given a CG as shown in the following example.

Example 11.5. Let us consider the Petri net in Figure 11.3.(a), whose CG is given in Figure 11.3.(b).

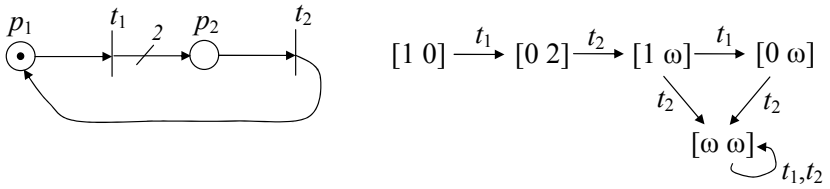
Sequence $t_1 t_2$ is a repetitive sequence that from $M_0 = [1\ 0]^T$ yields $[1\ 1]^T$ increasing the marking of place p_2: hence in the CG we have an ω-increasing production $\pi : [1\ 0]^T t_1 [0\ 2]^T t_2 [1\ 1]^T$ that from $s(\pi) = [1\ 0]^T$ leads to node $e(\pi) = [1\ \omega]^T$. It is easy to verify that the ω-increasing production associated with $t_1 t_2$ does not satisfy conditions of Proposition 11.4. ■

For the sake of simplicity we want to consider Petri nets whose increasing sequences can be easily recognized looking at the CG using Proposition 11.4. Thus, in the rest of this chapter the following assumption holds.

(A1) We want to identify Petri nets for which Proposition 11.4 gives necessary and sufficient conditions.

Let us introduce the following notation.

[1] This result is formally proved by Proposition 11.10.



(a) (b)

Figure 11.3: Net in Example 11.5.

- $\mathcal{O}$ is the set of all ω-increasing markings,
- Π is the set of ω-increasing productions,
- Π_k is the set of ω-increasing productions that end in q_k.

Example 11.6. From the graph in Figure 11.1.(d) we recognize the following nodes as associated to ω-increasing markings:

- q_2. With the notation used in the proof of the previous proposition, $\bar{q} = q_0$ and the corresponding ω-increasing production is $\pi = q_0 t_1 q_1 t_2 q_2$.
- q_4 (resp., q_5). Here $\bar{q} = q_2$ (resp., $\bar{q} = q_3$) and the two corresponding ω-increasing productions are $\pi = q_2 t_3 q_4$ or $\pi' = q_2 t_4 q_4$ (resp., $\pi = q_3 t_3 q_5$ or $\pi' = q_3 t_4 q_5$).
- q_6. We have two choices for $\bar{q}$: $\bar{q} = q_4$ and $\bar{q} = q_5$. The corresponding ω-productions are $\pi = q_4 t_2 q_6$ and $\pi' = q_5 t_2 q_6$.

■

Let us introduce some equivalence classes that will be useful in the following of this chapter.

Definition 11.7 (Set of ω components)**.** *Given a node q of a CG we define*

$$\Omega(q) = \{p \in P \mid q \text{ is labeled with } M, M(p) = \omega\}$$

the set of places associated to ω components in node q. ■

The set of ω components induces two relations on the nodes of a CG.

Definition 11.8. *Given two nodes q and q' we say that:*

- $q \equiv q'$ *if* $\Omega(q) = \Omega(q')$.
- $q \preceq q'$ *(resp., $q \prec q'$) if* $\Omega(q) \subseteq \Omega(q')$ *(resp., $\Omega(q) \subsetneq \Omega(q')$).*

■

One can immediately verify that the first one is an equivalence relation while the second one is a partial order relation. It is thus possible to partition the set of nodes according to the equivalence classes of $\equiv$ and to order them according to $\prec$.

Example 11.9. In the graph in Figure 11.1.(c), we recognize the following equivalence classes:

$$\mathcal{Q}_0 = \{q_0, q_1\}, \quad \mathcal{Q}_1 = \{q_2, q_3\}, \quad \mathcal{Q}_2 = \{q_4, q_5\}, \quad \mathcal{Q}_3 = \{q_6\}.$$

These classes are ordered as shown in the following Hesse diagram

$$\mathcal{Q}_0 \longrightarrow \mathcal{Q}_1 \longrightarrow \mathcal{Q}_2 \longrightarrow \mathcal{Q}_3$$

where an arc from $\mathcal{Q}_i$ to $\mathcal{Q}_j$ denotes that $\mathcal{Q}_i \prec \mathcal{Q}_j$. In this particular case, the classes are also completely ordered. ■

Here we consider the problem of determining, given a graph $\mathcal{G}$ whose edges are labeled with elements of T, a net $\langle N, M_0 \rangle$ with set of transitions T, for which assumption A1 holds and such that its CG is isomorphic to $\mathcal{G}$. Note that the input of our procedure is not a graph such as the one in Figure 11.1.(c) where each node is labeled by a marking, but an unlabeled graph such as the one in Figure 11.1.(d), where no information is given on the places.

Our identification procedure requires at first to partition the nodes of the graph into equivalence classes for the $\equiv$ relation (we call this partition $\mathcal{Q}$) and to order them. We first observe that although the unlabeled graph does not contain enough information to exactly reconstruct such a partition, it allows one to determine a partition $\hat{\mathcal{Q}}$ that refines[2] $\mathcal{Q}$.

Here we prove that this is possible just looking at the unlabeled graph — such as the one in Figure 11.1.(d).

We start with the following elementary observation.

Proposition 11.10. *Let us consider two nodes q_i and q_j such that*

$$q_i \xrightarrow{t} q_j$$

then the following results hold:

(i) $q_i \preceq q_j$,

(ii) $q_i \prec q_j$ *iff* $\exists\, \pi \in \Pi_j$ *and* $l(\pi) = q_i$.

Proof. (i) Assume nodes q_i and q_j are labeled, respectively, M and M'. In the construction of the CT whenever a marking M is such that $M(p) = \omega$, then the firing of an enabled transition t from M leads to $M' = M + C(\cdot, t)$, hence $M'(p) = M(p) + C(p,t) = \omega + C(p,t) = \omega$.

(ii) Follows from the fact that an ω is introduced in the graph only by an ω-increasing production. □

[2] Partition $\hat{\mathcal{Q}}$ refines partition $\mathcal{Q}$ iff for all q it holds $\hat{\mathcal{Q}}[q] \subseteq \mathcal{Q}[q]$, where $\mathcal{Q}[q]$ denotes the class that contains q.

This means that along any path of a CG the nodes that one encounters are ordered with respect to (wrt) $\preceq$. Moreover, the number of ω-components only increases when reaching an ω-increasing marking from an ω-increasing sequence.

Example 11.11. Let us consider the graph in Figure 11.1.(c) whose node labeling is shown in Figure 11.1.(d). It is easy to observe that $q_0 \equiv q_1$ because both of them contain no ω- component. We can also say that $q_1 \prec q_2$, in fact q_1 contains no ω-component while q_2 contains one. In the same way we can say that $q_2 \equiv q_3$, $q_2 \prec q_4$, $q_3 \prec q_5$, $q_4 \equiv q_5$ and $q_4, q_5 \prec q_6$. ■

We can finally state the procedure to partition an unlabeled CG in equivalence classes. In the following algorithm we will say that given two subsets of nodes $\hat{\mathcal{Q}}_i$ and $\hat{\mathcal{Q}}_j$ (with $i \neq j$) the following predicate holds:

- $c(i,j)$: **if** there exist two nodes $q' \in \hat{\mathcal{Q}}_i$ and $q'' \in \hat{\mathcal{Q}}_j$ such that $\delta(q',t) = q''$ for some transition $t \in T$ **and** it does not exist an ω-increasing production $\pi \in \Pi$ such that $l(\pi) = q'$ and $e(\pi) = q''$.

Algorithm 11.12. ***Partition of an unlabeled coverability tree***

1. *Consider an initial partition of the graph in strongly connected components,*

$$\hat{\mathcal{Q}}_0 \cup \hat{\mathcal{Q}}_1 \cup \cdots \cup \hat{\mathcal{Q}}_k$$

   where $\hat{\mathcal{Q}}_0$ is the component containing the initial node q_0 and $k+1$ is the number of such components.

2. ***While** there exist $\hat{\mathcal{Q}}_i$ and $\hat{\mathcal{Q}}_j$ such that $c(i,j)$ **do** merge $\hat{\mathcal{Q}}_i$ and $\hat{\mathcal{Q}}_j$.*

3. *The final partition is*

$$\hat{\mathcal{Q}}_0 \cup \hat{\mathcal{Q}}_1 \cup \cdots \cup \hat{\mathcal{Q}}_r, \qquad r \leq k.$$

■

Example 11.13. Consider the graph in Figure 11.1.(d). The initial partition is $\hat{\mathcal{Q}}_0 = \{q_0\}$, $\hat{\mathcal{Q}}_1 = \{q_1\}$, $\hat{\mathcal{Q}}_2 = \{q_2, q_3\}$, $\hat{\mathcal{Q}}_3 = \{q_4\}$, $\hat{\mathcal{Q}}_4 = \{q_5\}$ and $\hat{\mathcal{Q}}_5 = \{q_6\}$.

At step 2 of Algorithm 11.12 we merge $\hat{\mathcal{Q}}_0$ with $\hat{\mathcal{Q}}_1$ and we also merge $\hat{\mathcal{Q}}_3$ with $\hat{\mathcal{Q}}_4$. The resulting final partition is $\hat{\mathcal{Q}}_0 = \{q_0, q_1\}, \hat{\mathcal{Q}}_1 = \{q_2, q_3\}, \hat{\mathcal{Q}}_2 = \{q_4, q_5\}, \hat{\mathcal{Q}}_3 = \{q_6\}$ that coincides with the partition $\mathcal{Q}$ in equivalence classes discussed in Example 11.9. ■

In the previous example the algorithm determines the exact partition $\mathcal{Q}$ in equivalence classes. In general the following result holds.

Proposition 11.14. *The partition $\hat{\mathcal{Q}}$ determined by Algorithm 11.12 is a refinement of the partition $\mathcal{Q}$ in equivalence classes for the $\equiv$ relation.*

Proof. We first note that the initial partition refines $\mathcal{Q}$. In fact, according to Proposition 11.10.i if two nodes q_k and q_j belong to the same strongly connected component, then $q_k \preceq q_j$ and $q_j \preceq q_k$, hence $q_k \equiv q_j$.

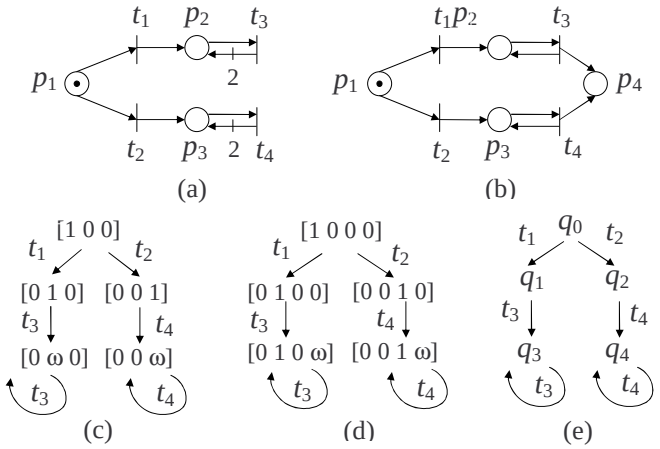


Figure 11.4: Nets in Example 11.15.

Secondly, we observe that the classes that are merged at step 2 of the algorithm belong to the same equivalence class being joined by a transition that is not the terminal path of an ω-increasing production, hence no component is changed to ω according to Proposition 11.10.ii.

□

Example 11.15. Let us consider the nets in Figures 11.4.(a) and (b). The CG of these nets are given in (c) and (d), respectively, while the unlabeled graph is reported in (e) and is the same in the two cases. In the case of the net in Figure 11.4.(a) the equivalence classes are: $\mathcal{Q}_0 = \{q_0, q_1, q_2\}$, $\mathcal{Q}_1 = \{q_3\}$, $\mathcal{Q}_2 = \{q_4\}$, while in the case of the net in Figure 11.4.(b) the equivalence classes are: $\mathcal{Q}_0 = \{q_0, q_1, q_2\}$, $\mathcal{Q}_1 = \{q_3, q_4\}$.

Thus in this case we cannot exactly reconstruct the equivalence classes by simply looking at the unlabeled graph. In particular, our algorithm always finds the final partition of the CG of the net in Figure 11.4.(a), that is a refinement of the CG of the net in Figure 11.4.(b). ■

We finally introduce the notion of ω-stationary sequence.

Definition 11.16. *A sequence σ is* ω-stationary *wrt a subset of places $P' \subseteq P$ if $x = C \cdot \pi(\sigma)$ is such that $x(p) = 0$, for all $p \in P'$.* ■

In simple words, a sequence σ is ω-stationary with respect to $P' \subseteq P$ if its firing does not produce a variation in the marking in places P'.

Proposition 11.17. In a CG $\mathcal{G}$ of a Petri net, a production π with $s(\pi) = q$ corresponds to a sequence ω-stationary wrt $\Omega(q)$ iff $e(\pi) = q$.

Proof. Follows from the previous definition, because a sequence σ is ω-stationary wrt $\Omega(q)$ if and only if it does not modify the token content of places that are not associated to ω components, i.e., if and only if $e(\pi) = q$. □

A production π such that $\sigma = \ell(\pi)$ is a stationary sequence, is an ω-stationary production. However, there may also exist ω-stationary productions that do not correspond to stationary sequences.

Example 11.18. Let us consider the net in Figure 11.1. Here $\sigma = t_3$ is an ω-stationary sequence wrt $\Omega(q_4) = \Omega(q_5) = \{p_2, p_3\}$ and wrt $\Omega(q_6) = \{p_1, p_2, p_3\}$. The firing of t_3 from any of these nodes corresponds to a cycle. ■

11.2 Synthesis of a PN system from its unlabeled coverability graph

Problem 11.19. *Let $\mathscr{G} = (Q, T, \delta, q_0)$ be a given finite state automaton. Choosing a set of places P of cardinality m, we want to identify the structure of a free-labeled Petri net $N = (P, T, Pre, Post)$ and an initial marking M_0 such that assumption A1 is satisfied and the CG of $\langle N, M_0\rangle$ is isomorphic to $\mathscr{G}$.*

The unknowns we want to determine are the elements of the two matrices Pre, $Post \in \mathbb{N}^{m\times n}$ and the elements of the vector $M_0 \in \mathbb{N}^m$. ■

In this section we provide a set of linear algebraic constraints and we prove that a net system $\langle N, M_0\rangle$ satisfying assumption A1 is a solution of Problem 11.19 if and only if it satisfies the given set of constraints. The proof will be a sketch, in the sense that we explain the meaning of each type of constraints.

In the previous section we have characterized the information on the net that can be extracted from the CG in terms of its language and of ω-increasing and ω-stationary sequences. However, to ensure that the synthesized net has a CG isomorphic to the given one, it is also necessary to impose two additional types of constraints.

The first type of constraints requires that if in the graph two sequences σ_k and σ'_k lead to the same node q_k, then for all places $p \notin \Omega(q_k)$ it should hold

$$M_0(p) + C(p,\cdot)\cdot y_k = M_0(p) + C(p,\cdot)\cdot y'_k, \tag{11.2}$$

where $y_k = \pi(\sigma_k)$ and $y'_k = \pi(\sigma'_k)$.

To do this we introduce the following definition.

Definition 11.20. Given a node $q_k \in Q$ we denote π_k a minimal[3] production starting from q_0 and ending in q_k. We also denote $\sigma_k = \ell(\pi_k)$ the associated sequence and y_k the corresponding firing vector, that will be used to represent the marking $M_k = M_0 + C\cdot y_k$ associated to node q_k. ■

Sequence σ_k will be used to identify node q_k while other sequences σ'_k yielding the same marking will have to satisfy (11.2).

[3]By minimal we mean that the production does not contain twice the same node. More than one such production may exist: we arbitrarily choose one.

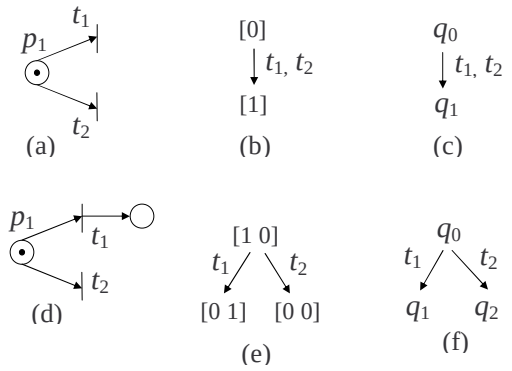


Figure 11.5: The resulting net in Example 11.22.

The second condition is the dual of the previous one. Assume that the graph contains two nodes q_k and q_j that are equivalent in the sense that all productions that start from them cannot be distinguished neither in terms of language nor in terms of ω-increasing nor of ω-stationary sequences. To make sure that in the graph of the synthesized net these two nodes are not collapsed into a single one, we need to specify that either the two nodes belong to two different classes, i.e., $\Omega(q_k) \neq \Omega(q_j)$, or they differ for at least a component different from ω, i.e., there exists $p \notin \Omega(q_k) \cup \Omega(q_j)$ such that

$$M_0(p) + C(p,\cdot) \cdot y_k \neq M_0(p) + C(p,\cdot) \cdot y_j. \tag{11.3}$$

The nodes that must be distinguished are the nodes that satisfy the following notion of bisimilarity.

Definition 11.21. Given a finite state automaton $\mathcal{G} = (Q, T, \delta, q_0)$. Let $q,\ q' \in Q$. We say that q is *simulated* by q' if the following conditions hold.

- *(Language equivalence.)* $\delta(q,\sigma)$ is defined $\Rightarrow \delta(q',\sigma)$ is defined, for any $\sigma \in T^*$.
- *(ω-increasing equivalence.)* If π_1 is a production and π_2 is an ω-increasing production such that $s(\pi_1) = q$ and $e(\pi_1) = s(\pi_2) \Rightarrow$ there exists a production π'_1 and an ω-increasing production π'_2 such that $s(\pi'_1) = q'$, $e(\pi'_1) = s(\pi'_2)$, $\ell(\pi_1) = \ell(\pi'_1)$ and $\ell(\pi_2) = \ell(\pi'_2)$ (where $s(\pi), e(\pi)$ and $l(\pi)$ are defined in Definition 11.2).
- *(ω-stationary equivalence.)* If π_1 is a production and π_2 is an ω-stationary production such that $s(\pi_1) = q$ and $e(\pi_1) = s(\pi_2) \Rightarrow$ there exists a production π'_1 and an ω-stationary production π'_2 such that $s(\pi'_1) = q'$, $e(\pi'_1) = s(\pi'_2)$, $\ell(\pi_1) = \ell(\pi'_1)$ and $\ell(\pi_2) = \ell(\pi'_2)$ (where $s(\pi), e(\pi)$ and $l(\pi)$ are defined in Definition 11.2).

We say that $q,\ q' \in Q$ are *bisimilar* if q is simulated by q' and q' is simulated by q. ■

Example 11.22. Let us consider the net system in Figure 11.5.(d) whose CG is shown in (e), while the unlabeled graph is reported in (f). It is immediate to verify that nodes q_1 and q_2 are bisimilar. ■

We finally introduce the notation to describe enabling and disabling of transitions.

- $\mathscr{E} = \{(q,t) \in Q \times T \mid \delta(q,t) \text{ is defined}\}$ is the set of couples (state q – transition t) such that t is enabled at the state q of $\mathscr{G}$.
- $\mathscr{D} = \{(q,t) \in Q \times T \mid \delta(q,t) \text{ is not defined}\}$ is the set of couples (state q – transition t) such that t is *not* enabled at the state q of $\mathscr{G}$.

The following theorem characterizes the set of solutions to Problem 11.19.

Theorem 11.23. *A net system $\langle N, M_0\rangle$ is a solution of the identification problem 11.19 if and only if it satisfies the following linear algebraic constraints.*

(a) Enabling constraints: $\forall\ (q_k, t_j) \in \mathscr{E}$ *let*

$$M_0 + C \cdot y_k + K \cdot \vec{s}_{c(k)} \geq Pre \cdot \vec{t}_j$$

where y_k is chosen as in Definition 11.20 and K is a very large constant (K must be larger than the maximum expected weight of pre and post arcs).

(b) Constraints related to ω-increasing sequences: $\forall\ q_k \in \mathscr{C}$ *and* $\forall\ \pi \in \Pi_k$ *let*

$$\begin{cases} C \cdot y - \vec{s}_{c(k)} + K \cdot \vec{s}_{c(i)} \geq \vec{0} & (b1) \\ -C \cdot y + K \cdot \vec{s}_{c(k)} \geq \vec{0} & (b2) \\ \vec{s}_{c(k)} \geq \vec{s}_{c(i)} & (b3) \\ \vec{1}^T \cdot \vec{s}_{c(k)} > \vec{1}^T \cdot \vec{s}_{c(i)} & (b4) \end{cases}$$

where y is the firing vector associated to the generic production π and $q_i = l(\pi)$.

(c) Constraints related to ω-stationary sequences: $\forall i = 0, 1, \ldots, r$ *and* $\forall y \in \mathscr{S}_i = \{y \in \mathbb{N}^n \mid \exists \pi : \nu(\pi) \subseteq \hat{\mathscr{Q}}_i, s(\pi) = e(\pi), \ell(\pi) = \sigma\}$, *where $\mathscr{S}_i$ is the set of firing vectors corresponding to cycles in component $\hat{\mathscr{Q}}_i$, let*

$$\begin{cases} C \cdot y + K \cdot \vec{s}_i \geq \vec{0} & (c1) \\ -C \cdot y + K \cdot \vec{s}_i \geq \vec{0} & (c2) \end{cases}$$

(d) Blocking constraints: $\forall\ (q_k, t_j) \in \mathscr{D}$ *let*

$$\begin{cases} M_0 + C \cdot y_k - K \cdot \vec{s}_{k,j} < Pre \cdot \vec{t}_j & (d1) \\ \vec{1}^T \cdot \vec{s}_{k,j} \leq m - 1 & (d2) \\ \vec{s}_{k,j} \geq \vec{s}_{c(k)} & (d3) \end{cases}$$

(e) Equivalence constraints: *Assume that the minimal production reaching q_k, as in Definition 11.20, is $\pi_k = q_0 \longrightarrow \cdots \longrightarrow q_{r(k)} \xrightarrow{t_{r(k)}} q_k$. Then we define $\mathscr{I}(q_k) = \{(q,t) \mid q \xrightarrow{t} q_k \wedge (q,t) \neq (q_{r(k)}, t_{r(k)})\}$. In other words $\mathscr{I}(q_k)$ is the set of couples (state q - transition t) that lead to q_k except the couple $(q_{r(k)}, t_{r(k)})$ that has already been considered in constraints (a).*

$\forall (q_i, t) \in \mathscr{I}(q_k)$ *let*

$$\begin{cases} C \cdot y_k - C \cdot y + K \cdot \vec{s}_{c(k)} \geq \vec{0} & (e1) \\ -C \cdot y_k + C \cdot y + K \cdot \vec{s}_{c(k)} \geq \vec{0} & (e2) \end{cases}$$

where $\sigma = \sigma_i t$, and σ_i is the sequence associated to node q_i, as in Definition 11.20.

(f) Discriminating constraints: *for all bisimilar nodes q_k, q_j that belong to the same equivalence class $\hat{\mathscr{Q}}_i$, let*

$$\begin{cases} C \cdot y_k - C \cdot y_j + K \cdot \vec{l}_{j,k} \geq \vec{1} & (f1) \\ C \cdot y_k - C \cdot y_j - K \cdot \vec{l}_{k,j} \leq -\vec{1} & (f2) \\ \vec{1}^T \cdot \left(\vec{l}_{j,k} + \vec{l}_{k,j} \right) \leq 2m - 1 & (f3) \\ \vec{l}_{j,k},\ \vec{l}_{k,j} \geq \vec{s}_i & (f4) \\ \vec{l}_{k,j},\ \vec{l}_{j,k} \in \{0,1\}^m & (f5) \end{cases}$$

(g) Discriminating constraints: *for all bisimilar nodes q_k, q_j that belong to equivalence classes among which an ordering does not exist, let*

$$\begin{cases} \vec{s}_{c(k)} - \vec{s}_{c(j)} + K \cdot \vec{v}_{j,k} + K \cdot z_1 \cdot \vec{1} \geq \vec{1} & (g1) \\ \vec{s}_{c(k)} - \vec{s}_{c(j)} - K \cdot \vec{v}_{k,j} + K \cdot z_1 \cdot \vec{1} \leq -\vec{1} & (g2) \\ C \cdot y_k - C \cdot y_j + K \cdot \vec{l}_{j,k} + K \cdot z_2 \cdot \vec{1} \geq \vec{1} & (g3) \\ C \cdot y_k - C \cdot y_j - K \cdot \vec{l}_{k,j} - K \cdot z_2 \cdot \vec{1} \leq -\vec{1} & (g4) \\ \vec{1}^T \cdot \left(\vec{v}_{j,k} + \vec{v}_{k,j} \right) \leq 2m - 1 & (g5) \\ \vec{1}^T \cdot \left(\vec{l}_{j,k} + \vec{l}_{k,j} \right) \leq 2m - 1 & (g6) \\ \vec{l}_{j,k},\ \vec{l}_{k,j} \geq \vec{s}_{c(k)} & (g7) \\ \vec{l}_{j,k},\ \vec{l}_{k,j} \geq \vec{s}_{c(j)} & (g8) \\ \vec{l}_{k,j},\ \vec{l}_{j,k}, \vec{v}_{k,j},\ \vec{v}_{j,k} \in \{0,1\}^m & (g9) \\ z_1 + z_2 \leq 1 & (g10) \\ z_1,\ z_2 \in \{0,1\} & (g11) \end{cases}$$

(h) Integrity constraints.

$$\begin{cases} M_0 \in \mathbb{N}^m & (h1) \\ C = Post - Pre & (h2) \\ Pre, Post \in \mathbb{N}^{m \times n} & (h3) \\ \vec{s}_i \in \{0,1\}^m,\ \forall\ \hat{\mathscr{Q}}_i & (h4) \\ \vec{s}_{k,j} \in \{0,1\}^m,\ \forall\ (q_k, t_j) \in \mathscr{D} & (h5) \end{cases}$$

In the following we denote as $\mathscr{C}(\mathscr{G}, P)$ the set of constraints (a) to (h) associated to the unlabeled graph $\mathscr{G}$ and to the set of places P.

Proof.

The proof consists in a detailed explanation of all constraints.

• *Constraints (a).* To each equivalence class $\hat{\mathscr{Q}}_i$ we associate a vector $\vec{s}_i \in \{0,1\}^m$ such that $s_i(p) = 1 \quad \Leftrightarrow \quad \forall\ q \in \hat{\mathscr{Q}}_i,\ p \in \Omega(q)$.

If $(q_k, t_j) \in \mathscr{E}$, then the marking $M_k = M_0 + C \cdot y_k$ is such that $M_k(p) \geq Pre(p,t)$ for all p such that $s_{c(k)}(p) = 0$. On the contrary, if $s_{c(k)}(p) = 1$ regardless of the value of $M_k(p)$, t_j is enabled.

• *Constraints (b).* Let q_k be any node in $\mathscr{O}$, π a production in Π_k, and $q_i = l(\pi)$.

We first observe that if there exists a place $p \in \Omega(q_i)$, then by Proposition 11.4 $p \in \Omega(q)$ for any $q \in \hat{\mathscr{Q}}_{c(k)}$. Thus, $\vec{s}_{c(i)} \geq \vec{s}_{c(k)}$.

Therefore, if $s_{c(i)}(p) = 1$ then $s_{c(k)}(p) = 1$, and constraints (b1) and (b2) are trivially verified.

If $s_{c(i)}(p) = 0$ it may either be $s_{c(k)}(p) = 0$ or $s_{c(k)}(p) = 1$. In the first case the firing of the sequence associated to π does not increase the token contents of p as imposed by constraints (b1) and (b2). In the second case, it must hold $C(p,\cdot) \cdot y > 0$ that is equivalent to impose constraint (b1), while (b2) is trivially verified.

• *Constraints (c).* Let π be a production whose characteristic vector is in $\mathscr{S}_i$, namely a production relative to an ω-stationary sequence for $\hat{\mathscr{Q}}_i$.

By definition π should not change the content of all places $p \notin \Omega(q)$ for any $q \in \hat{\mathscr{Q}}_i$, while no constraint should be imposed on the other places $p \in \Omega(q)$. This is actually the meaning of constraints (c). In fact, if $s_i(p) = 0$, then $C(p,\cdot) \cdot y = 0$; if $s_i(p) = 1$, then constraints (c) are trivially verified.

• *Constraints (d).* If transition t_j is not enabled at $M_k = M_0 + C \cdot y_k$, then for at least one place p it must hold $M_k(p) < Pre \cdot \vec{t}_j$.

We now define a vector $\vec{s}_{k,j} \in \{0,1\}^m$ such that $s_{k,j}(p) = 0 \quad \Leftrightarrow \quad M_k(p) < Pre \cdot \vec{t}_j$.

Assume that each component of M_k is less than or equal to K. Then the component of $\vec{s}_{k,j}$ relative to the generic place p must satisfy the equation

$$M_k(p) - K \cdot s_{k,j}(p) < Pre \cdot \vec{t}_j, \tag{11.4}$$

so that if $s_{k,j}(p) = 0$ it must hold $M_k(p) < Pre \cdot \vec{t}_j$, while if $s_{k,j}(p) = 1$, equation (11.4) is trivially verified. In vector form (and taking into account that all variables are integers) this equation rewrites as (d1).

Note that there exists at least one place that disables t_j if $\vec{1}^T \cdot \vec{s}_{k,j} \leq m - 1$ so that at least one $s_{k,j}(p)$ is null.

Finally, the constraint $\vec{s}_{k,j} \geq \vec{s}_{c(k)}$ imposes that, if $s_{c(k)}(p) = 1$ then $s_{k,j}(p) = 1$. That is to say, t_j cannot be disabled by a place $p \in \Omega(q_k)$.

• *Constraints (e).* Assume that there exists a production π that, as π_k, reaches node q_k from q_0. Assume also that $\pi = q_0 \longrightarrow \ldots \longrightarrow q_i \xrightarrow{t} q_k$ with $(q_i, t) \in \mathscr{F}(q_k)$. Then for all places $p \notin \Omega(q_k)$ it should be $C(p,\cdot) \cdot y_k = C(p,\cdot) \cdot y$, while for the other places no relationship can be deduced from the CG. This is exactly the meaning of constraints (e1) and (e2). In fact, if $s_{c(k)}(p) = 0$, then $C(p,\cdot) \cdot y_k = C(p,\cdot) \cdot y$; otherwise we get two constraints that are trivially verified.

• *Constraints (f).* Let q_k and q_j be two nodes that are bisimilar and that belong to the same equivalence class $\hat{\mathscr{Q}}_i$. To distinguish between these nodes we impose that there exists at least one place p, to which it does not correspond ω in $\hat{\mathscr{Q}}_i$, such that $C(p,\cdot) \cdot y_k \neq C(p,\cdot) \cdot y_j$.

We define two vectors $\vec{l}_{k,j}$, $\vec{l}_{j,k} \in \{0,1\}^m$ such that $\vec{l}_{k,j}$, $\vec{l}_{j,k} \geq \vec{s}_i$.

If $l_{j,k}(p) = l_{k,j}(p) = 1$ constraints (f1) and (f2) are trivially verified, and this occurs for all places $p \in \Omega(q_k) = \Omega(q_j)$ being $\vec{l}_{j,k},\ \vec{l}_{k,j} \geq \vec{s}_i$. If $l_{j,k}(p) = 1$ and $l_{k,j}(p) = 0$ then $C(p,\cdot) \cdot y_k \leq C(p,\cdot) \cdot y_j - 1$. If $l_{k,j}(p) = 1$ and $l_{j,k}(p) = 0$ then $C(p,\cdot) \cdot y_k \geq C(p,\cdot) \cdot y_j + 1$.

Note that one of the above two cases always occur being by (f3), $\vec{1}^T \cdot \left(\vec{l}_{j,k} + \vec{l}_{k,j}\right) \leq 2m - 1$.

• *Constraints (g).* Let q_k and q_j be two nodes that are bisimilar and that belong to equivalence classes among which an ordering does not exist. To distinguish between these nodes we impose that at least one of the following conditions hold: (I) q_k and q_j differ in a place not containing ω, (II) $\Omega(q_k) \neq \Omega(q_j)$, i.e., $\vec{s}_{c(k)} \neq \vec{s}_{c(j)}$.

Now, being $z_1 + z_2 \leq 1$, with $z_1,\ z_2 \in \{0,1\}$, three different cases may occur: $z_1 = 1$ and $z_2 = 0$, $z_1 = 0$ and $z_2 = 1$, $z_1 = z_2 = 0$.

— Assume that $z_1 = 1$ and $z_2 = 0$. In such a case constraints (g1) and (g2) are trivially verified, and the only significant constraints are (g3), (g4), (g6) to (g8) that are analogous to constraints (f): they impose that q_k and q_j differ in a place not containing ω (case (I) above).

— Assume that $z_1 = 0$ and $z_2 = 1$. In such a case constraints (g3) and (g4) are trivially verified, and the only significant constraints are (g1), (g2) and (g5). Using the same reasoning as above, it is easy to verify that these constraints impose that $s_{c(k)}(p) \neq s_{c(j)}(p)$ for at least one place $p \in P$ (case (II) above).

— Assume that $z_1 = z_2 = 0$. In such a case no constraint in (g) is trivial, thus q_k and q_j have ω in different places, and they also differ in some place not containing ω. □

In general the solution of $\mathscr{C}(\mathscr{G}, P)$ is not unique, thus there exists more than one Petri net system whose CG is isomorphic to $\mathscr{G}$. To select one among these Petri net systems we choose a given performance index and solving an appropriate IPP we determine a Petri net system that minimizes the considered performance index[4]. In particular, if $f(M_0, Pre, Post)$ is the considered performance index, an identification problem can be formally stated as follows.

Problem 11.24. *Let us consider the identification problem 11.19 and let $f(M_0, Pre, Post)$ be a given performance index. The solution to the identification problem 11.19 that minimizes $f(M_0, Pre, Post)$ can be computed by solving the following IPP*

$$\begin{cases} \min & f(M_0, Pre, Post) \\ s.t. & \mathscr{C}(\mathscr{G}, P). \end{cases} \tag{11.5}$$

■

Of particular interest are those objective functions that are linear in the unknowns, so that the problem to solve is a linear integer programming problem. As an example of a linear objective function, assume we want to determine a Petri net system that minimizes the weighted sum of the tokens in the initial marking and of the arc weights. The general case is:

$$f(M_0, Pre, Post) = \sum_{i=1}^{m} \left(a_i \cdot M_0(p_i) + \left(\sum_{j=1}^{n} b_{i,j} \cdot Pre(p_i, t_j) + c_{i,j} \cdot Post(p_i, t_j) \right) \right), \tag{11.6}$$

[4] Clearly, also in this case the solution may be not unique.

where $a_i, b_{i,j}$ and $c_{i,j}$ are given coefficients.

A typical choice, that we follow in the rest of the paper, is that of choosing all coefficients equal to 1. In this case (11.6) can be rewritten:

$$f(M_0, Pre, Post) = \vec{1}^T \cdot M_0 + \vec{1}^T \cdot (Pre + Post) \cdot \vec{1}.$$

Example 11.25. Let us consider the unlabeled graph $\mathscr{G}$ in Figure 11.1.(d). We want to determine a net system $\langle N, M_0 \rangle$ with $N = (P, T, Pre, Post)$ and $m = 3$ satisfying assumption A1 and whose reachability graph is isomorphic to $\mathscr{G}$. In particular, we want to minimize the tokens in the initial marking and the arc weights.

The set of ω-increasing nodes is $\mathscr{O} = \{q_2, q_3, q_4, q_5, q_6\}$. The set of ω-increasing productions that ends in q_2 is $\Pi_2 = \{\pi_2\}$, with $\ell(\pi_2) = t_1 t_2$. Then $\Pi_4 = \{\pi'_4, \pi''_4\}$, with $\ell(\pi'_4) = t_3$ and $\ell(\pi''_4) = t_4$; $\Pi_5 = \{\pi'_5, \pi''_5\}$, with $\ell(\pi'_5) = t_3$ and $\ell(\pi''_5) = t_4$; $\Pi_6 = \{\pi_6\}$ with $\ell(\pi_6) = t_2$.

We only have ω-stationary productions associated to the equivalence class $\mathscr{Q}_3 = \{q_6\}$ and the set of firing vectors is $\mathscr{S}_6 = \{y'_6, y''_6, y'''_6\}$ where $\sigma'_6 = t_1, \sigma''_6 = t_3, \sigma'''_6 = t_4$. Then,

$$\begin{aligned}\mathscr{E} = \{ &(q_0, t_1), (q_1, t_2), (q_2, t_1), (q_2, t_3), (q_2, t_4), (q_3, t_2), (q_3, t_3), (q_3, t_4), (q_4, t_1), (q_4, t_2),\\ &(q_4, t_3), (q_4, t_4), (q_5, t_2), (q_5, t_3), (q_5, t_4), (q_6, t_1), (q_6, t_2), (q_6, t_3), (q_6, t_4) \}\end{aligned}$$

$$\mathscr{D} = \{ (q_0, t_2), (q_0, t_3), (q_0, t_4), (q_1, t_1), (q_1, t_3), (q_1, t_4), (q_2, t_2), (q_3, t_1), (q_5, t_1) \}.$$

Moreover, the set of firing vectors associated to minimal sequences that enable to reach the different nodes of the graph are:

$$\begin{aligned}&\Sigma_1 = \{\vec{t}_1\}, \quad \Sigma_2 = \{\vec{t}_1 + \vec{t}_2\}, \quad \Sigma_3 = \{2\vec{t}_1 + \vec{t}_2\},\\ &\Sigma_4 = \{\vec{t}_1 + \vec{t}_2 + \vec{t}_3,\ \vec{t}_1 + \vec{t}_2 + \vec{t}_4\},\\ &\Sigma_5 = \{2\vec{t}_1 + \vec{t}_2 + \vec{t}_3,\ 2\vec{t}_1 + \vec{t}_2 + \vec{t}_4\},\\ &\Sigma_6 = \{\vec{t}_1 + 2\vec{t}_2 + \vec{t}_3,\ \vec{t}_1 + 2\vec{t}_2 + \vec{t}_4\}.\end{aligned}$$

Finally we observe that there are no bisimilar nodes, thus we have no constraints of the form (f) and (g).

Let us show now the set of constraints $\mathscr{C}(\mathscr{G}, P)$.

Let us preliminary observe that being

$$\hat{\mathscr{Q}}_0 = \{q_0, q_1\}, \hat{\mathscr{Q}}_1 = \{q_2, q_3\}, \hat{\mathscr{Q}}_2 = \{q_4, q_5\}, \hat{\mathscr{Q}}_3 = \{q_6\},$$

it holds $\vec{s}_{c(0)} = \vec{s}_{c(1)} = \vec{0}$, and

$$\vec{s}_{c(2)} = \vec{s}_{c(3)} = \vec{s}_1, \quad \vec{s}_{c(4)} = \vec{s}_{c(5)} = \vec{s}_2, \quad \vec{s}_{c(6)} = \vec{s}_3.$$

(a) Enabling constraints

$$
\begin{cases}
(q_0, t_1),\ q_0 \notin \mathcal{O} & : & M_0 \geq Pre \cdot \vec{t}_1 \\
(q_1, t_2),\ q_1 \notin \mathcal{O} & : & M_0 + C \cdot \vec{t}_1 \geq Pre \cdot \vec{t}_2 \\
(q_2, t_1),\ q_2 \in \mathcal{O} & : & M_0 + C \cdot (\vec{t}_1 + \vec{t}_2) + K \cdot \vec{s}_1 \geq Pre \cdot \vec{t}_1 \\
(q_2, t_4) & : & M_0 + C \cdot (\vec{t}_1 + \vec{t}_2) + K \cdot \vec{s}_1 \geq Pre \cdot \vec{t}_4 \\
(q_2, t_3) & : & M_0 + C \cdot (\vec{t}_1 + \vec{t}_2) + K \cdot \vec{s}_1 \geq Pre \cdot \vec{t}_3 \\
(q_3, t_2),\ q_3 \in \mathcal{O} & : & M_0 + C \cdot (2\,\vec{t}_1 + \vec{t}_2) + K \cdot \vec{s}_1 \geq Pre \cdot \vec{t}_2 \\
(q_3, t_3) & : & M_0 + C \cdot (2\,\vec{t}_1 + \vec{t}_2) + K \cdot \vec{s}_1 \geq Pre \cdot \vec{t}_3 \\
(q_3, t_4) & : & M_0 + C \cdot (2\,\vec{t}_1 + \vec{t}_2) + K \cdot \vec{s}_1 \geq Pre \cdot \vec{t}_4 \\
(q_4, t_1),\ q_4 \in \mathcal{O} & : & M_0 + C \cdot (\vec{t}_1 + \vec{t}_2 + \vec{t}_3) + K \cdot \vec{s}_2 \geq Pre \cdot \vec{t}_1 \\
(q_4, t_2) & : & M_0 + C \cdot (\vec{t}_1 + \vec{t}_2 + \vec{t}_3) + K \cdot \vec{s}_2 \geq Pre \cdot \vec{t}_2 \\
(q_5, t_2),\ q_5 \in \mathcal{O} & : & M_0 + C \cdot (2\,\vec{t}_1 + \vec{t}_2 + \vec{t}_3) + K \cdot \vec{s}_2 \geq Pre \cdot \vec{t}_2 \\
(q_6, t_1),\ q_6 \in \mathcal{O} & : & M_0 + C \cdot (\vec{t}_1 + 2\,\vec{t}_2 + \vec{t}_3) + K \cdot \vec{s}_3 \geq Pre \cdot \vec{t}_1 \\
(q_6, t_3) & : & M_0 + C \cdot (\vec{t}_1 + 2\,\vec{t}_2 + \vec{t}_3) + K \cdot \vec{s}_3 \geq Pre \cdot \vec{t}_3 \\
(q_6, t_4) & : & M_0 + C \cdot (\vec{t}_1 + 2\,\vec{t}_2 + \vec{t}_3) + K \cdot \vec{s}_3 \geq Pre \cdot \vec{t}_4
\end{cases}
$$

(b) Constraints related to ω-increasing sequences

$$
\begin{cases}
q_2 \in \mathcal{O},\ \pi = t_1 t_2 & : & C \cdot (\vec{t}_1 + \vec{t}_2) - \vec{s}_1 \geq \vec{0} \\
 & & -C \cdot (\vec{t}_1 + \vec{t}_2) + K \cdot \vec{s}_1 \geq \vec{0} \\
 & & \vec{1} \cdot \vec{s}_1 > 0 \\
q_4 \in \mathcal{O},\ \pi_1 = t_3,\ \pi_2 = t_4 & : & C \cdot \vec{t}_3 - \vec{s}_2 + K \cdot \vec{s}_1 \geq \vec{0} \\
 & & -C \cdot \vec{t}_3 + K \cdot \vec{s}_2 \geq \vec{0} \\
 & & C \cdot \vec{t}_4 - \vec{s}_2 + K \cdot \vec{s}_1 \geq \vec{0} \\
 & & -C \cdot \vec{t}_4 + K \cdot \vec{s}_2 \geq \vec{0} \\
 & & \vec{s}_2 \geq \vec{s}_1 \\
 & & \vec{1} \cdot \vec{s}_2 > \vec{1} \cdot \vec{s}_1 \\
q_5 \in \mathcal{O},\ \pi_1 = t_3,\ \pi_2 = t_4 & : & C \cdot \vec{t}_3 - \vec{s}_2 + K \cdot \vec{s}_1 \geq \vec{0} \\
 & & -C \cdot \vec{t}_3 + K \cdot \vec{s}_2 \geq \vec{0} \\
 & & C \cdot \vec{t}_4 - \vec{s}_2 + K \cdot \vec{s}_1 \geq \vec{0} \\
 & & -C \cdot \vec{t}_4 + K \cdot \vec{s}_2 \geq \vec{0} \\
 & & \vec{s}_2 \geq \vec{s}_1 \\
 & & \vec{1} \cdot \vec{s}_2 > \vec{1} \cdot \vec{s}_1 \\
q_6 \in \mathcal{O},\ \pi = t_2 & : & C \cdot \vec{t}_2 - \vec{s}_3 + K \cdot \vec{s}_2 \geq \vec{0} \\
 & & C \cdot \vec{t}_2 - \vec{s}_3 + K \cdot \vec{s}_2 \geq \vec{0} \\
 & & -C \cdot \vec{t}_2 + K \cdot \vec{s}_3 \geq \vec{0} \\
 & & \vec{s}_3 \geq \vec{s}_2 \\
 & & \vec{s}_3 \geq \vec{s}_2 \\
 & & \vec{1}^T \cdot \vec{s}_3 > \vec{1}^T \cdot \vec{s}_2 \\
 & & \vec{1}^T \cdot \vec{s}_3 > \vec{1}^T \cdot \vec{s}_2
\end{cases}
$$

(c) Constraints related to ω-stationary sequences

$$\begin{cases} i=3, \quad \sigma' = t_1 & : \quad C\cdot\vec{t}_1 + K\vec{s}_3 \geq \vec{0} \\ & \quad\quad -C\cdot\vec{t}_1 + K\vec{s}_3 \geq \vec{0} \\ \quad\quad \sigma'' = t_3 & : \quad C\cdot\vec{t}_3 + K\vec{s}_3 \geq \vec{0} \\ & \quad\quad -C\cdot\vec{t}_3 + K\vec{s}_3 \geq \vec{0} \\ \quad\quad \sigma''' = t_4 & : \quad C\cdot\vec{t}_4 + K\vec{s}_3 \geq \vec{0} \\ & \quad\quad -C\cdot\vec{t}_4 + K\vec{s}_3 \geq \vec{0} \end{cases}$$

(d) Blocking constraints

$$\begin{cases} (q_0,t_2) & : \quad M_0 - K\cdot\vec{s}_{0,2} < Pre\cdot\vec{t}_2 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{0,2} \leq 2 \\ (q_0,t_3) & : \quad M_0 - K\cdot\vec{s}_{0,3} < Pre\cdot\vec{t}_3 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{0,3} \leq 2 \\ (q_0,t_4) & : \quad M_0 - K\cdot\vec{s}_{0,4} < Pre\cdot\vec{t}_4 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{0,4} \leq 2 \\ (q_1,t_1) & : \quad M_0 + C\cdot\vec{t}_1 - K\cdot\vec{s}_{1,1} < Pre\cdot\vec{t}_1 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{1,1} \leq 2 \\ (q_1,t_3) & : \quad M_0 + C\cdot\vec{t}_1 - K\cdot\vec{s}_{1,3} < Pre\cdot\vec{t}_3 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{1,3} \leq 2 \\ (q_1,t_4) & : \quad M_0 + C\cdot\vec{t}_1 - K\cdot\vec{s}_{1,4} < Pre\cdot\vec{t}_4 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{1,4} \leq 2 \\ (q_2,t_2) & : \quad M_0 + C\cdot(\vec{t}_1+\vec{t}_2) - K\cdot\vec{s}_{2,2} < Pre\cdot\vec{t}_2 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{2,2} \leq 2 \\ & \quad\quad \vec{s}_{2,2} \geq \vec{s}_1 \\ (q_3,t_1) & : \quad M_0 + C\cdot(2\,\vec{t}_1+\vec{t}_2) - K\cdot\vec{s}_{3,1} < Pre\cdot\vec{t}_3 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{3,1} \leq 2 \\ & \quad\quad \vec{s}_{3,1} \geq \vec{s}_1 \\ (q_5,t_1) & : \quad M_0 + C\cdot(2\,\vec{t}_1+\vec{t}_2+\vec{t}_3) - K\cdot\vec{s}_{5,1} < Pre\cdot\vec{t}_1 \\ & \quad\quad \vec{1}^T\cdot\vec{s}_{5,1} \leq 2 \\ & \quad\quad \vec{s}_{5,1} \geq \vec{s}_2 \end{cases}$$

(e) Equivalence constraints

$$\begin{cases} \vec{\sigma}_4 = \vec{t}_1+\vec{t}_2+\vec{t}_3, & \vec{\sigma} = \vec{t}_1+\vec{t}_2+\vec{t}_4 & : \quad C\cdot\vec{\sigma}_4 - C\cdot\vec{\sigma} + K\cdot\vec{s}_2 \geq \vec{0} \\ & & \quad\quad -C\cdot\vec{\sigma}_4 + C\cdot\vec{\sigma} + K\cdot\vec{s}_2 \geq \vec{0} \\ \vec{\sigma}_5 = 2\,\vec{t}_1+\vec{t}_2+\vec{t}_3, & \vec{\sigma} = 2\,\vec{t}_1+\vec{t}_2+\vec{t}_4 & : \quad C\cdot\vec{\sigma}_5 - C\cdot\vec{\sigma} + K\cdot\vec{s}_2 \geq \vec{0} \\ & & \quad\quad -C\cdot\vec{\sigma}_5 + C\cdot\vec{\sigma} + K\cdot\vec{s}_2 \geq \vec{0} \\ \vec{\sigma}_6 = \vec{t}_1+2\,\vec{t}_2+\vec{t}_3, & \vec{\sigma} = \vec{t}_1+2\,\vec{t}_2+\vec{t}_4 & : \quad C\cdot\vec{\sigma}_6 - C\cdot\vec{\sigma} + K\cdot\vec{s}_3 \geq \vec{0} \\ & & \quad\quad -C\cdot\vec{\sigma}_6 + C\cdot\vec{\sigma} + K\cdot\vec{s}_3 \geq \vec{0} \\ & \vec{\sigma} = 2\,\vec{t}_1+2\,\vec{t}_2+\vec{t}_3 & : \quad C\cdot\vec{\sigma}_6 - C\cdot\vec{\sigma} + K\cdot\vec{s}_3 \geq \vec{0} \\ & & \quad\quad -C\cdot\vec{\sigma}_6 + C\cdot\vec{\sigma} + K\cdot\vec{s}_3 \geq \vec{0} \\ & \vec{\sigma} = 2\,\vec{t}_1+2\,\vec{t}_2+\vec{t}_4 & : \quad C\cdot\vec{\sigma}_6 - C\cdot\vec{\sigma} + K\cdot\vec{s}_3 \geq \vec{0} \\ & & \quad\quad -C\cdot\vec{\sigma}_6 + C\cdot\vec{\sigma} + K\cdot\vec{s}_3 \geq \vec{0} \end{cases}$$

(f-g) Discriminating constraints

There are no bisimilar nodes.

(h) Integrity constraints

$$\begin{cases} M_0 \in \mathbb{N}^3 \\ C = Post - Pre \\ Pre,\ Post \in \mathbb{N}^{m \times n} \\ \vec{s}_1,\ \vec{s}_2,\ \vec{s}_3 \in \{0,1\}^m \\ \vec{s}_{0,2},\ \vec{s}_{0,3},\ \vec{s}_{0,4},\ \vec{s}_{1,1},\ \vec{s}_{1,3},\ \vec{s}_{1,4},\ \vec{s}_{2,2},\ \vec{s}_{3,1},\ \vec{s}_{5,1} \in \{0,1\}^{m \times n} \end{cases}$$

The file LINDO representing the set of constraints here reported and that we used to solve the resulting IPP can be found at [50].

We identify the net system in Figure 11.1.(a) characterized by:

$$M_0 = [\ 1 \quad 0 \quad 0\ ]^T, \qquad Pre = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \qquad Post = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

■

The necessity of introducing constraints (f) and (g) will be shown in the following example.

Example 11.26. Let us consider the net systems in Figures 11.5.(a) and (d). The CG are shown in Figures 11.5.(b) and (e), respectively, while the unlabeled graphs are reported in (c) and (f). As already discussed in Example 11.22, nodes q_1 and q_2 in Figure 11.5.(f) are bisimilar.

Now, assume that our goal is that of synthesizing a net system satisfying assumption A1, whose unlabeled graph is isomorphic to that in Figure 11.5.(f) and that minimizes the sum of the tokens in the initial marking and the arc weights.

If the discriminating constraints are not taken into account we obtain the net system in Figure 11.5.(a) whose unlabeled graph is not isomorphic to the initial one, because nodes q_1 and q_2 collapse in a unique node. ■

A final remark should be done. Here we assumed that the number of places m is given. Clearly, if such is not the case, we should solve an IPP of the form (11.5) for increasing values of m, until a feasible solution is obtained. An alternative approach consists in rewriting the IPP (11.5) as suggested in Chapter 10, thus guaranteeing the optimality of the solution both in terms of minimum number of places and in terms of the chosen performance index. Note however that this results in a larger computational complexity.

11.3 Computational complexity of the proposed procedure

In this section we discuss the complexity of the IPP we must solve to identify a net. This complexity is given in terms of number of constraints and number of unknowns. Note however that it is well known that an IPP is an NP-hard problem itself.

Let us consider:

- m, n the number of places P and of transitions T of the net, respectively;
- n_g the number of nodes of the unlabeled graph;
- n_{q_i} the number of transitions enabled at state q_i;
- $|\hat{\mathscr{Q}}|$ the number of estimated equivalence classes obtained using Algorithm 11.12;
- $|\Pi|$ the number of ω-increasing sequences;
- $|\mathscr{S}|$ the number of ω-stationary sequences;
- $|\mathscr{I}(q_k)|$ the cardinality of the set of couples (state q - transition t) that lead to q_k except the couple $(q_{r(k)}, t_{r(k)})$ that has already been considered in constraints (a) (see constraints (e) in Theorem 11.23);
- $|B_i|$ the cardinality of the set of couples composed by all nodes that are bisimilar and that belong to the same equivalence class $\hat{\mathscr{Q}}_i$;
- $|B_n|$ the cardinality of the set of couples composed by all nodes that are bisimilar and that belong to equivalence classes among which an ordering does not exist.

Then the total number of linear algebraic constraints in Theorem 11.23 is shown in the following formula, where for each term it is specified to which type of constraints it corresponds:

$$\begin{aligned} c \;=\; & \overbrace{m \cdot \sum_{i=0}^{n_g-1} (n_{q_i})}^{constr.\ (a)} + \overbrace{(3 \cdot m + 1) \cdot |\Pi|}^{constr.\ (b)} + \overbrace{2 \cdot m \cdot |\mathscr{S}|}^{constr.\ (c)} + \overbrace{(2 \cdot m + 1) \cdot \sum_{i=0}^{n_g-1} (n - n_{q_i})}^{constr.\ (d)} \\ & + \underbrace{2 \cdot m \cdot \sum_{i=0}^{n_g-1} |\mathscr{I}(q_i)|}_{constr.\ (e)} + \underbrace{(3 \cdot m + 1) \cdot \sum_{i \in \hat{\mathscr{Q}}_i} |B_i|}_{constr.\ (f)} + \underbrace{(6 \cdot m + 3) \cdot |B_n|}_{constr.\ (g)}. \end{aligned}$$

The total number of unknowns is:

$$u = m + 2 \cdot (m \cdot n) + (m \cdot |\hat{\mathscr{Q}}|) + m \cdot \sum_{i=0}^{n_g-1} (n - n_{q_i}) + 2 \cdot m \cdot \sum_{i \in \hat{\mathscr{Q}}_i} |B_i| + (2 + 4 \cdot m) \cdot |B_n|,$$

where the right-side terms are due respectively to: the initial marking, the elements of the two matrices Pre and Post, the number of equivalence classes, the binary variables $\vec{s}_{k,j}$ in constraints (d), the binary variables $\vec{l}_{j,k}$ and $\vec{l}_{k,j}$ in constraints (f) and (g), the binary variables $z_1, z_2, \vec{v}_{j,k}$ and $\vec{v}_{k,j}$ in constraints (g).

The worst case in terms of number of unknowns occurs when

- $|\hat{\mathscr{Q}}| = n_g$;
- $n_{q_i} = 0$ for all the nodes q_i of the unlabeled graph.

Moreover, for sure $|B_i| \leq n^2$ for any equivalence class $\hat{\mathscr{Q}}_i$, and $|B_n| \leq n^2$, thus

$$\begin{aligned} u &\leq m + 2 \cdot (m \cdot n) + (m \cdot n_g) + m \cdot n_g \cdot n + 2 \cdot m \cdot n_g \cdot n^2 + (2 + 4 \cdot m) n^2 \\ &= \mathscr{O}(m \cdot n^2 \cdot n_g), \end{aligned}$$

i.e., it is linear with respect to m and n_g and is quadratic with respect to n.

Chapter 12

Linear Programming Techniques for the Identification of Place/Transition Nets

Summary

In Chapter 10 we presented a procedure based on integer programming to identify a Petri net, given a finite prefix of its language. In this chapter we show how to tackle the same problem using linear programming techniques, thus significantly reducing the complexity of finding a solution. The procedure we propose identifies a net whose number of places is equal to the cardinality of the set of disabling constraints. We provide a criterion to check if the computed solution has a minimal number of places, and, if such is not the case, we discuss two approaches to reduce this number. At the end of the chapter there is a comparison between our identification procedure and theory of regions approach (presented in Chapter 5).

In this chapter is presented an identification procedure that solves the problem of identify a Petri net given a finite prefix of its language (the same problem solved in Chapter 10) using linear programming techniques, thus reducing the computational complexity of finding a solution. Let us start the chapter introducing some special constraint sets and some results on these that will be useful in the rest of the chapter.

12.1 Special constraint sets

We define a special class of linear constraint sets (CS).

Definition 12.1. *Given $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$, consider the linear constraint set:*

$$\mathscr{C}(A,b) = \{x \in \mathbb{R}^n \mid Ax \geq b\}.$$

The set $\mathscr{C}(A,b)$ is called:

- ideal: *if $x \in \mathscr{C}(A,b)$ implies $\alpha x \in \mathscr{C}(A,b)$ for all $\alpha \geq 1$;*
- rational: *if $A \in \mathbb{Q}^{m\times n}$ and $b \in \mathbb{Q}^m$, i.e., if the entries of matrix A and of vector b are rational.* ■

The following result provides a simple characterization of ideal CS's.

Proposition 12.2. *A linear constraint set $\mathscr{C}(A,b)$ is ideal if $b \geq 0$.*

Proof. Since $Ax \geq b \geq 0$ then for all $\alpha \geq 1$ it holds $A(\alpha x) \geq Ax \geq b$, hence it is ideal. □

Proposition 12.3. *If a CS is ideal and rational, then it has a feasible solution if and only if it has a feasible integer solution.*

Proof. The *if* part is trivial.

To prove the *only if* part, we reason as follows. If there exists a solution there exists a basis solution x_B , i.e., such that

$$x_B = A_B^{-1} b,$$

where A_B is obtained by A selecting a set of basis columns.

If the CS is rational the entries of A_B and b are rational, hence the entries of A_B^{-1} and of x_B are rational as well.

If the CS is ideal, we just need to multiply the rational vector x_B by a suitable positive integer to obtain an integer solution. □

12.2 P/T net identification

The problem we consider in this chapter can be formally stated as follows.

Problem 12.4. *Let $\mathscr{L} \subset T^*$ be a finite prefix-closed language (see Appendix A), and*

$$k \geq \max_{\sigma \in \mathscr{L}} |\sigma|$$

be an integer greater than or equal to the length of the longest string in $\mathscr{L}$. We want to identify the structure of a net $N = (P, T, Pre, Post)$ and an initial marking M_0 such that

$$L_k(N, M_0) = \mathscr{L}.$$

The unknowns we want to determine are the elements of the two matrices $Pre,\ Post \in \mathbb{N}^{m \times n}$ and the elements of the vector $M_0 \in \mathbb{N}^m$. ■

Associated to an identification problem are the two sets defined in the following.

Definition 12.5. *Let $\mathscr{L} \subset T^*$ be a finite prefix-closed language and let $k \in \mathbb{N}$ be defined as in Problem 12.4.*

We define the set of enabling conditions

$$\mathscr{E} = \{(y, t) \mid (\exists \sigma \in \mathscr{L}) : |\sigma| < k, \sigma \in \mathscr{L}(y), \sigma t \in \mathscr{L}\} \subset \mathbb{N}^n \times T \tag{12.1}$$

and the set of disabling conditions

$$\mathscr{D} = \{(y, t) \mid (\exists \sigma \in \mathscr{L}) : |\sigma| < k, \sigma \in \mathscr{L}(y), \sigma t \notin \mathscr{L}\} \subset \mathbb{N}^n \times T. \tag{12.2}$$

■

Clearly, a solution to Problem 12.4 is a net $\langle N, M_0 \rangle$ such that:

- for all $(y, t) \in \mathscr{E}$ transition t is enabled after the firing of all $\sigma \in \mathscr{L}(y)$, i.e., $M_0[\sigma\rangle M_y[t\rangle$, where $M_y = M_0 + C \cdot y$ represents the marking reached after the firing of sequence σ.
- for all $(y, t) \in \mathscr{D}$ transition t is disabled after the firing of $\sigma \in \mathscr{L}(y)$, i.e., $M_0[\sigma\rangle M_y \neg[t\rangle$.

We can characterize the number of places required to solve our identification problem.

Definition 12.6. *Let $\mathscr{L}$ be a finite prefix-closed language on alphabet T, whose words have length less than or equal to k. Given the set of disabling conditions* (12.2) *let $m_{\mathscr{D}} = |\mathscr{D}|$.*

We say that a Petri net system $\langle N, M_0 \rangle$ with set of places P and $L_k(N, M_0) = \mathscr{L}$, is $\mathscr{D}$-canonical *if*

1. $|P| = m_{\mathscr{D}}$;

2. *there exists a bijective mapping* $h : \mathscr{D} \to P$ *such that, for all* $(y,t) \in \mathscr{D}$*, place* $p = h(y,t)$ *satisfies*

$$M_y(p) \triangleq M_0(p) + C(p,\cdot) \cdot y < Pre(p,t),$$

i.e., place p *disables* t *after any* $\sigma \in \mathscr{L}(y)$. ■

In simple words, a net system $\langle N, M_0 \rangle$ is $\mathscr{D}$*-canonical* if a different place is associated to each element in the set of disabling constraints $\mathscr{D}$.

Proposition 12.7. *Let* $\mathscr{L}$ *be a finite prefix-closed language on alphabet* T*, whose words have length less than or equal to* k*.*

If there exists a net system $\langle \tilde{N}, \tilde{M}_0 \rangle$ *such that* $L_k(\tilde{N}, \tilde{M}_0) = \mathscr{L}$*, then there exists a net system* $\langle N, M_0 \rangle$ *that is* $\mathscr{D}$*-canonical.*

Proof. Let

$$P_{(y,t)} = \{p \in \tilde{P} \mid \tilde{M}_0(p) + \tilde{C}(p,\cdot) \cdot y < \tilde{Pre}(p,t)\}$$

be the set of all places of $\tilde{N}$ that disable transition t after sequence $\sigma \in \mathscr{L}(y)$ has occurred (here $\tilde{C}$ is the incidence matrix of $\tilde{N}$). For each pair $(y,t) \in \mathscr{D}$ let $h(y,t)$ be one place arbitrary selected from $P_{(y,t)}$; let P be the set of selected places and $m = |P|$.

Two different cases may occur.

Case 1: $m = m_{\mathscr{D}}$, i.e., a different place has been selected from any set $P_{(y,t)}$. In such a case we define N as the net obtained from $\tilde{N}$ removing all places not in P (if any), and assume M_0 as the restriction of $\tilde{M}_0$ to places in P. We claim that $L_k(N, M_0) = \mathscr{L}$. In fact, since we have removed some places from $\langle \tilde{N}, \tilde{M}_0 \rangle$ then $L(\tilde{N}, \tilde{M}_0) \subseteq L(N, M_0)$. On the other hand, by construction we know that for all words wt of length less than or equal to k it holds

$$wt \notin L(\tilde{N}, \tilde{M}_0) \qquad \Longrightarrow \qquad wt \notin L(N, M_0),$$

hence $L_k(N, M_0) = L_k(\tilde{N}, \tilde{M}_0) = \mathscr{L}$.

By construction, a different place in P is associated to any couple $(y,t) \in \mathscr{D}$, thus proving that $\langle N, M_0 \rangle$ is $\mathscr{D}$-canonical.

Case 2: $m < m_{\mathscr{D}}$, i.e., some place $p \in P$ has been selected from $P_{(y,t)}, P_{(y',t')}, P_{(y'',t'')}, \ldots$, for two or more different pairs $(y,t), (y',t'), (y'',t''), \ldots$ in $\mathscr{D}$. In such a case we add to the net — without changing its language — additional places $p', p'' \ldots$ such that $Pre(p,\cdot) = Pre(p',\cdot) = Pre(p'',\cdot) = \cdots$, $Post(p,\cdot) = Post(p',\cdot) = Post(p'',\cdot) = \cdots$, and $M_0(p) = M_0(p') = M_0(p'') = \cdots$, thus obtaining a net with $m_{\mathscr{D}}$ places.

We redefine $h(y',t') = p'$, $h(y'',t'') = p'', \ldots$. Function h is now bijective and the resulting net system is $\mathscr{D}$-canonical. □

Theorem 12.8. *Let us consider a finite prefix-closed language on alphabet* T*, whose words have length less than or equal to* k *and let* $\mathscr{E}$ *and* $\mathscr{D}$ *be the corresponding sets of enabling and disabling conditions.*

Let

$$\mathscr{N}(\mathscr{E},\mathscr{D}) \triangleq \begin{cases} M_0 + Post \cdot y \\ \qquad -Pre \cdot (y + \vec{t}\,) \geq 0 & \forall (y,t) \in \mathscr{E} \\ M_0(p_{(y,t)}) + Post(p_{(y,t)},\cdot) \cdot y \\ \qquad -Pre(p_{(y,t)},\cdot) \cdot (y + \vec{t}\,) \leq -1 & \forall (y,t) \in \mathscr{D} \\ M_0 \in \mathbb{R}_{\geq 0}^{m_{\mathscr{D}}} \\ Pre, Post \in \mathbb{R}_{\geq 0}^{m_{\mathscr{D}} \times n} \end{cases} \tag{12.3}$$

Consider a net system $\langle N, M_0 \rangle$ *with* $N = (P, T, Pre, Post)$*. The system* $\langle N, M_0 \rangle$ *is a* $\mathscr{D}$*-canonical solution of the identification problem 12.4 iff* Pre*,* $Post$*,* M_0 *are integer solutions of CS* (12.3)*.*

Proof. We first show that any integer solution $\langle N, M_0 \rangle$ of CS (12.3) is a solution of Problem 12.4.

- Any constraint $M_0 + Post \cdot y - Pre \cdot (y + \vec{t}\,) \geq 0$ can be rewritten as $M_y = M_0 + (Post - Pre) \cdot y \geq Pre(\cdot, t)$ or equivalently $M_y \geq Pre(\cdot, t)$ where $M_0[\sigma\rangle M_y$ for all $\sigma \in \mathscr{L}(y)$. This shows that transition t is enabled on $\langle N, M_0 \rangle$ from marking M_y and by induction on the length of σ (since language $\mathscr{L}$ is prefix-closed) we conclude that $\sigma t \in \mathscr{L}$.
- Assume that sequence $\sigma \in \mathscr{L}(y)$ is firable on the net and $M_0[\sigma\rangle M_y$. If for at least a place p in the net it holds $M_0(p) + Post(p,\cdot) \cdot y - Pre(p,\cdot) \cdot (y + \vec{t}\,) \leq -1$, then $M_y = M_0 + (Post - Pre) \cdot y \not\geq Pre(\cdot, t)$ or equivalently $M_y \not\geq Pre(\cdot, t)$. This shows that transition t is not enabled on $\langle N, M_0 \rangle$ from marking M_y and we conclude that $\sigma t \notin \mathscr{L}$.

Since net $\langle N, M_0 \rangle$ satisfies all enabling and disabling constraints, $L_k(N, M_0) = \mathscr{L}$.

We now show that any solution of CS (12.3) is $\mathscr{D}$-canonical. In fact, the mapping $h(y,t) = p_{(y,t)}$ for each couple $(y,t) \in \mathscr{D}$ is bijective.

We now show that any $\mathscr{D}$-canonical net system $\langle N, M_0 \rangle$ with $L_k(N, M_0) = \mathscr{L}$ is a solution of CS (12.3). In fact, let $h : \mathscr{D} \to P$ be the bijective function of the net system. If we define $p_{(y,t)} = h(y,t)$ for all $(y,t) \in \mathscr{D}$, then all equations in CS (12.3) are satisfied. □

Proposition 12.9. *The linear CS* (12.3) *is ideal and rational.*

Proof. We first observe that the linear CS (12.3) can be rewritten as a set of linear inequalities of the form $Ax \geq b$ as follows. Let us denote as pre_i and $post_i$ the i-th row of matrices Pre and $Post$, respectively, for $i = 1, \ldots, m_{\mathscr{D}}$.

For any $(y,t) \in \mathscr{E}$ the first matrix inequality in (12.3) can be rewritten as the following set of $m_{\mathscr{D}}$ scalar inequalities:

$$\begin{bmatrix} 1 & y^T & -(y + \vec{t}\,)^T \end{bmatrix} \cdot \begin{bmatrix} M_{0,i} \\ post_i^T \\ pre_i^T \end{bmatrix} \geq 0$$

where $i = 1, \ldots, m_{\mathscr{D}}$. Analogously, for any $(y,t) \in \mathscr{D}$ the second scalar inequality in (12.3) can be rewritten as:

$$\begin{bmatrix} -1 & -y^T & (y + \vec{t}\,)^T \end{bmatrix} \cdot \begin{bmatrix} M_{0,i} \\ post_i^T \\ pre_i^T \end{bmatrix} \geq 1.$$

This defines matrix A and vector b. Since A and b have integer entries, CS (12.3) is rational. Since $b \geq 0$, by Proposition 12.2 CS (12.3) is ideal. □

The following theorem provides a practical and efficient procedure to solve our identification problem.

Theorem 12.10. *The identification problem 12.4 admits a solution if and only if the (linear) CS* (12.3) *is feasible.*

Proof. (if) By Proposition 12.9 CS (12.3) is ideal and rational thus by Proposition 12.3 if it has solutions, then it also has feasible integer solutions. However, by Theorem 12.8 this also implies that such integer solutions are also solutions of the identification problem 12.4.

(only if) If there exists a solution of Problem 12.4, by Proposition 12.7 there also exists a net system that is $\mathscr{D}$-canonical. But all $\mathscr{D}$-canonical systems are solutions of CS (12.3) by Theorem 12.8, thus CS (12.3) is feasible. □

Note, finally, that once a solution of CS (12.3) is found, if this solution is rational we can always find an integer solution by simply multiplying Pre, $Post$ and M_0 by a suitable $\alpha \geq 1$.

12.3 Place reduction

One drawback of the identification procedure outlined in the previous section consists in the requirement that the net contains a number of places equal to $m_{\mathscr{D}} = |\mathscr{D}|$ although an equivalent net with a smaller number of places may exist. Note that in the worst case the cardinality of the set $\mathscr{D}$ can be $|T|^k$ [20].

We propose two (types of) approaches to overcome this problem. In the first approach, that we call *place pre-reduction*, CS (12.3) is written in a modified form, using a reduced number of places. In the second approach, that we call *place post-reduction*, we first determine a solution of the standard CS (12.3) obtaining a net with $m_{\mathscr{D}}$ places and then we identify redundant places that can be removed without affecting the correctness of the result.

12.3.1 Place pre-reduction

We start with a general result that allows one to check if the net obtained by solving CS (12.3) has a minimal number of places. The test we propose requires solving a series of modified CS's and this is why we present this result in the subsection devoted to the pre-reduction.

Definition 12.11. *Consider a partition*

$$\Pi(\mathscr{D}) = \{D_1, D_2, \ldots, D_q\}$$

of the set $\mathscr{D}$. *The sets* D_i *are called* blocks *of partition* $\Pi(\mathscr{D})$.

We define the following CS

$$\mathcal{N}(\mathscr{E},\Pi(\mathscr{D})) \triangleq \begin{cases} M_0 + Post \cdot y \\ \quad -Pre \cdot (y + \vec{t}\,) \geq 0 & \forall (y,t) \in \mathscr{E} \\ M_0(p_i) + Post(p_i,\cdot) \cdot y \\ \quad -Pre(p_i,\cdot) \cdot (y + \vec{t}\,) < 0 & \forall (y,t) \in D_i, \\ & i = 1,\ldots,q \\ M_0 \in \mathbb{R}^q_{\geq 0} \\ Pre, Post \in \mathbb{R}^{q \times n}_{\geq 0} \end{cases} \tag{12.4}$$

where $\mathscr{E}$ and $n = |T|$ have the usual meaning as in Theorem 12.10. ■

The only difference between CS (12.4) and CS (12.3) consists in the fact that in the former only q places (as many as the blocks of partition $\Pi(\mathscr{D})$) are used: place p_i ($i = 1,\ldots,q$) will ensure that all disabling conditions in D_i are enforced. It is immediate to prove, with the same reasoning of Proposition 12.9, that CS (12.4) is rational and ideal, and that any of its integer solutions is a solution to the identification problem 12.4.

Definition 12.12. *Given the identification problem 12.4, a partition $\Pi(\mathscr{D}) = \{D_1, D_2, \ldots, D_q\}$ with q blocks is said to be*

- feasible *if CS $\mathcal{N}(\mathscr{E},\Pi(\mathscr{D}))$ admits a solution;*
- minimal *if it is feasible and there exists no other partition $\Pi'(\mathscr{D})$ with $q' < q$ that is feasible.*

■

Thus the number of blocks of a minimal partition represents the minimal number of places that a net solving the given identification problem may have.

The following corollary trivially follows from the previous definitions and from Theorem 12.10.

Corollary 12.13. *A net with q places solution of the identification problem 12.4 exists iff there exists a feasible partition $\Pi(\mathscr{D}) = \{D_1, D_2, \ldots, D_q\}$ with q blocks solution of CS* (12.4).

We now state an intuitive result that allows one to determine if a partition is minimal.

Proposition 12.14. *A feasible partition with q blocks is minimal iff there exists no feasible partition with $q-1$ blocks.*

Proof. The *only if* part follows from the definition of minimal partition.

To prove the *if* part we need to show that if no feasible partition with $q-1$ blocks exists, then no partition with a smaller number of blocks is feasible. This can be proved by contradiction, by means of the same argument used in the proof of Proposition 12.7, Case 2. In fact, assume there exists a feasible partition with $q' < q-1$ blocks; then there exists a net solving the identification problem with q' places. However, we can add an arbitrary number of duplicate

place to this net and this implies that there exists a net solving the identification problem with $q'+1, q'+2, \ldots, q-1$ places. Thus, according to Proposition 12.13, there exists a feasible partition with $q-1$ blocks which is a contradiction. □

According to the previous proposition to prove that a feasible partition $\Pi(\mathscr{D})$ with q blocks is minimal it is necessary to check the feasibility of all partitions $\mathscr{D}$ with $q-1$ blocks. However, the number of partitions of a set of cardinality n into k blocks is given by the *Sterling number of the second kind*[1] $S(n,k)$ [36] which may be too large for an exhaustive analysis.

With the terminology introduced in this section, CS (12.3) can be seen as a special case of CS (12.4) when the considered partition $\Pi(\mathscr{D})$ contains all singleton sets, i.e., it is the unique partition with $m_{\mathscr{D}}$ blocks. Thus it is possible to check if this partition is minimal by checking that all partitions of $m_{\mathscr{D}}-1$ blocks (obtained by merging any two singleton sets) are not feasible. There exists

$$S(m_{\mathscr{D}}, m_{\mathscr{D}}-1) = \frac{m_{\mathscr{D}}(m_{\mathscr{D}}-1)}{2}$$

of these partitions.

Although the previous results provide a procedure for reducing the number of places of a Petri net, a brute force search to determine a minimal feasible partition is not viable given the large number of such partitions. We conclude this subsection with an informal discussion on how it may possible to exploit some additional information on the net to determine a feasible partition of cardinality smaller than $m_{\mathscr{D}}$.

As an example, assume it is known that a transition t has only one input place — such is the case if the net to identify or a subnet of it containing t is a state machine. In such a case, it is possible to consider a partition of $\mathscr{D}$ in which a single block

$$D = \{(y_1, t), \ldots, (y_r, t)\} \subset \mathscr{D}$$

contains all disabling conditions for transition t and a single place p will be used to disable t after a sequence $\sigma_i \in \cup_{i=1}^{r} \mathscr{L}(y_i)$ has been executed.

As a second example, assume it is known that transitions t and t' are in a free choice relation, i.e., there exists a place $p = {}^\bullet t = {}^\bullet t'$ that is the unique input place for both transitions. In such a case, it is possible to consider a partition of $\mathscr{D}$ in which a single block

$$D = \{(y_1, t), \ldots, (y_r, t), (y'_1, t'), \ldots, (y_{r'}, t')\} \subset \mathscr{D}$$

contains all disabling conditions for transitions t and t' which will be enforced by place p.

Example 12.15. Let us consider a language

$$\mathscr{L} = \{\varepsilon, t_1, t_1 t_2, t_1 t_3, t_1 t_2 t_3, t_1 t_3 t_2, t_1 t_3 t_3\}$$

[1] An explicit formula for the Sterling number of the second kind is

$$S(n,k) = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n.$$

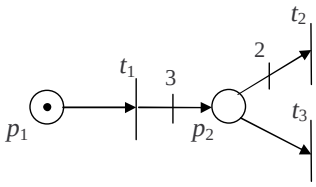


Figure 12.1: The Petri net systems in Example 12.15.

and let $k = 3$. We have additional information: the transition t_1 has only one input place and transitions t_2 and t_3 are in a free choice relation. The set of enabling and disabling constraints are respectively:

$$\mathscr{E} = \{(\varepsilon, t_1), (t_1, t_2), (t_1, t_3), (y_{12}, t_3), (y_{13}, t_2), (y_{13}, t_3)\}$$

and

$$\mathscr{D} = \{(\varepsilon, t_2), (\varepsilon, t_3), (t_1, t_1), (y_{12}, t_1), (y_{12}, t_2), (y_{13}, t_1)\},$$

where y_{12}, y_{13} are the firing vectors of $t_1 t_2$ and $t_1 t_3$ respectively. The additional information allows us to consider two different blocks of $\mathscr{D}$:

$$D_1 = \{(t_1, t_1), (y_{12}, t_1), (y_{13}, t_1)\} \subset \mathscr{D}$$

and

$$D_2 = \{(\varepsilon, t_2), (\varepsilon, t_3), (y_{12}, t_2)\} \subset \mathscr{D}.$$

Let us observe that $\mathscr{D} = D_1 \cup D_2$, then the Petri net solution has $|P| = 2$. A net system solution of $\mathscr{N}(\mathscr{E}, \mathscr{D})$ computed with a commercial LP solver (LINDO) is reported in Figure 12.1.

Note that such a solution has been determined associating a linear objective function $f(M_0, Pre, Post)$ to $\mathscr{N}(\mathscr{E}, \mathscr{D})$, and solving the resulting linear programming problem. In particular, we assumed

$$f(M_0, Pre, Post) = 1^T \cdot M_0 + 1^T \cdot Pre \cdot 1 + 1^T \cdot Post \cdot 1.$$

Note finally that the solution we found out was integer. ■

12.3.2 Place post-reduction

Once a net has been identified solving CS (12.3) (or even CS (12.4)) it is always possible to check if some of the places are *redundant* and can be removed without affecting the correctness of the result. This check is based on the notion of *minimal hitting set* defined in the following.

Proposition 12.16. *Consider a net system $\langle N, M_0 \rangle$ solution of the identification problem 12.4 and define for all disabling conditions $(y, t) \in \mathscr{D}$ the set*

$$P_{(y,t)} = \{p \in P \mid M_0(p) + C(p, \cdot) \cdot y < Pre(p, t)\} \tag{12.5}$$

consisting of all places of the net that disable transition t after a sequence $\sigma \in \mathscr{L}(y)$ has been executed.

Assume $\hat{P} \subset P$ is a hitting set for all $P_{(y,t)}$'s, i.e., $\hat{P} \cap P_{(y,t)} \neq \emptyset$ for all $(y,t) \in \mathscr{D}$. Then the net system $\langle \hat{N}, \hat{M}_0 \rangle$ obtained from $\langle N, M_0 \rangle$ removing all places in $P \setminus \hat{P}$ is a solution of the identification problem 12.4.

Proof. As already discussed in the proof of Proposition 12.7 the removal of a place does not affect any enabling condition. Furthermore, if $\hat{P}$ is a hitting set for all $P_{(y,t)}$'s then it is capable of enforcing all disabling conditions in $\mathscr{D}$. Hence $L_k(\hat{N}, \hat{M}_0) = L_k(N, M_0)$. □

The places in $P \setminus \hat{P}$ that can be removed from the net system $\langle N, M_0 \rangle$ without changing its language $L_k(N, M_0)$ are called *redundant places*.

Since the net $\tilde{N}$ has set of places $\hat{P}$, to obtain a net with a minimal set of places we need to determine the minimal hitting set. This problem is known to be NP-hard and there exists several ways to compute minimal hitting sets (see [61] for a review). Here we present a straightforward algorithm based on integer programming.

Proposition 12.17. *Consider a net system $\langle N, M_0 \rangle$ with $m = |P|$ places solution of the identification problem 12.4. For all disabling conditions $(y,t) \in \mathscr{D}$, let $z_{(y,t)} \in \{0,1\}^m$ be the characteristic vector of set $P_{(y,t)}$ defined in (12.5), i.e., $z_{(y,t)}(p) = 1$ if $p \in P_{(y,t)}$, else $z_{(y,t)}(p) = 0$.*

Consider the following integer programming problem (IPP):

$$\begin{array}{lll} \min & 1^T \cdot x & \\ s.t. & x^T \cdot z_{(y,t)} \geq 1 & \forall (y,t) \in \mathscr{D} \\ & x \in \{0,1\}^m & \end{array} \tag{12.6}$$

and let x^* be an optimal solution.

Then a minimal hitting set for all $P_{(y,t)}$'s is the set $\hat{P} = \{p \in P \mid x^*(p) = 1\}$.

Proof. It is immediate to see that any feasible solution x of IPP (12.6) is the characteristic vector of a hitting set for all $P_{(y,t)}$'s because it contains at least one element from each of these sets ($x^T \cdot z_{(y,t)} \geq 1$). The optimal solution x^* has the minimal number of non-zero components, hence it corresponds to a minimal hitting set. □

Example 12.18. Let

$$\mathscr{L} = \{\varepsilon, t_1, t_2\}$$

and $k = 2$. We observe that this is a particular case of a language containing no word of length k, i.e., all words of length k have to be disabled.

The set of enabling and disabling constraints are respectively:

$$\mathscr{E} = \{(\varepsilon, t_1), (\varepsilon, t_2)\}$$

and

$$\mathscr{D} = \{(t_1, t_1), (t_1, t_2), (t_2, t_1), (t_2, t_2)\}.$$

A net system solution of $\mathscr{N}(\mathscr{E}, \mathscr{D})$ is reported in Figure 13.5.(a): here it is obviously $|P| = m_{\mathscr{D}} = 4$.

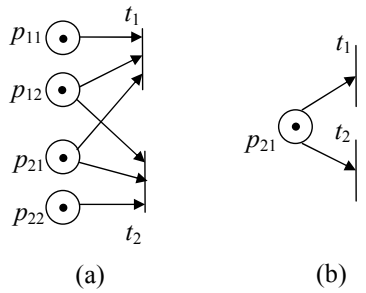


Figure 12.2: The Petri net systems in Example 12.18.

Note that such a solution has been determined associating the same linear objective function $f(M_0, Pre, Post)$ used in Example 12.15 to $\mathcal{N}(\mathcal{E}, \mathcal{D})$, and solving the resulting linear programming problem.

We now try to reduce the number of places using the place post-reduction approach. To this aim for any (y, t) we compute the set $P_{(y,t)}$ defined as in equation (12.5). Being $P_{(t_1,t_1)} = \{p_{11}, p_{12}, p_{21}\}$, $P_{(t_1,t_2)} = \{p_{12}, p_{21}\}$, $P_{(t_2,t_1)} = \{p_{12}, p_{21}\}$, $P_{(t_2,t_2)} = \{p_{12}, p_{21}, p_{22}\}$, it is immediate to see that a possible hitting set for all $P_{(y,t)}$'s is $\hat{P} = \{p_{21}\}$, that is also minimal. The resulting net system $\langle \hat{N}, \hat{M}_0 \rangle$ obtained from the previous one removing all places in $P \setminus \hat{P}$ is shown in Figure 13.5.(b). All solutions, obtained with LINDO, are integer. ■

It is important to observe that the place post-reduction procedure does not necessarily ensure that the resulting net is the solution of a given identification procedure with the minimal number of places. In fact, it only determines, amongst the solutions of a given identification procedure *that can be obtained from a solution N by removing places*, the one with a minimal number of places. The following example will clarify this point.

Example 12.19. Let

$$\mathcal{L} = \{\varepsilon, t_1, t_2, t_3\}$$

and $k = 2$, thus as in Example 12.18, all words of length k have to be disabled. The set of enabling and disabling constraints are respectively:

$$\mathcal{E} = \{(\varepsilon, t_1), (\varepsilon, t_2), (\varepsilon, t_3)\}$$

and

$$\mathcal{D} = \{(t_1, t_1), (t_1, t_2), (t_1, t_3), (t_2, t_1), (t_2, t_2), (t_2, t_3), (t_3, t_1), (t_3, t_2), (t_3, t_3)\}.$$

A net system solution of CS (12.3) is shown in Figure 12.3.(a), where obviously $|P| = m_{\mathcal{D}} = 9$. It has been determined solving a linear programming problem whose objective function is the same of that used in Example 12.15.

Now, being $P_{(t_1,t_1)} = \{p_{11}, p_{12}, p_{13}, p_{21}, p_{31}\}$, $P_{(t_1,t_2)} = \{p_{12}, p_{21}\}$, $P_{(t_1,t_3)} = \{p_{13}, p_{31}\}$, $P_{(t_2,t_1)} = \{p_{12}, p_{21}\}$, $P_{(t_2,t_2)} = \{p_{12}, p_{21}, p_{22}, p_{23}, p_{32}\}$, $P_{(t_2,t_3)} = \{p_{23}, p_{32}\}$, $P_{(t_3,t_1)} = \{p_{13}, p_{31}\}$, $P_{(t_3,t_2)} = \{p_{23}, p_{32}\}$, $P_{(t_3,t_3)} = \{p_{13}, p_{23}, p_{31}, p_{32}, p_{33}\}$, it is easy to see that a possible hitting set for all $P_{(y,t)}$'s is $\hat{P} = \{p_{21}, p_{31}, p_{32}\}$. The net system $\langle \hat{N}, \hat{M}_0 \rangle$ obtained from $\langle N, M_0 \rangle$ removing all places in $P \setminus \hat{P}$ is shown in Figure 12.3.(b).

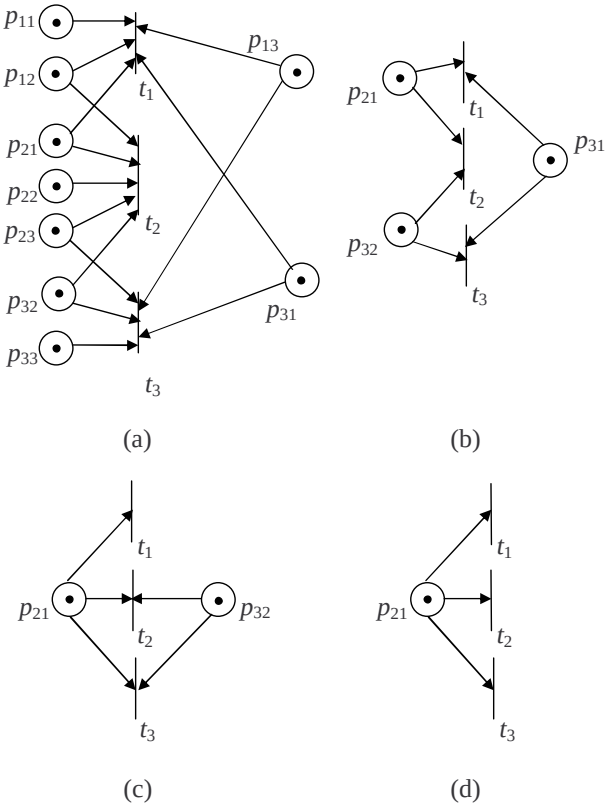


Figure 12.3: The Petri net systems in Example 12.19.

Note that the same hitting set can also be obtained solving the IPP (12.6).

Such a solution is not minimal in terms of places, as it can be verified using the place pre-reduction procedure.

To this aim we first look at a solution with two places, namely using the notation of Definition 12.11, we check if there exists a solution with only two blocks $\mathscr{D}'_1$ and $\mathscr{D}'_2$, obtained by merging two of the three blocks

$$\mathscr{D}_1 = \{(t_1,t_1),(t_1,t_2),(t_2,t_1),(t_2,t_2)\},$$

$$\mathscr{D}_2 = \{(t_2,t_3),(t_3,t_2),(t_3,t_3)\}$$

and

$$\mathscr{D}_3 = \{(t_1,t_3),(t_3,t_1)\}$$

relative respectively to the places p_{21}, p_{32} and p_{31} of the net in Figure 12.3.(b).

In particular, we find that CS (12.4) admits a solution when

$$\mathscr{D}'_1 = \{t_1,t_1),(t_1,t_2),(t_1,t_3),(t_2,t_1),(t_2,t_2),(t_3,t_1)\}$$

and

$$\mathscr{D}'_2 = \{(t_2,t_3),(t_3,t_2),(t_3,t_3)\},$$

where

$$\mathscr{D}_1' = \mathscr{D}_1 \cup \mathscr{D}_3.$$

The resulting net $\langle \bar{N}, \bar{M}_0 \rangle$ is shown in Figure 12.3.(c), where $\bar{P} = \{p_{21}, p_{32}\}$.

We can further on reduce the net finding a minimal hitting set. Being, $P_{(t_1,t_1)} = \{p_{21}\}$, $P_{(t_1,t_2)} = \{p_{21}\}$, $P_{(t_1,t_3)} = \{p_{21}\}$, $P_{(t_2,t_1)} = \{p_{21}\}$, $P_{(t_2,t_2)} = \{p_{21}, p_{32}\}$, $P_{(t_2,t_3)} = \{p_{21}, p_{32}\}$, $P_{(t_3,t_1)} = \{p_{21}\}$, $P_{(t_3,t_2)} = \{p_{21}, p_{32}\}$, $P_{(t_3,t_3)} = \{p_{21}, p_{32}\}$, the unique minimal hitting set is $\hat{P}' = \{p_{21}\}$. The net system $\langle \hat{N}', \hat{M}_0' \rangle$ obtained from $\langle \bar{N}, \bar{M}_0 \rangle$ removing all places in $\bar{P} \setminus \hat{P}'$ is minimal and is shown in Figure 12.3.(d).

Note that in all cases, the net solutions, obtained with LINDO, are integer. ■

As a final remark we observe that the notion of redundant place that we give here is different from the one of *implicit place* used by other authors [43]. In fact, an implicit place is a place that can be removed from a net system $\langle N, M_0 \rangle$ without changing its overall behavior $L(N, M_0)$. On the contrary, a redundant place according to our definition is a place that can be removed from the net without changing the finite prefix behavior $L_k(N, M_0)$. Thus our notion is weaker and the techniques used in [43] to determine implicit places cannot be used in our framework.

12.4 A comparison between theory of regions and our identification approach

The identification approach based on theory of regions and our identification approach described in Chapters 10, 11, 12 have a lot of points of contact.

- They both set out to synthesize a PN starting either from a labeled graph or from a given language.
- The method presented in Section 5.1 uses a set of constraints very similar to those presented in this chapter. Moreover, in both cases is used a linear technique to solve the problem.
- Both identification procedures have a computational complexity that is exponential. In our procedure the complexity is exponential in the length of the longest string in the input language. In theory of regions, as discussed in Section 5.3, the complexity is exponential in the number of states and events of the input graph.

Note that our approach based on integer programming problems and presented in Chapter 10 presents an high computational complexity. However, it can minimize the number of places and in general determine an "optimal" net according to a given objective function, because it does not require to introduce as many places as the number of disabling constraints.

Chapter 13

Fault Model Identification with Petri Nets

Summary

Most of the fault identification problems in the Discrete Event Systems literature assume knowledge of the structure of the net system, including the nature (and behavior) of the possible faults. In this chapter we deal with this problem within the framework of Petri nets by removing the requirement that the nature (and behavior) of the fault is known. In particular, we devise a way to identify the structure of the faulty transitions of the system given its language. Then, we generalize this procedure to unobservable faults, in which case the structure of the faulty system needs to be recognized from the knowledge of the structure of the fault-free system, and the projection of the faulty system language on the set of non-faulty events, that are assumed to be observable.

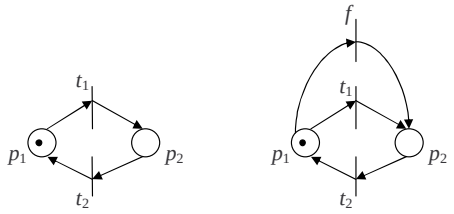


Figure 13.1: A motivational example.

13.1 Motivational example

Let us suppose we know the fault-free system and our goal is to identify the structure of the faulty system, namely the additional transitions that contribute to the faulty behavior. We consider two different cases. First, we assume that the language of the faulty system is completely known. In such a case the problem reduces to an identification problem that can be solved using the approach presented in Chapter 10. The only difference is the addition of appropriate constraints that enforce the (known) structure of the fault-free system. Second, we consider faults that are unobservable, which implies that identification should only be based on the projection of the faulty system language on the set of non-faulty (observable) events.

As an example, consider the fault-free net system in Figure 13.1.(a), whose language is $\mathscr{L} = \{\varepsilon\} \cup \{(t_1 t_2)^n \mid n \geq 0\} \cup \{(t_1 t_2)^n t_1 \mid n \geq 0\}$. Assume that a fault f may occur, and that the observable language of the system with faults is $\mathscr{L}^F = \{\varepsilon\} \cup \{((t_1 + \varepsilon)\ t_2)^n \mid n \geq 0\} \cup \{(t_1 t_2)^n t_1 \mid n \geq 0\}$. We have to identify a net system that coincides with the net system in Figure 13.1.(a) if the fault transition and its connected arcs are removed, and whose language projected on $\{t_1, t_2\}$ is equal to $\mathscr{L}^F$. Clearly, a solution to this is given by the net system in Figure 13.1.(b); however, this is not the only possible solution. Thus, we have to associate an appropriate performance index to select one solution within the set of admissible ones.

13.2 Problem Statements

Assume that a net system $\langle N, M_0 \rangle$ generating a nominal (i.e., *fault-free*) language $\mathscr{L}$ is given and let $N = (P, T, Pre, Post)$ be its net structure. We consider a *faulty* net system $\langle N^F, M_0 \rangle$, where $N^F = (P, T^F, Pre^F, Post^F)$, with the same number of places and the same initial marking as the nominal one. However, its set of transitions is $T^F = T \cup T_f$, where $T_f = \{f_1, \cdots, f_q\}$ is the set of faulty transitions. Furthermore we make the following assumption.

Assumption (A1): The pre and post incidence matrices of the faulty net are

$$Pre^F = \left[\ Pre \quad Pre^{f_1} \quad \cdots \quad Pre^{f_q}\ \right],$$

$$Post^F = \left[\ Post \quad Post^{f_1} \quad \cdots \quad Post^{f_q}\ \right],$$

where Pre^{f_i} (resp., $Post^{f_i}$) is the $m \times 1$ Pre (resp., Post) incidence matrix of transition f_i. ■

According to this assumption, the faulty net retains the structure of the nominal one but includes a number of additional faulty transitions.

We consider two different problem statements: in the first one the occurrence of fault transitions is observable, whereas in the second one faults are unobservable.

13.2.1 Case I: Faults are Known

Problem 13.1. *Let us consider a fault-free net system $\langle N, M_0\rangle$. Let $\mathscr{L}^F$ be a finite prefix-closed language over alphabet $T^F = T \cup T_f$ (see Appendix A), where $T_f = \{f_1, \ldots, f_q\}$, and such that all strings in $\mathscr{L}^F$ have length less than or equal to k.*

We want to identify a faulty net system $\langle N^F, M_0\rangle$, satisfying (A1) and such that $L_k(N^F, M_0) = \mathscr{L}^F$. ■

In simple terms, here we are assuming that the number of faults and their effect on the net behavior (i.e., the language of the resulting system) are known. Our goal is that of identifying the structure of the system with faults, namely the weights of the arcs incident on fault transitions f_1, …, f_q, under the constraint that the structure of the fault-free system is kept intact.

The next result characterizes the existence of a solution for this problem.

Proposition 13.2. *Given a fault-free system $\langle N, M_0\rangle$, let $\mathscr{L} = L_k(N, M_0) \subset T^*$.*

A necessary condition for the existence of a solution to Problem 13.1 is that $\mathscr{L}^F \subset (T^F)^*$ satisfies $\mathscr{L} = \mathscr{L}^F \cap T^*$, i.e., all words that are firable in the faulty system and consist of fault-free transitions can also be fired in the fault-free system.

Proof. Consider a word $w \in T^*$. According to assumption (A1), this word is firable in $\langle N, M_0\rangle$ if and only if it is also firable in $\langle N^F, M_0\rangle$. □

13.2.2 Case II: Faults are Unobservable

Let us define the *projection operator* $P : (T^F)^* \to T^*$ recursively as follows:

(i) $P(t_j) = t_j \;\; \forall t_j \in T$;

(ii) $P(f_i) = \varepsilon \;\; \forall f_i \in T_f$;

(iii) $P(\sigma t_j) = P(\sigma)P(t_j) \;\; \forall \sigma \in (T^F)^*, t_j \in T^F$.

Problem 13.3. *Let us consider a fault-free net system $\langle N, M_0\rangle$. Let $\mathscr{L}^F$ be a finite prefix-closed language over T whose strings have length less than or equal to k.*

Let $\Lambda(\mathscr{L}^F)$ be the set of languages over T^F whose projection over T is equal to $\mathscr{L}^F$, i.e.,

$$\Lambda(\mathscr{L}^F) = \{\mathscr{L} \subset (T^F)^* \;:\; P(\mathscr{L}) = \mathscr{L}^F\}.$$

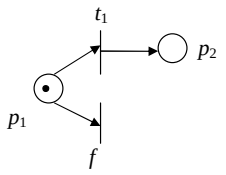


Figure 13.2: A Petri net where $\mathscr{L} = \mathscr{L}^F = \{\varepsilon, t_1\}$.

We want to identify a faulty net system $\langle N^F, M_0\rangle$ satisfying (A1) and such that $L_k(N^F, M_0) \in \Lambda(\mathscr{L}^F)$. ■

In simple terms, here we are assuming that faults are *unobservable* events. Our goal is to identify the structure of the faulty system, based on the knowledge of its observable language, namely the projection of its firing sequences over the set of observable transitions T.

Our next result characterizes the existence of a solution for this problem.

Proposition 13.4. *Given a fault-free system $\langle N, M_0\rangle$, let $\mathscr{L} = L_k(N, M_0) \subset T^*$.*

A necessary condition for the existence of a solution to Problem 13.3 is that $\mathscr{L} \subseteq \mathscr{L}^F$.

Proof. Assumption (A1) guarantees that all sequences firable in $\langle N, M_0\rangle$ can also be fired in $\langle N^F, M_0\rangle$. Thus Problem 13.3 is well-posed only if $\mathscr{L}^F$ contains all the sequences in $\mathscr{L}$. □

As a final remark, note that in Case II we can only identify faults generating strings whose observable projection is not contained in the language of the nominal system. The following example clarifies this.

Example 13.5. Let us consider the net system in Figure 13.2, where $T = \{t_1\}$ and $T_f = \{f\}$. Here $\mathscr{L} = \mathscr{L}^F = \{\varepsilon, t_1\}$, i.e., the nominal language coincides with the observable language of the faulty system. This means that after the firing of fault transition f no anomalous string will be observed, thus this fault cannot be identified. ■

13.3 Fault Identification in Case I

In this section we show how Problem 13.1 can be easily solved using our results presented in Chapter 10. The idea is that of providing an algebraic characterization of the set of admissible faulty systems. Then, the identification problem is formulated in terms of a linear IPP.

First let us introduce some sets that we will use in the following.

Definition 13.6. *Let $\mathscr{L} \subset T^*$ be defined as in Proposition 13.2, and $\mathscr{L}^F \subset T^F$ and $k \in \mathbb{N}$ be defined as in Problem 13.1.*

We define the following sets

$$\bar{\mathscr{E}}_A^F = \{(\sigma, t_j) \mid \sigma \in \mathscr{L}^F, |\sigma| < k, \sigma t_j \in \mathscr{L}^F \setminus \mathscr{L}\},$$

$$\bar{\mathscr{E}}^F = \bar{\mathscr{E}}_A^F|_{\equiv}, \tag{13.1}$$

$$\begin{aligned}\bar{\mathscr{D}}_A^F = \; & \{(\sigma, t_j) \mid \sigma \in \mathscr{L}^F \setminus \mathscr{L}, |\sigma| < k, t_j \in T, \sigma t_j \notin \mathscr{L}^F\} \\ & \cup \\ & \{(\sigma, t_j) \mid \sigma \in \mathscr{L}^F, |\sigma| < k, t_j \in T_f, \sigma t_j \notin \mathscr{L}^F\},\end{aligned}$$

$$\bar{\mathscr{D}}^F = \bar{\mathscr{D}}_A^F|_{\equiv}, \tag{13.2}$$

where $\bar{\mathscr{E}}^F$ and $\bar{\mathscr{D}}^F$ are the sets containing only one element of each equivalent class for the $\equiv$ relation in Definition 10.2. ■

Proposition 13.7. *Let us consider Problem 13.1, and let*

$$g(Pre^{f_1}, \ldots, Pre^{f_q}, Post^{f_1}, \ldots, Post^{f_q}) = \sum_{i=1}^{m} \sum_{j=1}^{q} \left[b_{i,j} Pre^{f_j}(p_i) + c_{i,j} Post^{f_j}(p_i) \right]$$

be a given linear performance index, where $b_{i,j}$, $c_{i,j} \in \mathbb{R}_0^+$.

A solution of Problem 13.1 that is optimal with respect to $g(Pre^{f_1}, \ldots, Pre^{f_q}, Post^{f_1}, \ldots, Post^{f_q})$ can be computed by solving the following IPP

$$\begin{cases} \min & g(Pre^{f_1}, \ldots, Pre^{f_q}, Post^{f_1}, \ldots, Post^{f_q}) \\ s.t. & \mathscr{G}_m(\bar{\mathscr{E}}^F, \bar{\mathscr{D}}^F) \\ & M_0 \; is \; given \end{cases} \tag{13.3}$$

where $\bar{\mathscr{E}}^F$ and $\bar{\mathscr{D}}^F$ are defined in Eq. (13.1), (13.2)*, respectively.*

Proof. Follows from Theorem 10.4 and the fact that we have to impose the enabling and disabling constraints only for those sequences that contain fault transitions. In fact, by assumption (A1) all sequences that are enabled in the fault-free net are also enabled in the faulty system. □

Example 13.8. Let us consider the token passing communication system represented in Figure 13.3(a), where p_1, p_2, p_3 represent three different agents.

The nominal language of this system is $\mathscr{L} = \{\varepsilon, t_1, t_3, t_1t_2, t_3t_4, t_1t_2t_1, t_1t_2t_3, t_3t_4t_3, t_3t_4t_1, t_1t_2t_1t_2, t_1t_2t_3t_4, t_3t_4t_3t_4, t_3t_4t_1t_2\}$.

Assume we can detect four different types of events denoting faults: f_1, f_2, f_3 and f_4. Monitoring several identical instances of this communication system the following set of strings have been observed: $\mathscr{L}^F = \{\varepsilon, f_1, t_1, t_3, t_1t_2, t_1f_2, t_1f_4, t_3t_4, t_3f_3, t_1t_2t_1, t_1t_2t_3, t_1t_2f_1, t_1f_4f_3, t_1f_4t_4, t_3t_4t_3, t_3t_4t_1, t_3t_4f_1, t_1t_2t_1t_2, t_1t_2t_1f_2, t_1t_2t_1f_4, t_1t_2t_3t_4, t_1t_2t_3f_3, t_1f_4t_4f_1, t_1f_4t_4t_1, t_1f_4t_4t_3, t_3t_4t_3t_4, t_3t_4t_3f_3, t_3t_4t_1f_4, t_3t_4t_1t_2, t_3t_4t_1f_2\}$ thus $k = 4$. Assume that we want to

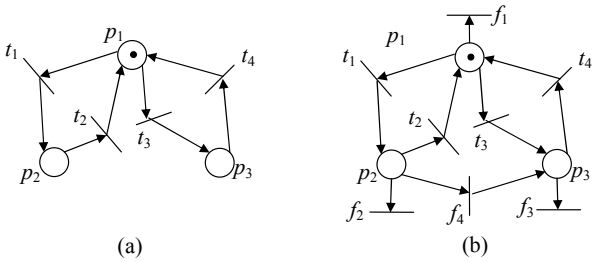


Figure 13.3: (a) The faulty-free net system and (b) the faulty net system identified in Example 13.8.

determine the Petri net system that minimizes the arc weights incident on the fault transitions such that $L_k(N^F, M_0) = \mathscr{L}^F$. This requires the solution of a linear IPP of the form (13.3) where

$$\begin{aligned}\bar{\mathscr{E}}^F = \bar{\mathscr{E}}^F_A = \quad & \{(\varepsilon, f_1), (t_1, f_2), (t_1, f_4), (t_3, f_3), (t_1t_2, f_1), (t_1f_4, f_3), (t_1f_4, t_4), (t_3t_4, f_1), (t_1t_2t_1, f_2),\\ & (t_1t_2t_1, f_4), (t_1t_2t_3, f_3), (t_1f_4t_4, f_1), (t_1f_4t_4, t_1), (t_1f_4t_4, t_3), (t_3t_4t_3, f_3), (t_3t_4t_1, f_4),\\ & (t_3t_4t_1, f_2)\},\end{aligned}$$

and

$$\begin{aligned}\bar{\mathscr{D}}^F = \bar{\mathscr{D}}^F_A = \quad & \{(\varepsilon, f_2), (\varepsilon, f_3), (\varepsilon, f_4), (f_1, t_1), (f_1, t_2), (f_1, t_3), (f_1, t_4), (f_1, f_1), (f_1, f_2), (f_1, f_3), (f_1, f_4),\\ & (t_1, f_1), (t_1, f_3), (t_3, f_1), (t_3, f_2), (t_3, f_4), (t_1t_2, f_2), (t_1t_2, f_3), (t_1t_2, f_4), (t_1f_4, f_1),\\ & (t_1f_4, f_2), (t_1f_4, f_4), (t_1f_4, t_1), (t_1f_4, t_2), (t_1f_4, t_3), (t_3t_4, f_2), (t_3t_4, f_3), (t_3t_4, f_4),\\ & (t_1t_2t_1, f_1), (t_1t_2t_1, f_3), (t_1t_2t_3, f_1), (t_1t_2t_3, f_2), (t_1t_2t_3, f_4), (t_1f_4t_4, t_2), (t_1f_4t_4, t_4),\\ & (t_1f_4t_4, f_2), (t_1f_4t_4, f_3), t_1f_4t_4, f_4), (t_3t_4t_3, f_1), (t_3t_4t_3, f_2), (t_3t_4t_3, f_4), (t_3t_4t_1, f_1),\\ & (t_3t_4t_1, f_3)\}.\end{aligned}$$

We find the faulty net system in Figure 13.3(b). By inspection of the faulty net, one can associate the following meaning to the faults: f_1 (resp., f_2, f_3) corresponds to a token loss for agent 1 (resp., 2, 3); f_4 corresponds to a token passage from 2 to 3. ■

13.4 Fault Identification in Case II

In this section we consider Problem 13.3, and make the following additional assumptions.

Assumption (A2): The net contains a single fault, i.e., $q = 1$ thus $T_f = \{f\}$. ■

Assumption (A3): Transition t_f is loop-free, i.e., ${}^\bullet t_f \cap t_f^\bullet = \emptyset$. ■

The idea is to use these assumptions to provide an algebraic characterization of the set of admissible faulty systems. In particular, we show that if an upper bound is given on the number of times the fault transition may fire, then the characterization is linear and the identification problem can be written as a linear IPP.

Note that Assumption (A1) restricts the structure of the faulty Petri net to only include one additional faulty transition; it does not restrict the number of times this faulty transition can fire.

13.4.1 Preliminary Results

Definition 13.9. *Let $\mathscr{L}$ be the prefix-closed language of a fault-free net system, and $\mathscr{L}^F$ be the prefix-closed language of the faulty net system we want to identify.*

We define the following sets:

$$\mathscr{E} = \{(\sigma, t_j) \mid \sigma \in \mathscr{L}, |\sigma| < k, \sigma t_j \in \mathscr{L}\}, \tag{13.4}$$

$$\mathscr{E}^F = \{(\sigma, t_j) \mid \sigma \in \mathscr{L}^F, |\sigma| < k, t_j \in T, \sigma t_j \in \mathscr{L}^F\}, \tag{13.5}$$

$$\tilde{\mathscr{E}}^F = (\mathscr{E}^F \setminus \mathscr{E})|_{\equiv} \tag{13.6}$$

$$\tilde{\mathscr{D}}_B^F = \{(\sigma, t_j) \mid \sigma \in \mathscr{L}^F, |\sigma| < k, t_j \in T, \sigma t_j \notin \mathscr{L}^F\}. \tag{13.7}$$

$$\tilde{\mathscr{D}}^F = \tilde{\mathscr{D}}_B^F|_{\equiv} \tag{13.8}$$

where $\tilde{\mathscr{E}}^F$ and $\tilde{\mathscr{D}}^F$ are the sets containing only one element of each equivalent class for the $\equiv$ relation in Definition 10.2. ■

Proposition 13.10. *Consider a pair $(\sigma, t_j) \in \tilde{\mathscr{E}}^F$. Under assumptions (A2) and (A3), the net $\langle N^F, M_0\rangle$ generates a word $(\sigma t_j)^F \in P^{-1}(\sigma t_j)$ such that $|(\sigma t_j)^F|_{t_f} = \alpha_{\sigma,j}$, iff the following conditions are both verified:*

(a) The net $\langle N^F, M_0\rangle$ generates a word $\sigma^F \in P^{-1}(\sigma)$ with $|\sigma^F|_{t_f} = \alpha_\sigma$.

(b) There exists an integer $\alpha_{\sigma,j}$ such that

$$\begin{cases} M_0 + \alpha_{\sigma,j}(Post^f - Pre^f) + C \cdot y \geq Pre(\cdot, t_j) \\ \alpha_{\sigma,j} \geq \alpha_\sigma \end{cases} \tag{13.9}$$

where $y = \pi(\sigma)$.

Proof. (*If* part) If the net $\langle N^F, M_0\rangle$ generates a word whose projection is σt_j, then there exists a firing sequence

$$M_0[\sigma^F\rangle M[t_f^l\rangle M'[t_j\rangle,$$

where $\sigma^F \in P^{-1}(\sigma)$ and $l \geq 0$ additional occurrences of the unobservable transition t_f may be necessary to enable transition t_j after σ^F has fired. Let $|\sigma^F|_{t_f} = \alpha_\sigma$; then according to the state equation it holds

$$M' = M_0 + C \cdot y + \alpha_\sigma \cdot (Post^f - Pre^f) + l \cdot (Post^f - Pre^f) = M_0 + C \cdot y + \alpha_{\sigma,j} \cdot (Post^f - Pre^f)$$

with $\alpha_{\sigma,j} = \alpha_\sigma + l$ and, since M' enables t_j, we obtain (13.9).

(*Only if* part) Assume condition (a) is verified so that there exists a marking M such that $M_0[\sigma^F\rangle M$. This allows us to rewrite (13.9) as

$$\begin{cases} M + l \cdot (Post^f - Pre^f) \geq Pre(\cdot, t_j) \\ l \geq 0 \end{cases}$$

where $l = \alpha_{\sigma,j} - \alpha_\sigma$. Consider now the subnet obtained from N by removing all transitions except t_f with initial marking M. By assumption (A3) the net is acyclic, hence the fact that equation

$$M + l \cdot (Post^f - Pre^f) \geq Pre(\cdot, t_j) \geq \vec{0}$$

is satisfied implies that there exists a marking M' such that $M[t_f^l\rangle M'$ (see Theorem 3.2). This means that a sequence $(\sigma t_j)^F \in P^{-1}(\sigma t_j)$ is firable in the faulty net with $|(\sigma t_j)^F|_{t_f} = \alpha_{\sigma,j} = \alpha_\sigma + l$. □

Proposition 13.11. *Consider a pair $(\sigma, t_j) \in \tilde{\mathscr{D}}^F$ and let $\bar{\gamma}_\sigma$ be the minimum number of fault transition firings necessary to enable σ, i.e.,*

$$\bar{\gamma}_\sigma = \min_{\sigma^F \in P^{-1}(\sigma)} |\sigma^F|_{t_f}. \tag{13.10}$$

Under assumptions (A2) and (A3) the net $\langle N^F, M_0\rangle$ disables a transition t_j after all sequences $\sigma^F \in P^{-1}(\sigma)$ that are enabled at M_0, iff $\forall\ \gamma \in \mathbb{N}$, with $\gamma \geq \bar{\gamma}_\sigma$, it holds

$$M_0 + C \cdot y + \gamma \cdot (Post^f - Pre^f) \ngeq Pre(\cdot, t_j). \tag{13.11}$$

Proof. Let us show the *if* part. As well known, a transition t is not enabled at a marking $M' \in R(N, M_0)$ iff $M' \ngeq Pre(\cdot, t)$.

Now, if t_j is not enabled after the firing of all sequences $\sigma^F \in P^{-1}(\sigma)$ at M_0, then $\forall\ \gamma^F = |\sigma^F|_{t_f}$ it should be

$$M_0 + C \cdot y + \gamma^F \cdot (Post^f - Pre^f) \ngeq Pre(\cdot, t_j),$$

or, equivalently, equation (13.11) should be verified for all $\gamma \geq \bar{\gamma}_\sigma$, where $\bar{\gamma}_\sigma$ is defined as in equation (13.10).

Let us prove the *only if* part. Since the net is acyclic, the state equation gives conditions that are necessary and sufficient for the reachability (and for non-reachability as well). Thus, if equation (13.11) is satisfied for all $\gamma \geq \bar{\gamma}_\sigma$, then it means that for any marking M such that $M_0[\sigma^F\rangle M$ it is $M \ngeq Pre(\cdot, t_j)$. □

13.4.2 IPP Formulation

Proposition 13.12. *Let us consider Problem 13.3 under assumptions (A1) to (A3), and let*

$$g(Pre^f, Post^f) = \sum_{i=1}^{m} \left[b_i Pre^f(p_i) + c_i Post^f(p_i) \right]$$

be a given linear performance index, where b_i, $c_i \in \mathbb{R}_0^+$.

A solution that is optimal with respect to $g(Pre^f, Post^f)$ can be computed by solving the following nonlinear IPP

$$\begin{cases} \min & g(Pre^f, Post^f) \\ s.t. & \mathscr{G}_m^f(\tilde{\mathscr{E}}^F, \tilde{\mathscr{D}}^F) \\ & M_0\ is\ given \end{cases} \tag{13.12}$$

where

$$\mathscr{G}_m^f(\tilde{\mathscr{E}}^F,\tilde{\mathscr{D}}^F) \triangleq \left\{ \begin{array}{l} \left.\begin{array}{l} M_0+\alpha_{\sigma,j}\cdot(Post^f-Pre^f) \\ \qquad\qquad +C\cdot y \geq Pre(\cdot,t_j) \\ \alpha_{\sigma,j}\in\mathbb{N} \\ \forall\ (\sigma,t_j)\in\tilde{\mathscr{E}}^F \end{array}\right\}(a) \\ \\ \left.\begin{array}{l} -KS^f_{\sigma,j}+M_0+C\cdot y \\ \quad +\gamma\cdot(Post^f-Pre^f)-Pre(\cdot,t_j)\leq -\vec{1}_m \\ \vec{1}^{\,T}S^f_{\sigma,j}\leq m-1 \\ S^f_{\sigma,j}\in\{0,1\}^m \\ \forall(\sigma,t_j)\in\tilde{\mathscr{D}}^F \\ \forall\gamma\in\mathbb{N} \end{array}\right\}(b) \\ \\ \left.\begin{array}{l} Pre^f(p_i)-z_{1,i}\cdot K\leq 0 \\ Post^f(p_i)-z_{2,i}\cdot K\leq 0 \\ z_{1,i}+z_{2,i}=1, \quad i=1,\ldots,m \end{array}\right\}(c) \end{array}\right. \tag{13.13}$$

with K (as usual) being a very large constant.

Proof. We first prove that, under assumptions (A1) to (A3), a net system $\langle N^F, M_0\rangle$ is such that $P(L_k(N^F,M_0)) = \mathscr{L}^F$ if and only if it satisfies the set of algebraic constraints (13.13).

Constraints (a) are *enabling constraints* relative to those sequences that can only be observed when the fault occurs. They trivially follow from Proposition 13.10.

Constraints (b) are *disabling constraints* relative to those sequences that are not enabled even if the fault occurs. They follow from Proposition 13.11 and their equivalence to constraints

$$\left\{ \begin{array}{l} M_0+C\cdot y+\gamma\cdot(Post^f-Pre^f)\not\geq Pre(\cdot,t_j) \\ \forall(\sigma,t_j)\in\tilde{\mathscr{D}}^F \\ \forall\gamma\in\mathbb{N} \end{array}\right.$$

To prove the equivalence between the two sets of constraints we first observe that, if t_j is not enabled at $M_0+C\cdot y+\gamma\cdot(Post^f-Pre^f)$, then there exists at least one place $p\in P$ such that

$$M_0(p)+C(p,\cdot)\cdot y+\gamma\cdot(Post^f(p)-Pre^f(p))\leq Pre(p,t_j)-1. \tag{13.14}$$

This holds for all p such that $S^f_{\sigma,j}(p)=0$. But, being $\vec{1}^{\,T}S^f_{\sigma,j}\leq m-1$, this occurs for at least one place $p\in P$.

Finally, we observe that assuming $\gamma\in\mathbb{N}$ in (b) rather than $\gamma\geq\bar{\gamma}_\sigma$ (see equation (13.11)), introduces no spurious markings. In fact, by definition of $\bar{\gamma}_\sigma$, constraints (b) are redundant for all $\gamma\in[0,\bar{\gamma}_\sigma)$.

Constraints (c) force transition t_f to be loop-free. In fact, they imply that if $Pre^f(p_i)>0$, then $Post^f(p_i)=0$, and viz. □

Remark 13.13. *In Problem 13.3 we assumed that all sequences that are not in $\mathscr{L}^F$ are not observable, namely there exists no sequence of fault transitions that can enable them. In several practical applications it could be of interest to slightly modify the problem statement by assuming that no information can be deduced if a given sequence is not observed (it may be possible that no fault sequence is able to make it firable, but it can also be possible that such faults have not yet occurred even if their firing would have enabled it). In such a case a solution to the identification problem can still be computed by solving the IPP (13.12), provided that the disabling constraints (b) are removed from the set (13.13).*

■

13.4.3 Constraints Linearization

Proposition 13.12 provides a systematic approach to solve Problem 13.3 under assumptions (A1) to (A3). However, some of the constraints that are necessary to characterize the set of admissible solutions are nonlinear.

The nonlinearity can be removed by assigning an upper bound Γ on the number of times the fault transition t_f may fire[1].

In particular, constraint (a) for the generic couple $(\sigma, t_j) \in \tilde{\mathscr{E}}^F$ can be translated into an OR constraint that can be written as a set of $\Gamma+1$ linear constraints

$$\begin{cases} M_0 + Post^f - Pre^f + C\cdot y - Pre(\cdot,t_j) \geq z^1_{\sigma,j}\cdot\vec{K} \\ M_0 + 2\cdot(Post^f - Pre^f) + C\cdot y - Pre(\cdot,t_j) \geq z^2_{\sigma,j}\cdot\vec{K} \\ \quad\vdots \\ M_0 + \Gamma\cdot(Post^f - Pre^f) + C\cdot y - Pre(\cdot,t_j) \geq z^\Gamma_{\sigma,j}\cdot\vec{K} \\ z^1_{\sigma,j} + z^2_{\sigma,j} + \ldots z^\Gamma_{\sigma,j} = \Gamma - 1 \\ z^1_{\sigma,j}, z^2_{\sigma,j}, \ldots, z^\Gamma_{\sigma,j} \in \{0,1\} \end{cases}$$

where, as usual, K is a very large constant (see Chapter 10), and $\vec{K} = K\cdot\vec{1}_m$.

Similarly, if $\gamma \leq \Gamma$, the nonlinear inequality in (b) translates into an AND constraint that can be written as a set of Γ linear constraints

$$\begin{cases} -KS^f_{\sigma,j} + M_0 + C\cdot y + Post^f - Pre^f - Pre(\cdot,t_j) \leq -\vec{1}_m \\ -KS^f_{\sigma,j} + M_0 + C\cdot y + 2\cdot(Post^f - Pre^f) - Pre(\cdot,t_j) \leq -\vec{1}_m \\ \quad\vdots \\ -KS^f_{\sigma,j} + M_0 + C\cdot y + \Gamma\cdot(Post^f - Pre^f) - Pre(\cdot,t_j) \leq -\vec{1}_m \end{cases}$$

[1]Note that a tradeoff should be made while choosing Γ. In fact, a large value of Γ makes the linearization less restrictive but results in a higher computational complexity. We assume here that a tentative value of Γ is initially taken, and it is then increased if the resulting set of linear constraints is infeasible.

13.4.4 Complexity of the Identification Procedure

We now discuss the complexity of the IPP we must solve to identify the faulty system. This complexity is given in terms of the number of constraints and the number of unknowns. Note however that it is well known that an IPP is an NP-hard problem itself.

Let n be the cardinality of T, k the length of the longest string in $\mathscr{L}$, and ν_r (ν'_r), for $r = 0,\ldots,k$, the number of pairs $(\sigma,t_j) \in \tilde{\mathscr{E}}^F$ ($(\sigma,t_j) \in \mathscr{E}^F|_{\equiv}$) of length r.

Then the nonlinear constraint set (13.13) contains

- $m \sum_{r=1}^{k} \nu_r$ constraints of type (a),
- $(m+1) \sum_{r=0}^{k-1} (n \cdot \nu'_r - \nu'_{r+1})$ constraints of type (b),
- $3 \cdot m$ constraints of type (c).

When linearized, the number of constraints (a) and (b) becomes equal to

$$m \cdot (\Gamma + 1) \sum_{r=1}^{k} \nu_r$$

and

$$(m \cdot \Gamma + 1) \sum_{r=0}^{k-1} (n \cdot \nu'_r - \nu'_{r+1})$$

respectively.

The total number of unknowns in the nonlinear IPP is

$$u_{nl} \quad = \quad 2m + \sum_{r=1}^{k} \nu_r \cdot + m \sum_{r=0}^{k-1} (n \cdot \nu'_r - \nu'_{r+1}) + 2m,$$

where the right-side terms are due respectively to the number of pre and post-incidence arcs, the integer variables $\alpha_{\sigma,j}$ in (a), the binary vectors $S^f_{\sigma,j}$ in (b), and the binary variables $z_{i,1}$ and $z_{i,2}$ in (c).

The total number of unknowns in the linear IPP is

$$u_l \quad = \quad 2m + \Gamma \cdot \sum_{r=1}^{k} \nu_r + m \sum_{r=0}^{k-1} (n \cdot \nu'_r - \nu'_{r+1}) + 2m.$$

Note that given values of k and n, it is possible to find a worst case bound for $\rho = \sum_{r=0}^{k-1} (n \cdot \nu'_r - \nu'_{r+1})$. In fact, it holds:

$$\rho = \sum_{r=0}^{k-1} (n \cdot \nu'_r - \nu'_{r+1}) = n \cdot \nu'_0 + (n-1) \cdot \left(\sum_{r=1}^{k-1} \nu'_r \right) - \nu'_k = n + (n-1) \cdot \left(\sum_{r=1}^{k-1} \nu'_r \right) - \nu'_k.$$

This expression is maximized if we assume $\nu'_k = 0$ while all other ν'_r take the largest possible value, i.e., $\nu'_r = n^r$. Hence, we have

$$\rho \leq n + (n-1) \cdot (n + \cdots + n^{k-1}) = n^k,$$

so that the total number of unknowns in the nonlinear IPP in the worst case is

$$u_{\mathrm{nl}_{\mathrm{MAX}}} \leq 4m + \sum_{r=1}^{k} n^r + m \cdot n^k = \mathcal{O}(m\, n^k),$$

and the total number of unknowns in the linear IPP in the worst case is

$$u_{\mathrm{l}_{\mathrm{MAX}}} \leq 4m + \Gamma \cdot \sum_{r=1}^{k} n^r + m \cdot n^k = \mathcal{O}(m\, \Gamma\, n^k),$$

i.e., this problem has exponential complexity with respect to k.

13.4.5 Numerical Examples

In this section we present two examples. First, we provide an example of the procedure previously presented and then we show the problem of acyclicity and then the necessity of the assumption (A3).

Example 13.14. Let us consider the net in Figure 13.4(a) and the two languages $\mathscr{L} = \{\varepsilon, t_1, t_1 t_1, t_1 t_1 t_2\}$ and $\mathscr{L}^F = \{\varepsilon, t_1, t_2, t_1 t_1, t_2 t_1, t_2 t_2, t_1 t_1 t_2, t_2 t_1 t_1, t_2 t_2 t_1, t_2 t_2 t_2\}$ thus $k = 3$. Assume that we want to determine the Petri net system that minimizes the arc weights associated with the fault transition such that $P(L_k(N^F, M_0)) \in \Lambda(\mathscr{L}^F)$. This requires the solution of a linearized IPP of the form (13.13) where

$$\mathscr{E} = \{(\varepsilon, t_1), (t_1, t_1), (t_1 t_1, t_2)\},$$

$$\mathscr{E}^F = \{(\varepsilon, t_1), (\varepsilon, t_2), (t_1, t_1), (t_2, t_1), (t_2, t_2), (t_1 t_1, t_2), (t_2 t_1, t_1), (t_2 t_2, t_1), (t_2 t_2, t_2)\},$$

$$\tilde{\mathscr{E}}^F = \{(\varepsilon, t_2), (t_2, t_1), (t_2, t_2), (t_2 t_1, t_1), (t_2 t_2, t_1), (t_2 t_2, t_2)\},$$

and

$$\tilde{\mathscr{D}}^F = \{(t_1, t_2), (t_1 t_1, t_1), (t_2 t_1, t_2)\}.$$

For $\Gamma = 1$ and $\Gamma = 2$ we get no feasible solution, while for $\Gamma = 3$ we find the faulty net system in Figure 13.4(b), where $Pre^f = [2\ 0]^T$ and $Post^f = [0\ 2]^T$. ■

Example 13.15. Let us consider the net in Figure 13.5(a) and the two languages $\mathscr{L} = \{\varepsilon, t_1, t_1 t_1, t_1 t_1 t_2\}$ and $\mathscr{L}^F = \{\varepsilon, t_1, t_1 t_1, t_1 t_2, t_1 t_1 t_2, t_1 t_2 t_1\}$, thus $k = 3$. We note that a solution for these two languages exists and it is represented by the faulty net system in Figure 13.5(b), but if we apply the identification procedure proposed we obtain that no integer solution is found, even if the constraints relative to the acyclicity of the fault transition, i.e., the constraints (c) in (13.13), are removed. Since our constraints are based on the incidence matrix, the two nets shown in Figure 13.5(b) and Figure 13.5(c) are equivalent as far as our procedure is concerned. The problem is that for the net in Figure 13.5(c) the disabling constraint on the couple (ε, t_2) is not verified, since transition t_2 can be enabled at M_0 after t_f has fired twice. Thus, to the best of our knowledge, there is no remedy to this problem.

■

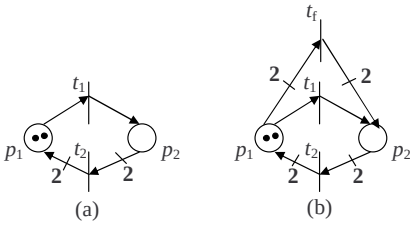


Figure 13.4: (a) The faulty-free net system and (b) the faulty net system identified in Example 13.14.

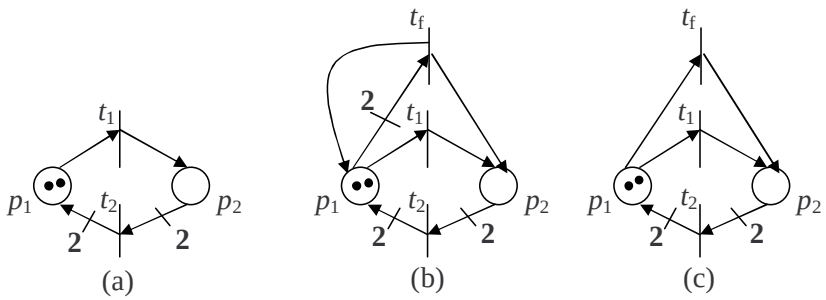


Figure 13.5: (a) The faulty-free net system of Example 13.15, (b) the faulty net system where t_f is not loop-free, (c) the equivalent net of the one represented in (b).

Part IV

Conclusion

Chapter 14

Concluding remarks

In the last part of the thesis conclusion for diagnosis and identification of Petri nets are drawn.

14.1 Concluding remarks for diagnosis

The second part of the thesis has been dedicated to the diagnosis of Petri nets (PNs). One of the goal of this part is to provide some results on diagnosis and diagnosability of labeled PNs. In particular, in Chapter 6 we presented a fault detection approach for discrete event systems using PNs. We assume that some of the transitions of the net are unobservable, including all those transitions that model faulty behaviors. Our diagnosis approach is based on the notion of basis marking and justification, that allow us to characterize the set of markings that are consistent with the actual observation, and the set of unobservable transitions whose firing enable it. Four diagnosis states are defined, each one corresponding to a different degree of alarm. This approach applies to all net systems whose unobservable subnet is acyclic. If the net system is also bounded the proposed approach may be significantly simplified moving the most burdensome part of the procedure off-line, thanks to the construction of a graph, called the basis reachability graph.

In Chapter 7 we extended the diagnosis approach presented in Chapter 6 to labeled PNs, i.e., nets where two or more transitions may share the same label. We extended the notions of basis marking and justifications. As for the unlabeled PNs, we defined four diagnosis states, we gave procedure to compute the actual diagnosis state given the current observation, and we showed that for bounded PNs the most burdensome part of the procedure can be moved off-line defining a particular graph, that we call basis reachability graph.

In Chapter 8 we first gave necessary and sufficient conditions for diagnosability and then we presented a method to test diagnosability. This method, that is inspired by the method proposed by the University of Michigan Group [76, 77], for finite state automata, is based on the analysis of two graphs that depend on the structure of the net, including the faults

model, and the initial marking. The first graph is called basis reachability diagnoser, the second one is called modified basis reachability graph. At the end of the chapter we also made a comparison between our diagnosis procedure and Diagnoser approach.

Finally, Chapter 9 is dedicated to the computational complexity comparison of two diagnostic procedures. The first procedure has been developed by the University of Michigan Group and has been presented in Chapter 4 while the second procedure is the one presented in Chapter 7.

14.2 Concluding remarks for identification

The third part of this thesis is dedicated to the identification of PNs. The goal of this part is to show new results and provide original techniques for the identification of bounded and unbounded PNs. In particular, in Chapter 10 we presented an identification procedure that synthesizing a PN starting from a finite language. First we considered the problem of identifying a free labeled PN system, i.e., all transition labels are distinct. The set of transitions and the number of places is assumed to be known, while the net structure and the initial marking are computed solving an integer programming problem. Then we extended this approach in several ways introducing additional information about the model (structural constraints, conservative components, stationary sequences) or about its initial marking. We also treated the problem of synthesizing a bounded net system starting from an automaton that generates its language. Moreover, we showed how the approach can also be generalized to the case of labeled PNs, where two or more transitions may share the same label. In particular, in this case we imposed that the resulting net system is deterministic. In both cases the identification problem can still be solved via an integer programming problem. In the last part of the chapter, we analyze the computational complexity of our identification approach. In particular, we investigated how the computational time depends on the cardinality of the set of finite length strings that describe the language, and on the chosen performance index. To this aim we considered the language generated by a particular PN system that models a sender-receiver process. Different cases are examined with different number of places and transitions, and thus different languages generated. The numerical simulations we carried out enabled us to conclude that the computational time become prohibitive for languages that are described by a large number of finite length strings, if we want to determine a solution that is optimal with respect to a given performance index. On the contrary, computational times are negligible if we limit to consider any admissible net system, e.g., the first admissible solution computed by CPLEX when solving the optimization problem.

In Chapter 11 we presented an identification procedure for unbounded PNs. In particular we solved the following problem: given an automaton that represents the coverability graph of a PN, determine a PN system whose coverability graph is isomorphic to the automaton. The proposed approach requires solving an integer programming problem whose set of unknowns contains the elements of the pre and post incidence matrices and the initial marking of the net.

The main drawback of our identification procedure is the computational complexity that grows exponentially with the length of the longest string in the input language. We tried to

solve this problem in Chapter 12 showing an identification technique that solves the same problem presented in Chapter 10 using linear programming techniques. The procedure we proposed identifies a net whose number of places is equal to the cardinality of the set of disabling constraints. We provided a criterion to check if the computed solution has a minimal number of places, and, if such is not the case, we discussed two approaches to reduce this number. Note however that the procedure does not ensure to determine a net with minimal number of places. At the end of the chapter we also compared our identification procedure with the theory of regions approach.

Finally, a fault model identification with PNs is presented Chapter 13. It is well known that most of the fault identification problems in the DES literature assume knowledge of the structure of the net system, including the nature (and behavior) of the possible faults. Here we dealt with this problem within the framework of PNs by removing the requirement that the nature (and behavior) of the fault is known. In particular, we devised a way to identify the structure of the faulty transitions of the system given its language. Then, we generalized this procedure to unobservable faults, in which case the structure of the faulty system needs to be recognized from the knowledge of the structure of the fault-free system, and the projection of the faulty system language on the set of non-faulty events, that are assumed to be observable.

Bibliography

[1] D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, 1982. [cited at p. 5, 17]

[2] E. Badouel, L. Bernardinello, and P. Darondeau. Polynomial algorithms for the synthesis of bounded nets. *Proceedings of CAAP'95, Lecture Notes in Computer Science*, 915:647–679, 1995. [cited at p. 5, 18, 19, 40]

[3] E. Badouel and P. Darondeau. On the synthesis of general Petri nets. Technical Report 3025, 1996. [cited at p. 5, 18, 19, 45]

[4] E. Badouel and P. Darondeau. Theory of regions. *Lecture Notes in Computer Science: Lectures on Petri Nets I: Basic Models*, 1491:529–586, 1998. [cited at p. 5, 18, 19, 40]

[5] F. Balduzzi, A. Giua, and G. Menga. First–order hybrid Petri nets: a model for optimization and control. *IEEE Trans. on Automation*, 16:382–399, 2000. [cited at p. 15]

[6] F. Basile, P. Chiacchio, and G. De Tommasi. An efficient approach for on-line diagnosis of discrete event systems. In *Proc. 15th IEEE Mediterranean Conference on Control and Automation*, Athens, Greece, June 2007. [cited at p. 5, 16]

[7] F. Basile, P. Chiacchio, and G. De Tommasi. An efficient approach for online diagnosis of discrete event systems. *IEEE Trans. on Automatic Control*, 2008, in press. [cited at p. 5, 16]

[8] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics and constraints. *Automatica*, 35(3):407–429, 1999. [cited at p. 209]

[9] A. Benveniste, E. Fabre, S. Haar, and C. Jard. Diagnosis of asynchronous discrete event systems: A net unfolding approach. *IEEE Trans. on Automatic Control*, 48(5):714–727, 2003. [cited at p. 5, 16]

[10] R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Synthesis of Petri nets from infinite partial languages. In *ACSD08: Proceedings of the 8th International Conference on Application of Concurrency to System Design*, pages 170–179, Xian, China, 2008. [cited at p. 5, 18, 19]

[11] R.K. Boel and G. Jiroveanu. Distributed contextual diagnosis for very large systems. In *Proc. IFAC WODES'04: 7th Work. on Discrete Event Systems*, pages 343–348, September 2004. [cited at p. 52]

[12] R.K. Boel and J.H. van Schuppen. Decentralized failure diagnosis for discrete-event systems with costly communication between diagnosers. In *Proc. WODES'02: 6th Work. on Discrete Event Systems*, pages 175–181, October 2002. [cited at p. 5, 14]

[13] T. Bourdeaud'huy and P. Yim. Synthèse de réseaux de Petri à partir d'exigences. In *Actes de la 5me conf. francophone de Modélisation et Simulation*, pages 413–420, Nantes, France, September 2004. [cited at p. 5, 18]

[14] R.E. Burkard and F. Rendl. Lexicographic bottleneck problems. *Operations Research Letters*, 10:303–308, 1991. [cited at p. 126]

[15] M.P. Cabasino. Diagnosis of discrete event systems using automata and Petri nets. Master's thesis, Dep. Electric and Electronic Engineering, University of Cagliari, Cagliari, Italy, 2005. (In Italian). [cited at p. 110]

[16] M.P. Cabasino, A. Giua, C. N. Hadjicostis, and C. Seatzu. Fault model identification with Petri nets. In *Proc. IFAC WODES'08: 9th Work. on Discrete Event Systems*, pages 455–461, May 2008. [cited at p. 6, 9]

[17] M.P. Cabasino, A. Giua, and C. Seatzu. Computational complexity analysis of a Petri net identification procedure. In *2006 Int. Symposium on Nonlinear Theory and its Applications*, September 2006. [cited at p. 9]

[18] M.P. Cabasino, A. Giua, and C. Seatzu. Identification of deterministic Petri nets. In *Proc. IFAC WODES'06: 8th Work. on Discrete Event Systems*, Ann-Arbor, MI, USA, July 2006. [cited at p. 9]

[19] M.P. Cabasino, A. Giua, and C. Seatzu. Identification of unbounded Petri nets from their converability graph. In *Proc. 45th IEEE Conf. on Decision and Control*, San Diego, California USA, December 2006. [cited at p. 9]

[20] M.P. Cabasino, A. Giua, and C. Seatzu. Identification of Petri nets from samples of their languages. *Discrete Events Dynamical Systems*, 17(4):447–474, 2007. [cited at p. 9, 164]

[21] M.P. Cabasino, A. Giua, and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 2008. Preliminary accepted. [cited at p. 8]

[22] M.P. Cabasino, A. Giua, and C. Seatzu. Linear programming techniques for the identification of Place/Transition nets. In *Proc. 47th IEEE Conf. on Decision and Control*, Cancun, Mexico, December 2008. [cited at p. 9]

[23] M.P. Cabasino, A. Giua, and C. Seatzu. Diagnosability of bounded Petri nets. In *Proc. 48th IEEE Conf. on Decision and Control*, December 2009. Submitted. [cited at p. 9]

[24] M.P. Cabasino, A. Giua, and C. Seatzu. Diagnosis of discrete event systems using labeled Petri nets. In *Proc. 2nd IFAC Workshop on Dependable Control of Discrete Systems (Bari, Italy)*, June 2009. Submitted. [cited at p. 8]

[25] M.P. Cabasino, A. Giua, and C. Seatzu. Identification of unbounded Petri nets from their coverability graph. 2009. Preliminary accepted. [cited at p. 9]

[26] B. Caillaud. Synet : A synthesizer of distributable bounded Petri-nets from finite automata. *http://www.irisa.fr/s4/tools/synet/*. 2002. [cited at p. 19]

[27] J. Carmona, J. Cortadella, and A. Kishinevsky. A region-based algorithm for discovering Petri nets from event logs. In *BPM08: Proceedings of the 6th International Conference on Business Process Management*, pages 358–373, Berlin, Heidelberg, 2008. [cited at p. 20]

[28] J. Carmona, J. Cortadella, A. Kishinevsky, L. Lavagno, A. Kondratyev, and A. Yakovlev. A symbolic algorithm for the synthesis of bounded Petri nets. In *ICATPN08: Proceedings International Conferences on Application and Theory of Petri Nets and Other Models of Concurrency*, pages 92–111, Berlin, Heidelberg, 2008. [cited at p. 5, 18, 19]

[29] S. Lafortune C.G. Cassandras. *Introduction to discrete event systems, Second Edition*. Springer, 2007. [cited at p. 30, 31, 32, 33, 35, 37, 86, 205]

[30] S.L. Chung. Diagnosing pn-based models with partial observable transitions. *International Journal of Computer Integrated Manufacturing*, 12 (2):158–169, 2005. [cited at p. 5, 17]

[31] D. Corona, A. Giua, and C. Seatzu. Marking estimation of Petri nets with silent transitions. *IEEE Trans. on Automatic Control*, 52(9):1695–1699, September 2007. [cited at p. 25, 52, 57]

[32] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri nets from finite transition systems. *IEEE Transactions on Computers*, 47(8):859–882, 1998. [cited at p. 5, 18, 19]

[33] P. Dague, P. Deves, P. Luciani, and P. Taillibert. Analog systems diagnosis. *Readings in Model Based Diagnosis*, 1992. [cited at p. 5, 13]

[34] R. Davis and W. Hamscher. Model based reasoning: Troubleshooting. *Readings in Model Based Diagnosis*, 1992. [cited at p. 5, 13]

[35] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete-event systems. *Discrete Events Dynamical Systems*, 10(1):33–86, January 2000. [cited at p. 5, 14]

[36] W.F. (IV) Doran and D.B. Wales. The partition algebra revisited. *Journal of Algebra*, 231(1):265–330, September 2000. [cited at p. 166]

[37] M. Dotoli, M.P. Fanti, and A.M. Mangini. Fault detection of discrete event systems using Petri nets and integer linear programming. In *Proc. of 17th IFAC World Congress*, Seoul, Korea, July 2008. [cited at p. 5, 16]

[38] M. Dotoli, M.P. Fanti, and A.M. Mangini. Real time identification of discrete event systems using Petri nets. *Automatica*, 44(5):1209–1219, 2008. [cited at p. 5, 18]

[39] M. Dotoli, M.P. Fanti, and A.M. Mangini. Fault monitoring of discrete event systems by first order hybrid Petri nets. In *Workshop on Petri Nets and Agile Manufacturing, a satellite event of the 29th Int. Conf. on Application and Theory of Petri Nets and Other Models of Concurrency*, Xi'an, China, June 2008 (to appear). [cited at p. 5, 15]

[40] D. Dvorak and B. Kuipers. Model based monitoring of dynamic systems. *Readings in Model Based Diagnosis*, 1992. [cited at p. 5, 13]

[41] A. Ehrenfeucht and G. Rozenberg. Partial (set) 2-structures. part i,ii. *Acta Informaticag*, 27(4):315–368, 1989. [cited at p. 18]

[42] P. Frank. Fault diagnosis in dynamic systems using analytical and knowledge based redundancy - a survey and some new results. *Automatica*, 26, 1990. [cited at p. 5, 12]

[43] F. Garcia-Valles and J.M Colom. Implicit places in net systems. In *Proc. of the 8th Int. Work. on Petri Nets and Performance Models*, pages 104 – 113, Zaragoza, Spain, 1999. [cited at p. 171]

[44] S. Genc and S. Lafortune. Distributed diagnosis of place-bordered Petri nets. *IEEE Trans. on Automation Science and Engineering*, 4(2):206–219, 2007. [cited at p. 5, 16]

[45] A. Ghaffari, N. Rezg, and X. Xie. Design of a live and maximally permissive Petri net controller using the theory of regions. *IEEE Transactions on Robotics and Automation*, 19(1):137–142, 2003. [cited at p. 6, 39, 40, 41]

[46] M. Ghazel, A. Togueni, and M. Bigang. A monitoring approach for discrete events systems based on a time Petri net model. In *Proc. of 16th IFAC World Congress*, Prague, Czech Republic, July 2005. [cited at p. 5, 15]

[47] E. Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978. [cited at p. 5, 17]

[48] C.N. Hadjicostis and G.C. Veghese. Monitoring discrete event systems using Petri net embeddings. *Lecture Notes in Computer Science*, 1639:188–207, 1999. [cited at p. 5, 15]

[49] K. Hiraishi. Construction of a class of safe Petri nets by presenting firing sequences. *Lecture Notes in Computer Science*. [cited at p. 5, 18]

[50] http://bode.diee.unica.it/dehs/CDC06. [cited at p. 156]

[51] S. Jiang and R. Kumar. Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *IEEE Trans. on Automatic Control*, 49(6):934–945, June 2004. [cited at p. 5, 14]

[52] G. Jiroveanu and R.K. Boel. Contextual analysis of Petri nets for distributed applications. In *16th Int. Symp. on Mathematical Theory of Networks and Systems (Leuven, Belgium)*, July 2004. [cited at p. 52]

[53] S. Lafortune. Executables of the umdes-lib software library for solaris, linux, mac and windows are publicly available. *http://www.eecs.umich.edu/umdes/toolboxes.html*. [cited at p. 107, 109]

[54] S. Lai, D. Nessi, M.P. Cabasino, A. Giua, and C. Seatzu. A comparison between two diagnostic tools based on automata and Petri nets. In *Proc. IFAC WODES'08: 9th Work. on Discrete Event Systems*, pages 144–149, May 2008. [cited at p. 9, 103, 104]

[55] S. Lapp and G. Powers. Computer aided synthesis of fault trees. *IEEE Trans. Reliability*, 26(1):2–13, 1977. [cited at p. 5, 12]

[56] W.S. Lee, D.L. Grosh, F.A. Tillman, and C.H. Lie. Fault tree analysis, methods, and applications - A review. *IEEE Trans. Reliability*, 34:194–203, 1985. [cited at p. 5, 12]

[57] D. Lefebvre and C. Delherm. Diagnosis of DES with Petri net models. *IEEE Trans. on Automation Science and Engineering*, 4(1):114–118, 2007. [cited at p. 5, 15]

[58] L.X. Li, Y. Ru, and C.N. Hadjicostis. Least-cost firing sequence estimation in labeled Petri nets. In *Proc. 45th IEEE Conf. on Decision and Control*, San Diego, California USA, December 2006. [cited at p. 21]

[59] F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems*, 4(2):197–212, 1994. [cited at p. 5, 13]

[60] F. Lin, J. Markee, , and B. Rado. Design and test of mixed signal circuits: a discrete event approach. In *Proc. 32rd IEEE Conf. on Decision and Control*, pages 246–251, 1993. [cited at p. 5, 13]

[61] L. Lin and Y. Jiang. The computation of hitting sets: review and new algorithms. *Information Processing Letters*, 86:177–184, 2003. [cited at p. 168]

[62] R. Lorenz, R. Bergenthum, J. Desel, and S. Mauser. Synthesis of Petri nets from finite partial languages. In *ACSD07: Proceedings of the 7th International Conference on Application of Concurrency to System Design*, pages 157–166, Washington, DC, USA, 2007. [cited at p. 5, 18, 19]

[63] R. Lorenz and G. Juhás. Towards synthesis of Petri nets from scenarios. In *Proc. of 27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency*, pages 302–321, 2006. [cited at p. 5, 18, 19]

[64] R. Lorenz, G. Juhás, and S. Mauser. How to synthesize nets from languages – a survey. In *Proc. 2007 Winter Simulation Conference*, Washington DC, USA, December 2007. [cited at p. 5, 6, 18, 19, 39, 42]

[65] J. Lunze and J. Schroder. Sensor and actuator fault diagnosis of systems with discrete inputs and outputs. 34(3):1096–1107, April 2004. [cited at p. 5, 14]

[66] J. Martinez and M. Silva. A simple and fast algorithm to obtain all invariants of a generalized Petri net. In *Informatik-Fachberichte 52: Application and Theory of Petri Nets.*, pages 301–310. Springer-Verlag, 1982. [cited at p. 53]

[67] M.E. Meda-Campaña and E. López-Mellado. Incremental synthesis of Petri net models for identification of discrete event systems. In *Proc. 41th IEEE Conf. on Decision and Control*, pages 805–810, Las Vegas, Nevada USA, December 2002. [cited at p. 5, 18]

[68] M.E. Meda-Campaña and E. López-Mellado. Required event sequences for identification of discrete event systems. In *Proc. 42th IEEE Conf. on Decision and Control*, pages 3778–3783, Maui, Hawaii, USA, December 2003. [cited at p. 5, 18]

[69] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989. [cited at p. 140]

[70] C.A. Petri. *Kommunication mit Automaten*. PhD thesis, Institut fur Instrumentelle Mathematik, Schriften des IIM, No. 3, Bonn, Germany, 1962. [cited at p. 4, 24]

[71] J. Prock. A new tecnique for fault detection using Petri nets. *Automatica*, 27(2):239–245, 1991. [cited at p. 5, 15]

[72] A. Rey. Diagnosis of Petri nets using the Basis Reachability Graph. Master's thesis, Dep. Electric and Electronic Engineering, University of Cagliari, Cagliari, Italy, 2007. (In Italian). [cited at p. 107, 109]

[73] Y. Ru and C. N. Hadjicostis. State estimation in discrete event systems modeled by labeled Petri nets. In *Proc. 45th IEEE Conf. on Decision and Control*, San Diego, California USA, December 2006. [cited at p. 21]

[74] A. Ramirez-Treviño E. Ruiz-Beltràn, I. Rivera-Rangel, and E. Lopez-Mellado. Online fault diagnosis of discrete event systems. A Petri net-based approach. *IEEE Trans. on Automation Science and Engineering*, 4(1):31–39, 2007. [cited at p. 5, 15]

[75] M. Sampath, S. Lafortune, and D. Teneketzis. Active diagnosis of discrete-event systems. *IEEE Trans. on Automatic Control*, 43(7):908–929, July 1998. [cited at p. 5, 13]

[76] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. on Automatic Control*, 40 (9):1555–1575, 1995. [cited at p. 5, 6, 13, 14, 17, 29, 30, 35, 36, 189]

[77] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Failure diagnosis using discrete-event models. *IEEE Trans. Control Systems Technology*, 4(2):105–124, 1996. [cited at p. 5, 6, 13, 17, 29, 30, 189]

[78] W. T. Scherer and C. C. White. A survey of expert systems for equipment maintenance and diagnostics. *Fault Detection and Reliability: Knowledge Based & Other Approaches*, 1987. [cited at p. 5, 12]

[79] R.S. Sreenivas. On minimal representations of Petri net languages. In *Proc. WODES'02: 6th Work. on Discrete Event Systems*, pages 237–242, Zaragoza, Spain, October 2002. [cited at p. 131, 132]

[80] R.S. Sreenivas. On minimal representations of Petri net languages. *IEEE Trans. on Automatic Control*, 51(5):799–804, 2006. [cited at p. 5, 21]

[81] V.S. Sreenivas and M.A. Jafari. Fault detection and monitoring using time Petri nets. *IEEE Trans. Systems, Man and Cybernetics*, 23(4):1155–1162, 1993. [cited at p. 5, 15]

[82] T. Ushio, L. Onishi, and K. Okuda. Fault detection based on Petri net models with faulty behaviors. In *Proc. SMC'98: IEEE Int. Conf. on Systems, Man, and Cybernetics (San Diego, CA, USA)*, pages 113–118, October 1998. [cited at p. 5, 16]

[83] W.M.P. van der Aalst, B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters. Workflow mining: A survey of issues and approaches. *IEEE Trans. on Data & Knowledge Engineering*, 47(2):237–267, 2003. [cited at p. 20]

[84] W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Trans. on Data & Knowledge Engineering*, 16(9):1128–1142, 2004. [cited at p. 20]

[85] J.M.E.M. van der Werf, B.F. van Dongen, B. F., C. A. Hurkens, and A. Serebrenik. Process discovery using integer linear programming. In *PETRI NETS '08: Proceedings of the 29th international conference on Applications and Theory of Petri Nets*, pages 368–387, 2008. [cited at p. 20]

[86] J.H. van Schuppen. System theory for system identification. *Journal of Econometrics*, 118(1-2):313–339, January-February 2004. [cited at p. 5, 17]

[87] N. Viswanadham. Control systems : Reliability. *Systems and Control Encyclopedia: Theory, Technology, Applications*, 1987. [cited at p. 5, 12]

[88] N. Viswanadham and T. L. Johnson. Fault detection and diagnosis of automated manufacturing systems. In *Proc. 27th IEEE Conf. on Decision and Control*, pages 2301–2306, Austin, Texas, December 1988. [cited at p. 5, 12]

[89] R. D. Vries. An automated methodology for generating a fault tree. *IEEE Trans. Reliability*, 39(1):76–86, 1990. [cited at p. 5, 12]

[90] Y. Wen and M. Jeng. Diagnosability analysis based on T-invariants of Petri nets. In *Networking, Sensing and Control, 2005. Proceedings, 2005 IEEE.*, pages 371– 376, March 2005. [cited at p. 17]

[91] Y. Wen, C. Li, and M. Jeng. A polynomial algorithm for checking diagnosability of Petri nets. In *Proc. SMC'05: IEEE Int. Conf. on Systems, Man, and Cybernetics*, pages 2542– 2547, October 2005. [cited at p. 17]

[92] A. S. Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 12, 1976. [cited at p. 5, 12]

[93] Y. Wu and C.N. Hadjicostis. Algebraic approaches for fault identification in discrete-event systems. *IEEE Trans. Robotics and Automation*, 50(12):2048–2053, 2005. [cited at p. 5, 15]

[94] S. Hashtrudi Zad, R.H. Kwong, and W.M. Wonham. Fault diagnosis in discrete-event systems: framework and model reduction. *IEEE Trans. on Automatic Control*, 48(7):1199–1212, July 2003. [cited at p. 5, 14, 111]

List of Publications Related to the Thesis

Published papers

Journal papers

- M.P. Cabasino, A. Giua, C. Seatzu, *Identification of Petri nets from knowledge of their language.* Discrete Event Dynamic Systems, Vol. 17, No. 4, pp. 447-474, Dec 2007. (Relation to Chapter 10)
- M.P. Cabasino, A. Giua, C. Seatzu, *Fault detection for discrete event systems using Petri nets with unobservable transitions.* Automatica, Preliminary accepted. (Relation to Chapter 6)
- M.P. Cabasino, A. Giua, C. Seatzu, *Identification of unbounded Petri nets from their coverability graph.* IEEE Transactions on Automatic Control, Conditionally accepted. (Relation to Chapter 11)

Conference papers

- M.P. Cabasino, A. Giua and C. Seatzu, *Diagnosability of bounded Petri nets,* 48th IEEE Conf. on Decision and Control,(Submitted). (Relation to Chapter 8)
- M.P. Cabasino, A. Giua and C. Seatzu, *Diagnosis of discrete event systems using labeled Petri nets,* Dependable Control of Discrete Systems,(Submitted). (Relation to Chapter 7)
- M.P. Cabasino, A. Giua and C. Seatzu, *Linear Programming Techniques for the Identification of Place/Transition Nets,* in 47th IEEE Conf. on Decision and Control, (Cancun, Mexico), December 2008. (Relation to Chapter 12)
- M.P. Cabasino, A. Giua, C. N. Hadjicostis and C. Seatzu, *Fault Model Identification with Petri Nets,* in WODES'08: 9th Int. Workshop on Discrete Event Systems, (Göteborg, Sweden), May 2008. (Relation to Chapter 13)
- S. Lai, D. Nessi, M.P. Cabasino, A. Giua and C. Seatzu, *A Comparison Between Two Diagnostic Tools Based on Automata and Petri Nets,* in WODES'08: 9th Int. Workshop on Discrete Event Systems, (Göteborg, Sweden), May 2008. (Relation to Chapter 9)

- M.P. Cabasino, A. Giua and C. Seatzu, *Identification of unbounded Petri nets from their coverability graph,* in 45th IEEE Conf. on Decision and Control, (San Diego, CA, USA), Dec 2006. (Relation to Chapter 11)
- M.P. Cabasino, A. Giua and C. Seatzu, *Computational complexity analysis of a Petri net identification procedure,* in 2006 Int. Symposium on Nonlinear Theory and its Applications, (Bologna, Italy), Sep 2006. (Relation to Chapter 10)
- M.P. Cabasino, A. Giua and C. Seatzu, *Identification of deterministic Petri nets,* in WODES'06: 8th Int. Workshop on Discrete Event Systems, Ann Arbor, MI, USA), Jul 2006. (Relation to Chapter 10)

Appendices

Appendix A

Language notation and definitions

In the following we give a brief description of languages. For more details we refer to [29].

A language is a set of words.

Definition A.1. *A* language *L defined over an alphabet E is a set of words defined over this alphabet. The* cardinality *of the language is the number of words that it contains and is denoted as $|L|$.* ■

A language can be represented in several ways.

Example A.2. *Given an alphabet $E = \{a, b, c\}$, let us consider the following languages:*

$$L_1 = \{a, b, ac, ba\},\ L_2 = \{\varepsilon, a\},\ L_3 = \{w \in E^* \mid |w| \geq 3\},\ L_4 = \emptyset,\ L_5 = E^*.$$

Language L_1 contains three words, thus $|L_1| = 3$. Language L_2 contains two words, one of them is the empty word. Language L_3 is composed by all words whose length is greater than or equal to 3. Language L_4 contains no word. Language L_5 contains all words defined over E. ■

Note that the cardinality of a language L can be either null (as for L_4 in the above example), or finite (as for L_1, L_2 in the above example), or infinite (as for L_3, L_5 in the above example). Moreover a language can be described either listing all its words (as for L_1, L_2 in the above example) or using a set theory notation (as for L_3, L_4, L_5 in the above example).

Operations on languages

The usual set operations, such as union, intersection, difference, and complement with respect to E^*, are applicable to languages since languages are sets. Here we recall some operations on languages:

- *Concatenation*: Let $L_1, L_2 \subseteq E^*$. We define *concatenation* of L_1 and L_2 the language

$$L_1 L_2 = \{w = w_1 w_2 \in E^* \ :\ w_1 \in L_1 \text{ and } w_2 \in L_2\}.$$

i.e., L_1L_2 is composed by all words composed by the concatenation of a word in L_1 with a word in L_2.

- *Prefix-closure*: Let $L \subseteq E^*$. The *prefix-closure of L* is the language

$$\bar{L} = \{u \in E^* \,:\, \exists w \in L \,:\, (u \preceq w)\},$$

i.e., the prefix-closure of L is the language denoted by $\bar{L}$ and consisting in all the prefixes of all words in L. In general $\bar{L} \subseteq L$. A language L is said to be *prefix-closed* if $L = \bar{L}$. Thus language is prefix-closure if any prefix of any word in L is also an element of L.

- *Kleene-closure*: Let $L \subseteq E^*$. The *Kleene-closure of L* is the language

$$L^* = \{\varepsilon\} \cup L \cup LL \cup LLL \cup \cdots$$

i.e., is the set of all words obtained by the concatenation of a finite (but possibly arbitrarily large) number of elements of L.

Example A.3. *Let $E = \{a, b, c\}$ and consider the three languages $L_1 = \{\varepsilon, a\}$, $L_2 = \{a, b, ab\}$ and $L_3 = \{\varepsilon, a, aca\}$. It holds:*

$$\begin{array}{rcl}
L_1L_2 & = & \{a, b, aa, ab, aab\} \\
L_1L_3 & = & \{\varepsilon, a, aa, aca, aaca\} \\
\bar{L}_1 & = & L_1 \\
\bar{L}_2 & = & \{\varepsilon, a, b, ab\} \\
\bar{L}_3 & = & \{\varepsilon, a, ac, aca\} \\
L_1^* & = & \{\varepsilon, a, aa, aaa, \ldots\} \\
L_2^* & = & \{\varepsilon, a, b, aa, ab, aab, \ldots\} \\
L_3^* & = & \{\varepsilon, a, aca, aa, aaca, acaa, acaaca, \ldots\}.
\end{array}$$

■

Appendix B

Equivalence relations and classes

Let us introduce the defintions of equivalence relation and equivalence class. Let A be a set. A *relation* R tells for any two members, say a and b, of S whether a is in that relation to b. For example, if S is a set of numbers one relation is $=$. For any two numbers a and b one can determine if $a = b$ or not.

Abstractly considered, any relation on the set S is a function from the set of ordered pairs from S, called the Cartesian product $S \times S$, to the set {*true, false*}. The relation is usually identified with the pairs such that the function value equals true.

An equivalence relation R is a special type of relation.

Definition B.1. *Let A be a set and $\sim$ be a binary relation on A. $\sim$ is called an* equivalence relation *if and only if for all elements $a, b, c \in A$ hold the following properties:*

- Reflexivity*: $a \sim a$*
- Symmetry*: if $a \sim b$ then $b \sim a$*
- Transitivity*: if $a \sim b$ and $b \sim c$ then $a \sim c$.*

The equivalence class *of a under $\sim$, denoted $[a]$, is defined as $[a] = \{b \in A \mid a \sim b\}$.* ■

In simple words, relation ~ on A is an equivalence relation if it is reflexive, symmetric and transitive. An example of such is equality on a set. One might think of equivalence as a way to glob together elements that can be considered the same relative to a property. That is elements become indistinguishable relative to the relation. For example in arithmetic we do not think twice about $1/2$ and $2/4$ as having the same value but they are different objects. In geometry, similarity of triangles is an equivalence relation. A right angled triangle with legs of length 3 and 4 and hypotenuse of length 5 is not the same as one with lengths 6, 8 and 10. Yet, we think of them as equivalent because the ratios of the corresponding sides are the same.

For each $a \in A$, we define the *equivalency class containing a* to be the set of those elements which are equivalent to a. We denote this set by $[a]$. The equivalence classes have some interesting properties.

- No equivalence class is empty since $a \in [a]$.

- Two equivalence classes are either equal or disjoint. In fact, a and b are equivalent if and only if $[a] = [b]$.
- Each element of A belongs to some equivalence class (in fact $a \in [a]$).
- The union of all the equivalence classes is A. We say that the equivalence classes partition A. A partition of A is a collection of non empty disjoint subsets of A and whose union is A. Thus the equivalence classes of a relation are a partition. Each equivalence relation on a set partitions the set into its equivalence classes but also for each partition of the set there is an equivalence relation whose equivalence classes are the sets in the partition. As an example, if $A = \{1,2,3\}$ one partition is $\{1,2\}$, $\{3\}$. Consequently, there is an equivalence relation on A whose equivalence classes are these two subsets.

One problem is how should we refer to the equivalence classes? As we noted above $[a] = [b]$ whenever a and b are equivalent. What we do is choose one and only one element of each equivalence class to represent the class. That element is called a *class representative*. The set of representatives is called the set of equivalence class representatives. There are usually many such ways to construct this set of representatives.

Appendix C

Logical constraints transformation

In the following we provide an efficient technique to convert logical *or* constraints into linear algebraic constraints, that is inspired by the work of Bemporad and Morari [8]. In particular, we consider two different cases: *inequality* constraints and *equality* constraints.

Inequality constraints

Let us consider the following constraint:

$$\bigvee_{i=1}^{r} \vec{a}_i \leq \vec{0}_n \tag{C.1}$$

where $\vec{a}_i \in \mathbb{R}^n$, $i = 1, \ldots, r$, and $\bigvee$ denotes the logical *or* operator.

Equation (C.1) can be rewritten in terms of linear algebraic constraints as:

$$\begin{cases} \vec{a}_1 \leq z_1 \cdot \vec{K} \\ \vdots \\ \vec{a}_r \leq z_r \cdot \vec{K} \\ z_1 + \ldots + z_r = r - 1 \\ z_1, \ldots, z_r \in \{0, 1\} \end{cases} \tag{C.2}$$

where $\vec{K}$ is any constant vector in $\mathbb{R}^n$ that satisfies the following relation

$$K_j > \max_{i \in \{1,\ldots,r\}} a_i(j), \quad j = 1, \ldots, n.$$

In fact, if $z_i = 0$ then the i-th constraint is active, while if $z_i = 1$ it is trivially verified, thus resulting in a redundant constraint. Moreover, the condition $z_1 + \ldots + z_r = r - 1$ implies that one and only one z_i is equal to zero, i.e., only one constraint is active. This means that $\vec{a}_i \leq \vec{0}_n$ for one i, while no condition is imposed for the other i's (in such cases the corresponding constraints may either be violated or satisfied). Obviously, analogous considerations can be repeated if the $\leq$ constraints in (C.1) are replaced by $\geq$ constraints.

Equality constraints

Let us now consider the constraint

$$\bigvee_{i=1}^{r} \vec{a}_i = \vec{b}_i \tag{C.3}$$

where $\vec{a}_i, \vec{b}_i \in \mathbb{R}^n$, $i = 1, \ldots, r$.

Equation (C.3) can be rewritten in terms of linear algebraic constraints as:

$$\begin{cases} \vec{a}_1 - \vec{b}_1 \leq z_1 \cdot \vec{K} \\ \vec{a}_1 - \vec{b}_1 \geq -z_1 \cdot \vec{K} \\ \vdots \\ \vec{a}_r - \vec{b}_r \leq z_r \cdot \vec{K} \\ \vec{a}_r - \vec{b}_r \geq -z_r \cdot \vec{K} \\ z_1 + \ldots + z_r = r - 1 \\ z_1, \ldots, z_r \in \{0, 1\} \end{cases} \tag{C.4}$$

where $\vec{K}$ is any constant vector in $\mathbb{R}^n$ such that

$$K_j > \max_{i \in \{1,\ldots,r\}} |a_i(j) - b_i(j)|, \quad j = 1, \ldots, n.$$

Repeating a similar reasoning as in the previous case, we can immediately observe that, if $z_i = 0$ then

$$\begin{cases} \vec{a}_i - \vec{b}_i \leq \vec{0}_n \\ \vec{a}_i - \vec{b}_i \geq \vec{0}_n \end{cases} \quad \Rightarrow \quad \vec{a}_i = \vec{b}_i.$$

On the contrary, if $z_i = 1$ then

$$\begin{cases} \vec{a}_i - \vec{b}_i \leq \vec{K} \\ \vec{a}_i - \vec{b}_i \geq -\vec{K} \end{cases}$$

that are trivially verified, i.e., they are redundant constraints. Finally, the condition on the sum of z_i's imposes that one constraint is active, i.e., $\vec{a}_i = \vec{b}_i$ for at least one $i \in \{1, \ldots, r\}$.