Contributions to Model-Based Diagnosis of Discrete-Event Systems

Abderraouf Boussif

N^o *d'ordre*: | * | * | *

Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux
Université de Lille1 - Sciences et Technologies

THÈSE

présentée en vue d'obtenir le grade de

DOCTEUR

en

Automatique, Génie Informatique, Traitement du Signal et des Images

par

Abderraouf BOUSSIF

Doctorat délivré par l'université de Lille1 - Sciences et Technologies

Titre de la thèse :

Contributions to Model-Based Diagnosis of Discrete-Event Systems

Contributions au Diagnostic à Base de Modèles des Systèmes à Évènements Discrets

Soutenance prévue le 12 Décembre 2016 devant le jury d'examen :

Président	Pr Jean-Jacques Lesage	LURPA, ENS Paris-Saclay
Rapporteur	Pr Dimitri Lefebvre	GREAH, Univ. Le Havre
Rapporteur	Pr Michel Combacau	LAAS, Univ. Toulouse 3
Examinateur	DR El-Miloudi El-Koursi	COSYS/ESTAS - IFSTTAR Lille
Examinateur	Pr Armand Toguyeni	CRIStAL-ECL Lille
Examinatrice	Pr Narjes Ben Rajeb	LIP2 - INSAT Tunis
Examinateur	Dr-HDR Kais Klai	LIPN, Univ. Paris 13
Directeur de thèse	Dr-HDR Mohamed Ghazel	COSYS/ESTAS - IFSTTAR Lille

Thèse préparée au Laboratoire d'Évaluation des Systèmes de Transports Automatisés et de leur Sécurité
IFSTTAR, COSYS/ESTAS, Villeneuve d'Ascq
École Doctorale Sciences pour l'ingénieur ED 072 - PRES Université Lille Nord de France

Contributions to Model-Based Diagnosis of Discrete-Event Systems

Abstract : This PhD thesis deals with fault diagnosis of discrete-event systems modeled as finite state automata with some extensions to bounded Petri net models. The developed contributions can be classified regarding two pioneering approaches from the literature: the diagnoser-based technique and the twin-plant based technique.

Regarding the diagnoser-based technique, we propose a new diagnoser variant with some interesting features that allow us to separately track the normal and the faulty traces directly in the diagnoser. On the basis of the developed diagnoser, we reformulate the necessary and sufficient condition for diagnosability of permanent faults and we propose a systematic procedure for checking such a condition without building any intermediate model. An on-the-fly algorithm, for simultaneously constructing the diagnoser and verifying diagnosability is developed. The algorithm aims to generate as less state-space as possible, particularly when the system is non-diagnosable, which improves the memory/time consumption. The developed diagnoser is then extended to deal with fault diagnosis of intermittent faults. Various notions of diagnosability are addressed and necessary and sufficient conditions are formulated on the basis of the new diagnoser structure.

A Hybrid version (in the sense of combining enumerative and symbolic representations) of the diagnoser variant is established in order to deal with fault diagnosis of labeled bounded Petri nets. The main idea consists in using the symbolic representation (Binary Decision Diagrams) to compact and handle the Petri net markings inside the diagnoser nodes and using explicit representation for the (observable) transitions that link the diagnoser nodes. Such a combination serves, on one hand, to reduce the memory required for constructing the diagnoser and, on the other hand, to easily explore the diagnoser paths. The developed approaches are implemented in dedicated tools and evaluated through benchmarks with respect to the reference approaches in the domain.

Regarding twin-plant based technique, the first contribution consists in elaborating a model-checking framework, that extends the Cimatti's work, for the actual verification of various diagnosability concepts pertaining to permanent failures based on the twin-plant structure. The main idea is to reformulate and express the diagnosability issues as temporal logics and then to tackle them using the model-checking engines. The second contribution is pertaining to the diagnosis of intermittent faults, and consists in addressing various notions of diagnosability while establishing their necessary and sufficient conditions on the basis of the twin-plant structure.

Keywords : Fault diagnosis, Discrete-event systems, Intermittent faults, Permanent faults, Finite state automata, Labeled Petri nets, Model-checking.

Contributions au Diagnostic à Base de Modèles des Systèmes à Évènements Discrets

Résumé : L'objet de cette thèse porte sur le diagnostic des systèmes à évènements discrets modélisés par des automates à états finis, avec une extension vers les réseaux de Petri bornés et étiquetés. Les différentes contributions de ce travail peuvent être présentées selon deux volets, au regard des approches pionnières existantes dans la littérature: le diagnostic à base de diagnostiqueur, et le diagnostic à base de twin-plant.

Sur le premier volet, i.e., le diagnostic à base de diagnostiqueur, nous proposons une variante du diagnostiqueur avec une nouvelle structure qui permet de suivre séparément les traces normales et les traces fautives dans le diagnostiqueur. A partir de cette structure, nous formulons une condition nécessaire et suffisante pour l'analyse de la diagnosticabilité des fautes permanentes et nous développons une procédure systématique pour l'analyse de la diagnosticabilité sans construire de modèle intermédiaire. La procédure de vérification est basée sur un algorithme à-la-volée pour construire le diagnostiqueur et analyser la diagnosticabilité en parallèle. Cet algorithme permet, d'une part de réduire l'espace-d'état généré, principalement pour les systèmes non-diagosticable, et d'autre part, d'améliorer nettement les besoins en mémoire et en temps de calcul. Cette approche est par la suite étendue pour traiter le cas des fautes intermittentes où plusieurs propriétés relatives à la diagnosticabilité sont considérées et les conditions nécessaires et suffisantes correspondantes sont établies.

Comme extension aux modèles réseaux de Petri bornés et étiquetés, nous avons développé une autre contribution qui porte sur l'établissement d'une version hybride de notre diagnostiqueur, dans le sens ou on combine les représentations énumérative et symbolique. L'idée principale de ce travail est de représenter symboliquement (sous forme de BDD) l'ensemble du marquages dans chaque noeud de diagnostiqueur, tout en gardant une représentation explicite pour les arcs reliant les noeuds. Une telle représentation permet d'une part, de réduire la mémoire utilisée pour la construction du diagnostiqueur, et d'autre part, de faciliter l'exploration des traces d'exécution du système. Cette technique a été implémentée dans un outil nommé SOG-Diag tool et évaluée à travers des benchmarks.

Sur le deuxième volet, i.e., le diagnostic à base de twin-plant, une première contribution porte sur la mise en oeuvre de nouvelles techniques pour l'analyse de diagnostic. Elle consiste notamment à développer un cadre de vérification par model-checking de la diagnosticabilité. Ainsi, la performance des techniques de model-checking et des outils associés peut être exploitée pour traiter des questions de diagnostic. Une deuxième contribution sur ce volet consiste à étendre ce cadre pour l'analyse des propriétés relatives à la diagnosticabilité des fautes intermittents en se basant sur des conditions nécessaires et suffisantes que nous avons développées.

Mots clés : Diagnostic des fautes, Systèmes à événements discrets, Fautes intermittentes, Fautes permanentes, Automates à états finis, Réseaux de Petri étiquetés, Vérification à base de modèles.

Acknowledgment

The work that resulted in this thesis could not have been accomplished without several persons' assistance, support, and encouragement!

First and foremost, my special appreciation and thanks are due to my supervisor Dr Mohamed Ghazel for his day-to-day constant support and guidance during the past three years. Without his help, this work would not be possible.

I am deeply indebted and thoroughly grateful to Professor Jean-Jacques Lesage for agreeing to be the chair of my thesis committee, for accepting me as a master student in LURPA Laboratory four years ago, and his support and encouragement.

I would like to thank those who agreed to be the referees of this thesis and allocated their valuable time in order to evaluate the quality of this work: Pr Dimitri Lefebvre, Pr Michel Combacau, DR El-Miloudi El-Koursi, Pr Armand Toguyeni, Pr Narjes Ben Rajeb, and Dr Kais Klai, for their examination of the report and their very helpful comments and suggestions.

I would also like to express my thanks to various members of IFSTTAR - ESTAS and LEOST Laboratories. I have learned so much from all of you!

I will fail in my duty if I do not acknowledge some of my friends who helped me a lot during the past three years. I mention Baisi (*I always enjoy discussing with you*!), Ci, Christophe, Antoine, Matthieu, Lucas, Rafik, Hassanein, Nadjah, Ayoub, Foued, Sohaib, WARSI-MHRSL group, and many others whose names only by lack of memory I failed to include in this list.

I cannot find words to express my gratitude to my mother, my father, my sisters & my brothers, for all their never-ending support ... Your prayers for me were what sustained me thus far!

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

Thank you very much, everyone!

A. BOUSSIF

Lille - December 12^{th} 2016.

To the memory of my grandmother Saliha!
To my mother & my father
my sisters & my brothers

You write so beautifully ...
the inside of your mind must be a terrible place!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sommaire

Preamble

This dissertation presents a synthesis of research work, which was carried out as a fruit of my Ph.D. (2013 – 2016) accomplished within the COSYS/ESTAS team (*Evaluation of Automated Transport Systems and their Safety*) at IFSTTAR, the French Institute of Science and Technology for Transport, Development and Networks (*Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux*). The Ph.D. thesis has been prepared within the doctoral school SPI (*Engineering Sciences*) at Lille 1 university. This thesis was supervised by Dr. Mohamed Ghazel, senior researcher with IFSTTAR – COSYS/ESTAS.

The work presented in this dissertation contributes to fault diagnosis of discrete-event systems (DESs) modeled by finite state automata with some extension to Petri net models. The intention of this research is twofold: (i) provide some new approaches to enhance the fault diagnosis of DESs and (ii) bring into play the formal methods, namely the model-checking techniques, for the actual verification of diagnosis properties.

1.1 General Context

Nowadays, due to the pressing demand in terms of productivity, safety, comfort and services, many modern systems depend on computers for their correct operation. Of greatest concern, of course, are safety-critical systems [Storey, 1996] since the consequences of failure occurring in such systems can be serious. There are many applications that have traditionally been considered as safety-critical, but the scope of the definition has to be expanded as computer systems continue spreading through many areas that affect our lives [Knight, 2002]. Examples of non-traditional safety-critical systems are railway-embedded control, banking and financial systems, electricity generation and distribution, telecommunications. All of these applications are extensively computerized, and computer failures can lead to serious safety concerns and extensive loss of service with consequent disruption of normal activities. Moreover, recent technological advances have resulted in increasingly complex systems that raise considerable challenges in terms of design, safety-analysis and verification/validation.

Ensuring performance, comfort and safety goals in large-scale and safety-critical systems is a tough and challenging issue. In particular, complexity is one of the most difficult aspect that must be tackled to make theoretical methodologies applicable to real industrial applications. In fact, such large-scale and complex systems are vulnerable to faults that can cause undesired behavior and, as a consequence, damages to technical parts of the plant, to personnel, or to the environment. It is plain that this also impacts the timely delivery of quality products/services.

In the railway domain for instance, embedded control/command systems are nowadays endowed with a multitude of sensing components that monitor different aspects in the system behavior and generate huge volumes of data. The expansion of such sensing/monitoring systems within the train control modules is due to the need of surveillance means regarding the increasing complexity of train control systems. Actually, this increasing complexity is in line with the diversification of the implemented functions on-board trains, the increasing demand in terms of safety, comfort and competitiveness (punctuality, cost, etc.). One of the major challenges that face the automated monitoring task in railway applications is to efficiently handle the important volumes of the generated data.

The work undertaken within this Ph.D thesis falls in the general scope that integrates research activities related to the safety analysis of guided transportation systems. Although the main contributions developed here are theoretical, various application will be tackled in the framework of this general scope.

The purpose of automated monitoring and fault diagnosis of large-scale and safety-critical systems is to improve their reliability and availability and to increase their opera-

tivity. In this context and to fulfill the performance, comfort and safety goals, developing effective monitoring techniques becomes essential starting from the design phase of the system. In particular, having efficient tools for monitoring and diagnosis is of great interest since this prevents, or at least minimizes, the failure-related consequences, especially in safety-critical systems. Fortunately, the need for these automated mechanisms and tools to deal with the automated monitoring and fault diagnosis of systems is well understood and appreciated, by decision-makers, industry and academia. A great deal of research effort has been and is being spent in the design and development of such automated monitoring means.

From the practical point of view, industrial end-users are interested in the following questions:

- *what kind of faults can be detected/diagnosed? what cannot be detected/diagnosed?*
- *what is the benefit in case of a successful detection/diagnosis? how much is the cost (hardware, software, manpower, etc.) of realizing monitoring tasks?*
- *for a given monitoring and diagnosis task, which method/approach is most suitable?*

All these issues and aspects are addressed in research works that deal with the automated monitoring and fault diagnosis.

One of the major distinctions in the approaches for automated monitoring and fault diagnosis is whether or not explicit models are used, and what type of models are used. When some models of the system behavior are used as a basis for fault detection and diagnosis, this is often referred to as *model-based diagnosis*. The concept of model-based diagnosis poses a promising approach to the systematic capture and analysis of diagnostic knowledge. The use of explicit models of a system's structure and behavior allows us to trace deviant system observations directly to causing components. Establishing the models then becomes a significant part of the application development effort. Moreover, in different engineering domains, there has been a shift towards model-based techniques. In many model-based diagnosis approaches, a model of the system which may depict both the normal and faulty behavior is considered. The faulty behavior is caused by the occurrence of some failures -or category of failures which are not directly observable. Thus, their occurrences must be inferred from the observable part of the behavior.

From the theoretical point of view and at a high level of abstraction, Discrete Event Systems (DESs) are more suitable for modeling, monitoring and model-based diagnosing complex dynamic systems, because of the convenience of their associated models algorithms [Cassandras and Lafortune, 2009, Lin, 1994]. DESs are a qualitative abstraction of continuous dynamic systems that has been receiving increasing attention from both

Artificial Intelligence and Automatic and Control communities. In fact, various classes of man-made systems can be viewed as DESs, provided that their evolution is driven by the occurrences of events and described by discrete state-space.

Fault diagnosis consists of many challenging problems to investigate. However, one of the main issues that must be firstly addressed is the diagnosability investigation. In fact, the diagnosability issue is relatively recent in the research field, and principally two distinct and parallel research communities have been working along the lines of the model-based diagnosis context: the FDI community and the DX community that have evolved in the fields of Automatic Control and Artificial Intelligence [Cordier et al., 2004]. Analyzing diagnosability of a system intends to determine whether or not any predetermined failure or class of failures can be accurately detected and identified within a finite delay following its occurrence. Then comes the issue of developing the diagnoser (called also fault detection and identification engine or simply, the diagnosis system), which performs the diagnosis task. Roughly speaking, the diagnoser role is to carry the state estimation (of the system) online and to emit verdicts regarding the behavior of the monitored system (normal of faulty behavior) based on the delivered partially observations by the sensors map. A crucial issue that the diagnosis activity has to deal with is partial observability on the system behavior. This is due to the fact that it is often quite expensive and technically difficult to observe all changes in a complex system. Therefore, it becomes vital to develop efficient and robust methods able to fill this partial observability on the system behavior.

1.2 Problems Statement

The DES diagnosis problem is basically concerned with determining which faults, if any, explain a given observed system behavior. Fault diagnosis is therefore closely related to the problem of state estimation [Zaytoon and Lafortune, 2013]. The DES diagnosis is often discussed through two main issues: diagnosability analysis and online diagnosis. Diagnosability refers to the capacity of providing a precise diagnosis verdict. Thus, the intention of analyzing diagnosability of a system is to determine accurately whether any predetermined failure can be detected and identified within a finite delay following its occurrence. Online diagnosis consists in inferring, online, the occurrence of predetermined faults from the observed behavior of the system [Sampath et al., 1995].

Various techniques have been developed to deal with DESs diagnosis, as will be detailed in the first part of this dissertation (Part I). Moreover, several improvement in terms of efficiency have been made and numerous issues still need to be tackled as discussed in what follows.

Diagnoser-based approaches

The pioneering work which has set the foundations of DESs fault diagnosis is [Sampath et al., 1995, Sampath et al., 1996] where a formal definition of diagnosability was introduced. Such a work provided a necessary and sufficient condition for diagnosability as well as a systematic approach, based on the so-called *diagnoser*, with the aim of analyzing diagnosability and performing the online diagnosis. However, the combinatorial explosion problem is inherent to the defined approach and the state-space of the diagnoser is, in the worst case, exponential w.r.t. the size of the model state-space.

Improvements in terms of complexity have been introduced in [Jiang and Huang, 2001, Yoo and Lafortune, 2002b, Moreira et al., 2011, Qiu and Kumar, 2006, Li et al., 2015a]. The basis idea was to build an intermediate structure called twin-plant/verifier, by performing a parallel composition of the system model with itself. The diagnosability issue can then be addressed by analyzing every pair of executions that share the same observation. Such a task is performed using polynomial-time algorithms. Nevertheless, these approaches deal only with diagnosability analysis and do not consider online diagnosis. Moreover, comparative studies show that the diagnoser-based approach is more efficient for analyzing diagnosability of some kinds of system models than the twin-plant/verifier based approaches [Liu, 2014]. Consequently, the diagnoser-based approach remains a principal technique to deal with both diagnosability analysis and online diagnosis.

The main issues related to the diagnoser-based approaches can be outlined as follows:

1. the synthesis of the diagnoser is performed with an exponential complexity in the original model state-space, and double-exponential in the cardinality of the fault classes. This consequently hampers the scalability of the approach.

2. the approach is based on the analysis of two graphs. The first graph is a non-deterministic observer (called pre-diagnoser, or generator), while the second one is a deterministic automaton, called diagnoser (or equivalently, generator/diagnoser [Sampath et al., 1995], MBRG/BRD [Cabasino et al., 2009b], FM-graph/FM-set graph [Liu, 2014], etc.).

3. the double-checking procedure, which consists of one verification on the diagnoser (i.e., the existence of F-*uncertain* cycles) and the other on the generator or the pre-diagnoser (i.e., checking whether the F-*uncertain* cycle is an F-*indeterminate* one or not). In fact, in general such a double-checking procedure highly increases the verification time.

Intermittent Fault Diagnosis

In a major part of the literature regarding the DESs diagnosis, faults are typically assumed to be permanent, i.e., *faults cause irremediable deviations form the normal behavior of the system*. However, experience with fault diagnosis of industrial systems shows that intermittent faults (i.e., *faults provoke only temporary deviations from the normal behavior of the system*) are predominant and are among the most challenging kinds of faults to detect and isolate [Fromherz et al., 2004]. Indeed, according to [Shen et al., 2016], intermittent faults exist in many systems, including those ranging from small components to the complex modules. The frequent occurrence of intermittent faults can bring on serious troubles and result in high safety risk and important maintenance costs and delays. In the late 1960s, Hardie [Ball and Hardie, 1969, Hardie and Suhocki, 1967] indicated that intermittent faults comprised over 30% of pre-delivery failures and almost 90% of field failures in computer systems. Moreover, intermittent faults bring on many maintenance problems, such as False Alarms (FAs), Can Not Duplicate (CND) and No Fault Found (NFF) [Sorensen et al., 1994]. This last one (i.e., NFF) has been identified as the source of the highest cost in aerospace maintenance. For instance, the annual NFF exchange cost of the F-16 avionics boxes due to intermittent faults was over $ 20,000,000 [Steadman et al., 2005, Steadman et al., 2008].

On the contrary to fault diagnosis of permanent faults, a few DES-based frameworks have been proposed to handle intermittent faults. In addition, the DES-based fault diagnosis methodologies which deal with permanent failures are unfortunately no longer suitable for the analysis of intermittent faults, since the case of intermittent faults shows some subtle configurations compared to the case of permanent failures. The same problem is also encountered with various methodologies developed in the field of model-based reasoning in Artificial Intelligence, as witnessed in [Contant, 2005].

Practical Verification of Diagnosability

In the last two decades, many research works have been concerned with the development of new models, new properties, new algorithms and efficient solutions for fault diagnosis of DES [Zaytoon and Lafortune, 2013]. Unfortunately, most of the approaches developed are oriented to the definition of theoretical frameworks and do not address the problems related to the practical application of the approaches developed. Moreover, such approaches propose ad-hoc algorithms for the actual verification of diagnosability which are implemented in academic tools. Hence, the practical implementation of the developed DES diagnosis techniques is an issue that still needs exploration.

The works in [Cimatti et al., 2003, Pecheur et al., 2002], propose a practical frame-

work for the formal verification of the diagnosability using model-checking techniques. In fact, the authors attempted to bring forward an effective framework for the analysis of diagnosability that can be practically applied in the development process of diagnosis systems. The main advantage of the proposed approach is that the actual verification is performed using the model-checking techniques. Fortunately, a wide range of powerful and optimized model-checkers have been developed in the formal verification community and successfully used for the verification/validation of large scale industrial systems. Such tools can be used for the verification of diagnosability using the practical framework proposed in [Cimatti et al., 2003, Pecheur et al., 2002]. However, the authors in [Cimatti et al., 2003, Pecheur et al., 2002] have only discussed a variant of the diagnosability that shows some restrictions.

1.3 Main Contributions

This dissertation focuses on fault diagnosis of discrete-event systems modeled by finite state automata with some extensions to bounded Petri net models. The main contributions are all discussed in the second and third part (Part II and Part III) of the thesis and can be summarized in the following items.

i.) *the development of a new diagnoser variant approach for diagnosis of discrete-event systems modeled by automata with permanent faults with an extension for fault diagnosis of Petri net models;*

ii.) *the investigation of various notions of diagnosability of intermittent faults in both the diagnoser-based approach (in Point i.) and twin-plant based approach in* [Jiang and Huang, 2001]*;*

iii.) *the elaboration of a model-checking framework for the practical verification of various diagnosability concepts on the basis of the work of Cimatti et al.* [Cimatti et al., 2003].

In what follows, we give the detailed features of our contributions:

A new diagnoser-based approach

— We develop a diagnoser with a new structure for representing the nodes. Such a structure explicitly separates between the normal and the faulty states in each node. This feature allows us to separately track the normal and the faulty traces directly in the diagnoser. Moreover, the diagnoser is constructed directly from the original model, without needing to construct any intermediate model;

— On the basis of the proposed structure of the diagnoser, a sufficient condition for the undiagnosability of the model is proposed. Such a condition is used for the on-the-fly verification of diagnosability;

— we prove for an $F-uncertain$ cycle in the diagnoser, there exists at least one fault-free cycle in the original model that share the same observation. Therefore, for an $F-uncertain$ cycle in the diagnoser to be an $F-indeterminate$ one, it is sufficient to check that at least one faulty cycle which shares the same observation with the $F-uncertain$ cycle exists in the original model;

— we establish a systematic procedure for checking the necessary and sufficient condition for diagnosabilily without returning to any intermediate model to check if an $F-$uncertain cycle corresponds to two cycles, a faulty one and a non-faulty one;

— we develop an on-the-fly algorithm, for simultaneously constructing the diagnoser and verifying diagnosability. The algorithm aims to generate as less state-space as possible, particularly when the system is undiagnosable, which improves the memory/time consumption;

— the developed approach is implemented and some experimentation and comparative studies (with existing tools for analyzing diagnosability) are conducted in order to assess the effectiveness and the scalability of the approach developed;

— we establish a hybrid version (in the sense of combining enumerative and symbolic representations) of the diagnoser variant we develop, in order to deal with fault diagnosis of bounded Petri nets. The maim idea consists in:

  – using binary decision diagrams (BDDs) to compact and handle the diagnoser nodes, which serves to reduce the memory consumption;

  – using an explicit representation for the (observable) transitions that link the diagnoser nodes. Such a representation allows for an easy exploration of the diagnoser paths.

— we develop a dedicated tool implementing the proposed approach, in order to assess the efficiency and the scalability of the approach. Some experimentations have been conducted through a PN benchmark. The obtained results are discussed with respect to a reference approach for fault diagnosis of LPNs, called MBRG/BRD technique [Cabasino et al., 2009a].

Intermittent fault diagnosis

— we extend the diagnoser-based approach introduced above, in order to deal with intermittent faults. Therefore, various notions of diagnosability are addressed, and necessary and sufficient conditions are formulated on the basis of the new diagnoser structure;

— we also address the issue of diagnosability of intermittent faults using the twin-plant approach. In fact, we develop new necessary and sufficient conditions for the various notions of diagnosability discussed in this dissertation.

Practical verification of diagnosability

Cimatti et al. [Cimatti et al., 2003] introduced an approach to deal with the diagnosability analysis in a model-checking framework. On the basis of this approach, we develop the following:

— we extend the approach in [Cimatti et al., 2003] in order to deal with practical version of diagnosability, namely, $K/K_{min}-$diagnosability properties.

— the necessary and sufficient condition for diagnosability of the intermittent faults on the basis of the twin-plant approach (introduced above) are expressed in temporal logics (LTL/CTL) for the actual verification using model-checking;

— in order to show the applicability of this approach, some experimentation are performed on the basis of two benchmarks which depicts the concept of permanent and intermittent faults respectively.

1.4 Organization and Structure of the Dissertation

This dissertation is divided into four parts:

PART I: in this part, we give a literature review regarding fault diagnosis, particularity, the DESs diagnosis. The part is composed of two chapters:

- *Chapter 2:* we review the concept of fault diagnosis and the model-based diagnosis techniques;
- *Chapter 3:* we give a brief summary of the background on fault diagnosis of DESs modeled by finite state automata.

PART II: in this part, we discuss our contributions regarding the diagnoser-based approach. This part is composed of three chapters:

- *Chapter 4:* we propose a new version of the well-known diagnoser-approach. It consists in separating normal states from faulty ones in each diagnoser node. Such a distinction serves to more efficiently track the faulty and fault-free traces in the diagnoser paths. On the basis of various features that characterize this new diagnoser, we develop a systematic procedure for checking the necessary and sufficient condition for diagnosability;

- *Chapter 5:* we extend the diagnoser-based approach, introduced in the previous chapter, in order to deal with intermittent fault diagnosis. First, we discuss two ways for modeling the intermittent faults in finite state automata framework. Then, various definitions of diagnosability from the SED literature are revisited and necessary and sufficient conditions for checking such properties are derived on the basis of the diagnoser structure. A systematic procedure for checking such conditions without needing any intermediate model is proposed;

- *Chapter 6:* we present an improvement of the diagnoser-based approach introduced in Chapter 4 while dealing with Petri net models. It consists in building a symbolic diagnoser called *Symbolic Reachability diagnoser* for both analyzing diagnosability and performing the online diagnosis of bounded labeled Petri nets.

PART III: in this part, we provide our contributions regarding the twin-plant approach. This part is composed of tow chapters:

- *Chapter 7:* we discuss the practical verification of permanent fault diagnosability in a model- checking framework. The diagnosability condition is expressed using CTL formula while the twin-plant structure is transformed into a Kripke structure and, therefore, diagnosability is investigated as a model-checking problem. Reformulation of $K/K_{min}-$diagnosability are also discussed;

- *Chapter 8:* we discuss the diagnosability analysis of intermittent fault using the twin-plant based approach. The various notions of diagnosability introduced in Chapter 5 are carried out, in this chapter. Necessary and sufficient condition for each property is firstly established and then reformulated (if possible) as a model-chcking problem.

PART IV: in this part, we provide conclusion remarks regarding the dissertation and we draw future research directions.

Publications

Journals

IJCCBS | **A. Boussif**, M. Ghazel and K. Klai. Fault Diagnosis of Discrete-Event Systems Based on the Symbolic Observation Graph. *The International Journal of Critical Computer-Based Systems.* (submitted)
(This paper is among a selection of the best conference papers in VeCOS'15 for publication in a special issue of IJCCBS)

IEEE-TASE | **A. Boussif** and M. Ghazel. Analyzing Various Notions of Diagnosability of Intermittent Faults in Discrete Event Systems. *IEEE Transactions on Automation Science and Engineering.* (submitted)
Review comments have been received for the first version submitted, where some modifications are asked. A second version is submitted on the light of the review comments.

IEEE-TAC | **A. Boussif** and M. Ghazel. A New Variant of the Diagnoser-Based Approach for Fault Diagnosis of Discrete-Event Systems. *IEEE Transactions on Automatic Control.* (submitted)

IEEE-SMC | **A. Boussif**, M. Ghazel and K. Klai. Fault Diagnosis of Bounded Labeled Petri Nets Using a Semi-Symbolic Diagnoser. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (submitted)

Conferences & Proceedings

DCDS'15 | **A. Boussif** and M. Ghazel. Diagnosability Analysis of Input/Output Discrete-Event Systems Using Model-Checking. *The* 5^{th} *International Workshop on Dependable Control of Discrete Systems*, Cancun, Mexico. |Mai 27-29, 2015

MSR'15 | **A. Boussif** and M. Ghazel. Une Approche par Décomposition de Modèles pour l'Analyse de la Diagnosticabilité des Systèmes à Évènements Discrets par Model-Checking. *10e Colloque sur la Modélisation des Systèmes Réactifs* Inria Nancy-Grand Est, France. |Nov 18-20, 2015.

VeCoS'15	**A. Boussif**, M. Ghazel and K. Klai. Combining Enumerative and Symbolic Techniques for Diagnosis of Discrete-Event Systems. *The* 9^{th} *International Workshop on Verification and Evaluation of Computer and Communication Systems (VeCos'15)*, Bucharest, Romania; |Sept. 10-11, 2015
WoDES'16	**A. Boussif**, B. Liu and M. Ghazel. A Twin-Plant Based Approach for Diagnosability Analysis of Intermittent Failures. *The* 12^{th} *International Workshop on Discrete-Event Systems (Wodes'16)*, Xi'an, China; | Mai 30 - Juin 1, 2016
ICPHM'16	**A. Boussif** and M. Ghazel. Intermittent Fault Diagnosis of Industrial Systems in the Model-Checking Framework. *IEEE International Conference on Prognostics and Health Management*, Ottawa, Canada; |Juin 20-22, 2016
VeCoS'16	**A. Boussif** and M. Ghazel. Using Model-Checking Techniques for Diagnosability Analysis of Intermittent Faults - A Railway Case-Study. *The* 10^{th} *International Workshop on Verification and Evaluation of Computer and Communication Systems*, Tunis, Tunisia; |Oct. 06-07, 2016
ACC'17	**A. Boussif** and M. Ghazel. A Diagnoser-Based Approach for Intermittent Fault Diagnosis of Discrete-Event Systems. *The 2017 American Control Conference*, Seattle, WA, USA, 2017; (submitted)
IFAC-WG'17	**A. Boussif**, M. Ghazel and K. Klai. DPN-SOG: A Software tool for Fault Diagnosis of Labeled Petri Nets Using Semi-Symbolic Diagnoser. 20^{th} *IFAC World Congress*, Toulouse, France, 2017; (submitted)

Part I

LITERATURE REVIEW

Chapter 2

Model-Based Diagnosis

Sommaire

2.1 Fault Diagnosis

According to Webster's Dictionary the meaning of the term diagnosis is as follows:

> **diagnosis**
> Etymology: New Latin, from Greek *diagnOsis*, from *diagignOskein*
> to distinguish, from *dia-* + *gignOskein* that means 'knowing the difference'
> Date: circa 1681.
> *Definition 1:* (from medicine) The art or act of recognizing the presence of disease from its signs or symptoms, and deciding as to its character; also, the decision arrived at.
> *Definition 2:* the act or process of identifying the nature or cause of some phenomenon, especially the abnormal behavior of an animal or artifactual device; as, diagnosis of a vibration in an automobile; diagnosis of the failure of a sales campaign; diagnosis of a computer malfunction.

Hereafter, we recall the definitions of some terms and concepts that are often used in this thesis.

2.1.1 Terminology in the Area of Fault Diagnosis

As a step towards a unified terminology, the IFAC Technical Committee SAFEPROCESS has suggested preliminary definitions of some terms in the field of fault diagnosis. Such a terminology was first published in [Isermann and Ballé, 1997, Isermann, 2006].

- **Fault:** an unpermitted deviation of at least one characteristic property or parameter of the system form the acceptable / usual/standard condition;
- **Failure:** a permanent interruption of a system's ability to perform a required function under specified operating conditions;
- **Fault detection:** determination of the faults present in a system and the time of detection;
- **Fault isolation:** determination of the kind, location and time of detection of a fault. Follows fault detection;
- **Fault identification:** determination of the size and time-variant behavior of a fault. Follows fault isolation;

- **Monitoring:** a continuous real-time task of determining the conditions of a physical system, by recording information, recognizing and indicating anomalies in the behavior;
- **Supervision:** monitoring a physical system and taking appropriate actions to maintain the operation in the case of faults;
- **Fault diagnosis:** determination of the kind, size, location and time of detection of a fault. Follows fault detection. Includes fault isolation and identification;

The ranking of these functions according to the importance is obviously subjective, however, the fault detection is an absolute must for any practical system and isolation is almost equally important. Fault identification, on the other hand, whilst undoubtedly helpful, may not be essential if no reconfiguration action is involved. Hence, fault diagnosis is very often considered as fault detection and isolation/identification, abbreviated as FDI, in the literature [Chen and Patton, 2012].

2.2 Fault Diagnosis Approaches

Approaches dealing with fault diagnosis can be classified into four main classes:

2.2.1 Expert Systems and Knowledge-Based Diagnosis

A popular method for diagnosis and supervision of complex systems has been the use of expert systems, often in conjunction with fault tree structures. The terms expert system and knowledge-based system are often used synonymously. Expert systems are eminently well suited for systems that are difficult to model, with complex interactions between and within components [Scherer and White III, 1989, Tzafestas and Watanabe, 1990]. Expert systems found broad application in fault diagnosis from their early stages because an expert system simulates human reasoning about a problem domain, performs reasoning over representations of human knowledge and solves problems using heuristic knowledge rather than precisely formulated relationships, in forms that reflect more accurately the nature of most human knowledge. Domain experts have heuristic knowledge of the system and of how symptoms relate to faults. In traditional expert systems, this knowledge is represented in a rule-base and used in conjunction with an inference engine [Venkatasubramanian et al., 2003].

This heuristic approach has several drawbacks. Acquiring knowledge from experts is difficult and time consuming, and for new systems 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. Furthermore, it is difficult to validate an expert system.

2.2.2 Data-Driven Based Diagnosis

In contrast to model-based approach, which requires reliable a priori quantitative or qualitative knowledge about the process, the data-driven approach makes use of this information from the huge amount of process history data. Thus, data-driven approaches work without models, or with only simplified ones [Venkatasubramanian et al., 2003]. Since most of the data-driven approaches assume that the process data have certain probability density functions, they are sometimes also called statistical process monitoring methods. The univariate control chart may be the earliest statistical approach based on a priori knowledge of process measurement distributions [Shewhart and Deming, 1939] and has been used for quality control in earlier industrial applications.

The main drawback of these techniques is the need to data for training. Data from the systems are not available until they have been built. Only during the testing phase of the physical system is it possible to collect useful data. This, however, drastically shortens the time available for the tuning of the diagnosis system, and makes it extremely difficult to validate the diagnoser when designing the system [Gario, 2016].

2.2.3 Fault-Tree Based Diagnosis

The most widely used scheme for alarm analysis, especially in the process control industry, is based on fault-trees [De Vries, 1990, Lapp and Powers, 1977, Lee et al., 1985] The basic idea behind fault tree approaches is that a failure can trigger other failures or events in the system and this can be traced back to the root cause [Hamscher, 1992]. A fault-tree graphically represents a cause-effect relationship among the failures in the system. The root of a fault tree, the so-called '*top*' event, is a system failure. The leaves of the tree are possibly contributing atomic events or basic faults, and inner nodes are AND- and OR-type. Sets of events that trigger the top event are computed using cut sets and minimal cut sets [Huang, 2003].

Fault-tree construction is laborious and error prone, and much work has been done on computer assisted and automatic fault tree construction [Lapp and Powers, 1977, Elliott, 1994]. Moreover, a fault tree is used to analyze a single fault event, and that one and only one event can be analyzed during a single fault tree. These drawbacks in fact limit the applicability of fault-tree in practice.

2.2.4 Model-Based Diagnosis

Model-based diagnosis techniques (shortly MBD) [Kleer, 1992, Sampath et al., 1995, Roth et al., 2012, Lin, 1994, Reiter, 1987, Lamperti and Zanella, 2013, Darwiche and Provan, 1996, Frank, 1996, Schneider et al., 2012, Brusoni et al., 1998, Venkatasubramanian et al., 2003, Hamscher, 1992, Isermann and Ballé, 1997] have been remarkably developed since the 80s and their efficiency for detecting faults has been demonstrated by a great number of applications in industrial processes and automatic control systems. The MBD approaches are based on a description of the system's behavior, called the '*system model*'. This model is generally provided by the system designer, and describes at least the normal behavior of the system. Better results are achieved when the model describes the behavior of the system under predefined faults, or when the model accounts for the system structure, i.e., the components that intervene in the system behavior. The basic idea behind the MBD is to compare observations of the real system with the predictions from a model. Therefore, the occurrence of a fault is captured by discrepancies between the actual observed behavior and the behavior that is predicted by the model. Fault localization then rests on interlining the model parts (i.e., components) that are involved in each of the detected discrepancies.

As discussed in [Cordier et al., 2004], two distinct and parallel research communities have been using the MBD approaches. The FDI (for Fault Detection and Isolation/Identification) community has evolved in the Automatic Control field from the seventies and uses techniques from control theory and statistical decision theory. It has now reached a mature state and a number of very good surveys exist in this field [Patton and Chen, 1991, Gertler, 1991]. The DX (for Diagnosis eXpert system) community emerged more recently, with foundations in the fields of Computer Science and Artificial Intelligence [Reiter, 1987, Kleer, 1992, Hamscher, 1992]. Hereafter, we give a brief description of these approaches.

2.2.4.1 FDI Analytical Redundancy Methods

In the control system community, the most common class of model-based diagnosis method proposed is the analytical redundancy method [Frank, 1996, Chen and Patton, 2012, Willsky, 1976]. The analytical redundancy methods, addressed for continuous systems, consist of two steps: (i) *generation of residuals* and (ii) *decision and fault isolation*.

Residuals are quantities that represent the inconsistency between the actual system variables and the mathematical model. In other words, residual signals are generated by comparing predicted values of system variables with the actual observed values, where the predicted values come from the available mathematical model of the system. In the decision and fault isolation stage, the residuals are examined for the likelihood of faults.

In other words, residuals are ideally zero and some residuals become non-zero if the actual system differs from the ideal one, which may be due to faults, disturbances, noise.

One of the main problems with the analytical redundancy method is the difficulty in acquiring good enough models. The demands on the accuracy of the models are usually higher than for control design, since the residual generator works open-loop. Robust methods for residual generation has received considerable attention in recent years and is an active research area [Chen and Patton, 2012, Huang, 2003].

2.2.4.2 DX Logical Diagnosis Methods

The model-based approach to diagnosis started to be investigated by Artificial Intelligence researchers in the late seventies, as a possible alternative to the expert-system approach. The fundamental paradigm of this approach, much like the analytical redundancy methods, is that of observation and prediction. The basic idea is to predict the behavior of the system using behavioral and structural models of the system and its components and compare it with observations of the actual behavior of the real system.

Two main characterizations of DX model-based diagnosis exist in the literature: (i) *consistency-based diagnosis* [Reiter, 1987] and (ii) *abductive diagnosis* [Poole, 1989]. The former is pioneered by Reiter [Reiter, 1987] known as the diagnosis from first principles. The goal of such a method is to find the set of constraints that are in conflict with the observation (output from a model which only depicts the nominal behavior). In this way, a diagnosis is a set of components that must misbehave in order to justify the observation. The abductive diagnosis methods exploit the causal model of a system, containing explicit information about which faults can occur and which chain of consequences they provoke, up to their observable manifestations.

The two approaches eventually converged into the parallel ideas of exploiting information about faults in consistency-based diagnosis, and information about correct behavior in abductive diagnosis, including in a component-oriented model information about faults corresponds to describing, along with the correct behavior of system components, also their possible faults and their consequences. Thus, models of correct behavior started to be endowed with fault models,and causal models started to include a description of nominal behavior [de Kleer and Kurien, 2004, Picardi,].

2.3 Diagnosis of Discrete-Event Systems

Discrete-Event Systems (DESs) [Cassandras and Lafortune, 2009] are systems, the dynamic of which is characterized by asynchronous occurrence of events. An event is a

fundamental concept which can be viewed and described as '*something happened*', either in systems designed by humans or in nature. Events have no property of continuation, they are instant, and can be observed only at discrete points in time. The second fundamental concept, characterizing a DES is a state, which is viewed as a result of temporally ordered discrete events, occurred starting from a moment when the system was in its initial state. In fact, the initial state of a DES characterizes the system before the occurrence of any event. The level of detail in DESs appears to be quite adequate for a large class of systems and a wide variety of failures to be diagnosed.

DESs diagnosis, particularly for finite state automata (shortly FSA), was initiated by Lin [Lin, 1994] and further treated in [Bavishi and Chong, 1994]. The off-line and on-line diagnosis issues are addressed separately. 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.

Diagnosis of DESs finds its roots in the work by Sampath et al [Sampath et al., 1995, Sampath et al., 1996]. In these works, a formal language framework for studying diagnosability properties (and online diagnosis) of un-timed DESs was proposed. The approach is closely related to the Ramage-Wonham framework for supervisory control of DESs [Ramadge and Wonham, 1987]. Actually, the DES model to be diagnosed depicts both its normal and faulty behaviors. The faults are considered as unobservable events and the diagnosis problem is then to infer about the past occurrences of fault events, on the basis of the observation recorded, within a finite delay. A diagnoser, which is an extended observer, that gives the (estimated) states and fault estimation of the system after the occurrence of each observable event is constructed from the model.

When a system model is not diagnosable, the authors in [Sampath et al., 1995] identify two means of making it diagnosable: i) *revisit the sensors' map* and ii) *design the controller* so that the faulty behavior is excited and can be detected. The former gives rise to reconfiguration/optimization of sensors [Pan and Hashtrudi-Zad, 2007, Jiang et al., 2003a, Ru and Hadjicostis, 2010, Yoo and Lafortune, 2002a, Cabasino et al., 2013b, Debouk et al., 2002a, Dallal and Lafortune, 2011] and the synthesis of observability requirements [Bittner et al., 2012], while the latter gives rise to active diagnosis issues [Sampath et al., 1998, Chanthery and Pencolé, 2009, Chen et al., 2014].

One of the main features of this approach is the ability to analyze diagnosability properties. In fact, diagnosability is the ability to detect and isolate the past occurrences of faults within finite delays. The formal definition of diagnosability, as well as the necessary and sufficient condition for checking such a property have been provided in this work. This approach will be detailed in Chapter 3.

Actually, Sampath's approach requires building a diagnoser for analyzing diagnosability. However, due to the deterministic nature of the diagnoser, its construction is performed in an **exponential** complexity in the model state-space. Aiming to reduce computational complexity for analyzing diagnosability, various approaches have been proposed on the basis of twin-plant/verifier structures [Jiang and Huang, 2001, Yoo and Lafortune, 2002b, Cimatti et al., 2003, Moreira et al., 2011, Grastien, 2009]. Such approaches allow checking diagnosability using polynomial-time algorithms and therefore, the diagnoser construction can only be performed for the diagnosable models.

2.3.1 Centralized/Decentralized DESs Diagnosis

The DES diagnosis was firstly discussed in centralized architecture [Sampath et al., 1995, Lin, 1994, Zad et al., 2003, Jiang and Huang, 2001, Yoo and Lafortune, 2002b]. In such an architecture, a global model of the system is used to be diagnosed (it may be obtained through a parallel composition of various components) and all the observations are performed at one site. Therefore, only one diagnoser is constructed, upon the current state of the diagnoser a decision o the fault occurrence is made.

Dealing with DES diagnosability using a decentralized architecture (called also codiagnosability) was proposed in [Debouk et al., 1998]. In such an architecture, a global DES model of the system is also used, but several local sites perform observations using only local diagnosers, i.e, one diagnoser is computed for each site. Since the local diagnosers do not communicate to each other, a coordinator is used to ensure the communication (via protocols) and therefore is responsible for diagnosing the fault occurrences. Particularly, the authors have specified three protocols that realize the architecture, analyze the diagnostic properties and decide about the diagnosability under each protocol. In the last two decades, DESs decentralized diagnosis has received a lot of consideration and various contributions have been proposed [Qiu and Kumar, 2006, Debouk et al., 2000, Philippot et al., 2013, Kumar and Takai, 2010, Pencole, 2000, Schumann et al., 2010, Wang et al., 2007, Sayed-Mouchaweh and Carre-Menetrier, 2008, Lafortune et al., 2005, Provan, 2002, Cabasino et al., 2013a, Basilio and Lafortune, 2009, Moreira et al., 2011, Zhou et al., 2008, Nunes et al., 2016, Takai and Kumar, 2016].

2.3.2 Distributed/Modular DESs Diagnosis

Distributed DESs diagnosis achieves diagnosis using a set of local models without referring to a global system model. The aim is to improve scalability and robustness of diagnostic methodologies. Each subsystem knows only its own part of the global model and has its local diagnoser in order to perform diagnosis locally [Zaytoon and Lafortune, 2013]. In

fact, for each component (subsystem or module), the local diagnosability information is computed [Fabre et al., 2002, Su and Wonham, 2005, Pencolé and Cordier, 2005, Genc and Lafortune, 2003, Ye and Dague, 2010, Ye et al., 2009, Ye and Dague, 2012] and (may) later combined to obtain the global diagnosability result [Pencolé et al., 2004, Pencolé et al., 2005, Schumann and Pencolé, 2007]. In comparison to centralized approaches, distributed ones require less space. In fact, due to the high space requirements of centralized methods, they can hardly be applied to large scale systems.

The notion of modular diagnosability meets the same architectural implications of distributed techniques, and it can be seen as distributed approach with the amount of information the observation spots communicate to each other being equal to zero. The challenge of modular diagnosis methodologies consists in performing diagnosis locally, i.e., at each module, while at the same time accounting for the coupling of each module with the rest of the system. Approaches that exploit the modular structure of a system for monitoring and diagnosis have been developed in [Benveniste et al., 2003, Ricker and Fabre, 2000, Contant et al., 2006, Schmidt, 2013, Zhou et al., 2008, García et al., 2005, Debouk et al., 2002b, Garcia et al., 2002, Holloway and Chand, 1994, Pandalai and Holloway, 2000]

2.4 DESs Diagnosis Using Petri Nets

DESs fault diagnosis was first discussed in the automata framework, where the basic definitions and the formal and algorithmic foundations of fault diagnosis and diagnosability analysis of DES were established. several original theoretical approaches and industrial applications have been performed. Subsequent contributions from much research teams have been concerned with the development of new models, new properties, new algorithms, and efficient solutions for fault diagnosis of DES [Zaytoon and Lafortune, 2013].

Although automata models are suitable for describing DESs, the use of Petri nets (PNs) offers significant advantages because of their twofold representation: graphical and mathematical [Murata, 1989]. In fact, the mathematical foundation underlying PNs, allows the use of standard techniques, such as Integer Linear Programming (ILP) to perform fault diagnosis tasks, while the graphical representation can be advantageously used to extend the automata-based approaches so as to deal with fault diagnosis of Petri nets by considering its reachability set. 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. Several adaptation of automata-based techniques and original theoretical approaches have been proposed to deal with diagnosis in Petri nets (PNs) framework [Wen et al., 2005, Ramírez-Treviño et al., 2007, Basile et al., 2008, Basile et al., 2009, Basile et al., 2010, Basile et al., 2012a, Dotoli

et al., 2009, Basile, 2014, Jiroveanu and Boel, 2010, Cabasino et al., 2009a, Germanos et al., 2015, Madalinski and Khomenko, 2010, Ushio et al., 1998, Chung, 2005, Jiroveanu and Boel, 2004, Cabasino et al., 2010, Cabasino et al., 2009b, Liu, 2014, Liu et al., 2014b, Li et al., 2015c, Li et al., 2015b]. Other formalisms have been used to deal with fault diagnosis of DESs. A brief literature review about DESs diagnosis using Petri nets is given in Chapter 6.

Chapter 3

Fault Diagnosis of Discrete-Event systems

Sommaire

Summary

This chapter gives a brief summary of the background on fault diagnosis of discrete-event systems. The discussion is limited to discrete-event systems modeled by finite state automata. We first present the discrete-event systems modeling and provide some notations that will be used throughout the thesis. Then, we review the fault diagnosis problem and some basic results concerning the analysis of the diagnosability property. Pioneering DES diagnosis approaches which deal with diagnosability analysis will be then recalled and a comparison discussion between these approaches is provided. It is worth noticing that the basic notions and definitions provided in this chapter are required for a better understanding of the following chapters.

This chapter is structured as follows: In Section 3.1, a general introduction to the fault diagnosis of discrete-event systems is presented. In Section 3.2, the SEDs modeling as well as the preliminary concepts needed in the sequel are introduced. In Section 3.3, we discuss the diagnosis problem in DESs. The notion of diagnosability and the main results related, are recalled in Section 3.4. The pioneering approaches for fault diagnosis of DESs with a comparison discussion are presented in Section 3.5.

3.1 Introduction

Fault diagnosis of failures in large, complex and dynamic systems is a crucial and challenging task, essentially in guaranteeing the reliable, safe, efficient and correct operation of complex engineered systems. In this context, and to fulfill such requirements, developing effective monitoring techniques becomes essential starting from the design phase of the system. In particular, having efficient tools for monitoring and diagnosis is of great interest since this prevents or at least minimizes the failure-related down-times, especially in safety-critical systems.

From the theoretical point of view and at a high level of abstraction, discrete-event systems (DESs) [Cassandras and Lafortune, 2009], are quite suitable for fault diagnosis for a wide range of applications because of the formal basis offered by the state/transition models and their associated algorithms [Lin, 1994, Biswas, 2012]. Discrete event systems (DESs) are dynamic systems with discrete state-spaces and event-driven transitions, which change their discrete states upon asynchronous occurrence of certain events. States in DESs are represented by some symbolic variables. Events in DESs are some discrete qualitative changes. In order to characterize DES, different modeling notations have been developed [Cassandras and Lafortune, 2009]: language and automata, PN theory, (max,+) algebra, Markov chains and queuing theory, discrete-event simulation, perturbation analysis, and concurrent estimation techniques. Among them, automata and PNs are the two models most used in DES-based diagnosis.

In general, there are two types of DESs: untimed DESs and timed DESs. Untimed DES models are built when we consider only the logic features, i.e., the logical order of event occurrences, without considering timing properties of the system. Thus, dynamics of untimed DESs are determined by the order/sequence of states or events, not their timing properties. In timed DESs, system behaviors are affected by timing properties, which are captured by extra "timing" events or states with clock ticks.

Fault diagnosis is a crucial task in complex dynamic systems. Due to its importance, this problem has received considerable attention from industrial and academic communities in both Artificial Intelligence (AI) and control engineering domains (CE). In particular,

an increasing amount of work has been devoted to diagnosis of DESs during the last two decades [Zaytoon and Lafortune, 2013]. Actually, the DES diagnosis problem is basically concerned with determining which faults, if any, explain a given observed abnormal behavior, based on the system model. Fault diagnosis is therefore closely related to the problem of state estimation [Zaytoon and Lafortune, 2013].

The fault diagnosis involves (1) detecting when a fault has occurred, (2) isolating the true fault from many possible fault candidates, and (3) identifying the true damage to the system. In the context of DES, fault diagnosis is often discussed through two main issues: diagnosability analysis and online diagnosis [Sampath et al., 1995, Lin, 1994]. Online diagnosis consists in inferring the occurrence of predetermined faults from the observed behavior of the system. Diagnosability refers to the capacity of the diagnoser to provide a precise diagnosis verdict. Thus, the intention of analyzing diagnosability of a system is to determine accurately whether any predetermined failure can be detected and identified within a finite delay following its occurrence [Sampath et al., 1995].

In this chapter, we are focused on fault diagnosis of untimed DESs modeled by finite state automata (FSA). To get a general overview of the literature pertaining to fault diagnosis of DESs, the reader can refer to the recent survey in [Zaytoon and Lafortune, 2013], where theoretical and practical issues, tools and other issues in relation to the diagnosis are discussed.

3.2 Discrete-Event Systems under Partial Observation

3.2.1 DESs Modeling

Discrete-event systems are quite convenient to perform the safety analysis of complex systems in a sufficiently high abstraction level [Cassandras and Lafortune, 2009, Ramadge and Wonham, 1987]. When systems are abstracted as DESs for diagnosis purposes, the model used is often finite state automata (FSA).

Définition 1 *(A Finite State Automaton* [Cassandras and Lafortune, 2009]*)*

An FSA is a tuple $G = \langle X, \Sigma, \delta, x_0 \rangle$ *where,*

- X *is a finite set of states;*
- Σ *is the alphabet of events;*
- $\delta : X \times \Sigma \to 2^X$ *is the (partial) transition function;*
- $x_0 \in X$ *is the initial state.* □

A triple $(x, \sigma, x') \in X \times \Sigma \times X$ is called a *transition* if $x' \in \delta(x, \sigma)$. The model G accounts for the normal and faulty behavior of the system, which can be described by the prefix-closed language $L \subseteq \Sigma^*$ generated by G, where Σ^* denotes the *Kleene-closure* of set Σ. An event-sequence $s = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, with $\sigma_i \in \Sigma$, is said to be *associated* with state-sequence $\pi = (x_1, x_2, \ldots, x_{n+1})$ if $\forall\, i : 0 < i \leq n, x_{i+1} \in \delta(x_i, \sigma_i)$. The partial transition function δ can be extended to sequences of events, i.e., one can write $x_{n+1} \in \delta(x_1, s)$. We write s_i to denote the i^{th} event in s. We denote by L/s the post-language of L upon s, i.e., $L/s := \{t \in \Sigma^* \,|s.t \in L\}$. We write $s \leq s'$ to denote that s is a prefix of s'. Let us consider $\sigma \in \Sigma$ and $s \in \Sigma^*$. We write $\sigma \in s$ to denote that $\exists\, i : 1 \leq i \leq |s| : s_i = \sigma$.

The partial observability issue plays a central role in fault diagnosis. In this regard, some events in Σ are observable, i.e., their occurrence can be observed, while the others are unobservable. Thus, event set Σ can be partitioned as $\Sigma = \Sigma_o \biguplus \Sigma_u$, where Σ_o denotes the set of observable events and Σ_u the set of unobservable events.

In the context of the diagnosis, faults are basically assumed to be unobservable events ($\Sigma_f \subseteq \Sigma_u$), since their detection and diagnosis would be trivial if they were observable. The set of fault events can be partitioned as disjoint fault classes $\Sigma_f = \Sigma_{f_1} \biguplus \Sigma_{f_2} \biguplus \ldots \biguplus \Sigma_{f_m}$, where $\Sigma_{f_i} (i = 1, 2, \ldots, m)$ denotes one class of faults. We consider that $\psi(\Sigma_f)$ denotes the set of event-sequences in L that end with a faulty event in Σ_f. That is, $\psi(\Sigma_f) := \{s.\sigma_f \in L : \sigma_f \in \Sigma_f\}$. With a slight abuse of notation, we write $\Sigma_f \in s$ to denote that $\exists \sigma_f \in \Sigma_f$ such that $\sigma_f \in s$. Without loss of generality, only one fault class Σ_f is considered in the sequel.

3.2.2 Operations on DESs

In what follow, we recall some useful operations on DESs , which are used in the sequel: To capture the observed behavior of the model, we define the associated projection mapping.

Définition 2 *(The projection mapping* [Lin and Wonham, 1988]*)*

With the set of observable events Σ_o, a projection mapping is associated such that $P : \Sigma^* \to \Sigma_o^*$, with $P(\varepsilon) = \varepsilon$ (ε is the empty event-sequence) and

$$P(\sigma) = \begin{cases} \varepsilon, & \sigma \in \Sigma_u \\ \sigma, & \sigma \in \Sigma_o \end{cases}$$

$$P(s\sigma) = P(s)P(\sigma), s \in \Sigma^*, \sigma \in \Sigma$$

□

The effect of P on an event-sequence $s \in \Sigma^*$ is simply to erase the unobservable events in it. The inverse projection operation P_L^{-1} is defined by $P_L^{-1}(y) = \{s \in L \subseteq \Sigma^* : P(s) = y\}$.

The general setting of the inverse projection P_L^{-1} is not restricted to the event-sequences which finished with an observable event (i.e., $\forall \omega \in \Sigma^*, P_L^{-1}(\omega) = \{s \in L \subseteq \Sigma^* : P(s) = \omega\} \cap \Sigma^*\Sigma_o$) [Fabre et al., 2016]. In this chapter, we consider that P_L^{-1} is restricted to the event-sequences. However, in some parts of our thesis, the general setting of the inverse projection P_L^{-1} is considered since it has an impact on the obtained results and will be discussed in Chapter 5. The projection operator can then be extended to language L by applying the projection to all traces of L, i.e., $P : 2^{\Sigma^*} \rightarrow 2^{\Sigma_o}$. Therefore, if $L \subseteq \Sigma^*$, then $P(L) = \{t \in \Sigma_o^* \mid (\exists\, s \ \in L)\, [P(s) = t]\}$.

We recall the classic notions of synchronous product and parallel composition of two FSA.

Définition 3 *(Synchronous product* [Cassandras and Lafortune, 2009]*)*

Consider $G_i = \langle X_i, \Sigma, \delta_i, x_{0_i} \rangle$ *(for* $i = 1, 2$*). The synchronous product of* G_i *is the FSA* $\mathcal{G} = G_1 \times G_2 = \langle (X_1 \times X_2), \Sigma, \delta, (x_{0_1} \times x_{0_2}) \rangle$ *where* $((x_1, x_2), \sigma, (y_1, y_2)) \in \delta$ *if and only if* $(x_1, \sigma, y_1) \in \delta_1$ *and* $(x_2, \sigma, y_2) \in \delta_2$.

Using the projection mapping, we can characterize the language resulting from the synchronous product as follows: $L(G_1 \times G_2) = L(G_1) \times L(G_2)$. The product between an FSA and itself is called a *self-product.*. Such an operation is also called the strict parallel composition.

Définition 4 *(Parallel composition* [Cassandras and Lafortune, 2009]*)*

Consider $G_i = \langle X_i, \Sigma_i, \delta_i, x_{0_i} \rangle$ *(for* $i = 1, 2$*). The parallel composition of of* G_i *is the FSA* $G_1 \parallel G_2 = \langle (X_1 \times X_2), \Sigma_1 \cup \Sigma_2, \delta_{1||2}, (x_{0_1} \times x_{0_2}) \rangle$ *where*

$$\delta_{1||2}((x_1, x_2), \sigma)) := \begin{cases} (\delta_1(x_1, \sigma), \delta_2(x_2, \sigma)) & if\ \sigma \in \Sigma_1 \cap \Sigma_2 \\ (\delta_1(x_1, \sigma), x_2) & if\ \sigma \in \Sigma_1 \backslash \Sigma_2 \\ (x_1, \delta_2(x_2, \sigma)) & if\ \sigma \in \Sigma_2 \backslash \Sigma_1 \end{cases} \tag{3.1}$$

□

Using the projection mapping, we can characterize the language resulting from the parallel composition as follows: $L(G_1 \parallel G_2) = P^{-1}[L(G_1)] \cap P^{-1}[L(G_2)]$.

Définition 5 *(Σ_u-closure)*

Σ_u-closure of G is FSA $G' = \langle X_o, \Sigma_o, \delta_{G'}, x_0 \rangle$, where

- $X_o = \{x_0\} \cup \{x \in X | \exists x' \in X, \exists \sigma \in \Sigma_o : x \in \delta(x', \sigma)\}$ *is the finite set of states;*
- Σ_o *is the finite set of observable events;*
- $x_0 \in X$ *is the initial state;*
- $\delta_{G'} \subseteq (X_o \times \Sigma_o \times X_o)$ *is the transition relation, defined as follows:* $(x, \sigma, x') \in \delta_{G'}$ *if* $\exists s \in \Sigma^* : x' \in \delta(x, s)$ *s.t.* $s = (\sigma_1, \sigma_2, \ldots, \sigma_n = \sigma)$*:* $\sigma_i \in \Sigma_u (i = 1, 2, \ldots, n-1)$ *and* $\sigma_n \in \Sigma_o$. □

In other words, the Σ_u-closure is an $\varepsilon-$reduction assuming all events of Σ_u are first replaced by ε. That is, every unobservable transition in G is erased in the closure, while preserving the set of observation, i.e., $L(G') = P(L(G))$.

Définition 6 *(Determinization)*
Determinization of FSA G *is* $Det(G) = \langle \mathcal{X}, \Sigma_o, \delta_o, \mathcal{X}_0 \rangle$ *where*

- $\mathcal{X} = 2^X$ *is the set of nodes, each node contains a set of system states;*
- $\mathcal{X}_0 = \{x_0\}$ *the initial node;*
- $\delta_o(q, \sigma) = \{x' \in X | \exists s \in \Sigma^*, \exists x \in q : x' \in \delta(x, s\sigma)\}$ *is the transition relation.* □

The determinization can be easily performed from the Σ_u-closure of G and similarly, determinization preservs the observation, i.e., $L(Det(G)) = P(L(G))$.

3.3 The Fault Diagnosis Problem

Partial observability on the system behavior is a main issue one has to deal with when performing diagnosis analysis. Diagnosability analysis and online diagnosis where firstly formulated by M. Sampath et al. [Sampath et al., 1995, Sampath et al., 1996] in the framework of automata models. In fact, the diagnosis activity is basically concerned with determining which faults (unobservable events), if any, explain a given observed sequence of events, based on the model of the system. Fault diagnosis is therefore closely related to the problem of state observability, which consists in building a deterministic automaton, called the observer, whose transitions are due to the observable events of the system and whose states are estimates of the true system state [Zaytoon and Lafortune, 2013].

In the original framework introduced in [Sampath et al., 1995, Sampath et al., 1996], the behavior of the DES is assumed to be known and a model of it is available as a finite state automaton G over an alphabet (set of events) $\Sigma = \Sigma_o \cup \Sigma_u$ with $\Sigma_f \subset \Sigma_u$. Therefore, the aim of fault diagnosis is to detect the faulty sequences of the DES by only observing

events in Σ_o. A faulty sequence is an event sequence of the DES containing at least one occurrence of an event in Σ_f. It is assumed that an observer which has to detect the faults knows the specification/model of the DES and on the basis of such knowledge, it has to announce whether an observation (a sequence in Σ^*) was produced by a faulty sequence or not [Cassez, 2009, Cassez and Tripakis, 2008].

The diagnosis problem is defined as the problem of synthesizing a diagnoser, that is, a function $Diag : P(L(G)) \longrightarrow \{yes, no, ?\}$, which answers the question whether all the event sequences consistent with the observation have experienced at least one fault event in Σ_f. The main properties of such a function are: Correctness, which means that Yes and No answers should be accurate, while bounded diagnosability means that the fault events should be diagnosed within finite number of observable events [Morvan and Pinchinat, 2009, Chédor et al., 2015, Jéron et al., 2006].

Formally, the diagnosis problem can be defined as follow:

Définition 7 *(The diagnosis problem* [Jéron et al., 2006]*) given an FSA G, Σ_o is the set of observable events, Σ_f is the set of fault events. For an observable event sequence $s \in \Sigma_o$, the function $Diag : P(L(G)) \longrightarrow \{yes, no, ?\}$ should verify*

$$Diag(s) := \begin{cases} \text{“}Yes\text{”} & if \quad \forall t \in P^{-1}(s) : \Sigma_f \in t \\ \text{“}No\text{”} & if \quad \forall t \in P^{-1}(s) : \Sigma_f \notin t \\ ? & otherwise. \end{cases} \tag{3.2}$$

□

Figure 3.1 illustrates the diagnoser function, where after each observation, a verdict regarding the status of the system is generated.

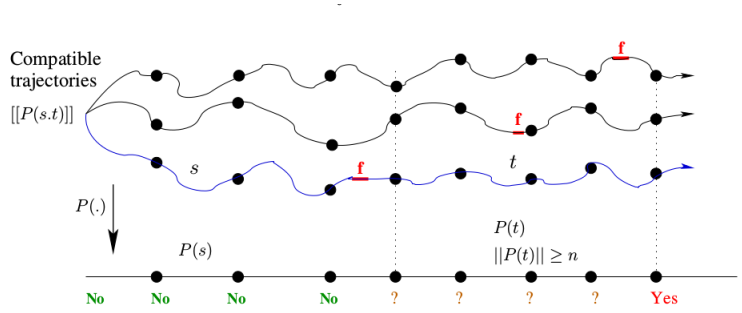


Figure 3.1 – Illustration of the diagnosis function [Jéron et al., 2006]

In fact, the diagnoser $\mathcal{D}$ of an FSA G (i.e., function $Diag$) can be derived from the corresponding deterministic DES $Det(G)$. A systematic procedure for building the diagnoser is firstly proposed in [Sampath et al., 1995, Sampath et al., 1996] (which we recall

in the next section). Building the diagnoser is not a difficult task, per se: it relies on classical power-set construction (Σ_u-closure, determinization). Hence, for finite state automata, it induces an unavoidable exponential blow-up [Sampath et al., 1995, Morvan and Pinchinat, 2009, Tsitsiklis, 1989], even for succinct representations [Rintanen et al., 2007a]. Therefore, on-the-fly computation of the diagnoser is a key technique that improves the construction and the analysis procedure [Liu, 2014]. Moreover, it can also deal with infinite state settings [Tripakis, 2002, Baldan et al., 2010, Morvan and Pinchinat, 2009].

The practical interest of using a diagnoser greatly depends on its capabilities to output accurate answers. That is, whatever method is used for the diagnoser, the central question is whether the diagnoser will eventually detect any faulty execution. Such a property can be formally captured by the notion of *diagnosability* [Sampath et al., 1995, Sampath et al., 1996]. In simple terms, diagnosability is a qualitative property of the diagnoser which refers to the ability (of the diagnoser) to infer accurately, from partially observed executions, about the faulty behavior within a finite delay after a possible occurrence of a fault. Such a property is widely studied in fault diagnosis of DESs. In other words, diagnosability is a qualitative property of the diagnoser which ensures a finite latency for any observation of a faulty execution, which corroborates the *completeness* of the diagnoser.

3.4 Diagnosability

The original definition of diagnosability was introduced in the seminal work of [Sampath et al., 1995] under the assumptions that faults are permanent (i.e., once a fault occurs, the system remains irreparably faulty), the language generated by G is live, and no cycles composed only of unobservable events exist in G. The formal definition of diagnosability is recalled as follows.

Définition 8 *(Diagnosability* [Sampath et al., 1995]*)*

A prefix-closed and live language L is said to be diagnosable, with respect to projection mapping P and class of faults Σ_f, if the following holds:

$$(\exists n \in \mathbb{N})\ [\forall s \in \psi(\Sigma_f)]\ (\forall t \in L/s)\ [|t| \geq n \Rightarrow D]$$

where the diagnosability condition D is:

$$\omega \in P_L^{-1}[P(s.t)] \Rightarrow \Sigma_f \in \omega$$

□

The above definition means the following: let s be any sequence generated by G that ends with a fault event in Σ_f, and let t be any sufficiently long continuation of s. Condition D then requires that every sequence ω belonging to language L, which produces the same observable event-sequence as $s.t$ $(P(\omega) = P(s.t))$, holds a fault event from Σ_f.

Lemma 1 *(non-diagnosability* [Jéron et al., 2006]*)*

An FSA G is non-diagnosable w.r.t. projection mapping P and class of faults Σ_f, if and only if there exist two indistinguishable infinite executions ω_1 and ω_2 such that ω_1 reaches fault event f while ω_2 does not.

Notice that diagnosability considers only infinite executions that do not diverge, where an infinite execution diverges if it has an unobservable infinite suffix. In other words, we are only interested in the fair behavior of the system w.r.t. observability [Morvan and Pinchinat, 2009].

Définition 9 *(Diagnosable system* [Zaytoon and Lafortune, 2013]*)*

An FSA G is diagnosable if it is possible to detect within a finite delay occurrences of faults of any type using the record of observed events. Alternatively speaking, diagnosability requires that every occurrence of every fault event leads to observations distinct enough to enable unique identification of the fault event within a finite delay. □

3.4.1 $K-$Diagnosability

The diagnosability problem consists in qualitatively determining the existence of a finite delay upon which any fault (or class of faults) can be detected and identified. Diagnosability just means the existence of an upper bound without specifying its value. In practice, such a property can be insufficient to ensure a safe operation of the system, namely when we deal with safety-critical systems. Indeed, this delay could be too long and faults may have dramatic consequences before being diagnosed and before some reconfiguration actions can be undertaken. Thus, some "quantitative versions" of diagnosability have been developed, namely $K-$diagnosability [Basile et al., 2012b, Sampath et al., 1995, Liu, 2014, Cabasino et al., 2012a, Dallal and Lafortune, 2010, Dallal and Lafortune, 2011] (it is also called *bounded diagnosability* [Jéron et al., 2006] or *latency* problem [Morvan and Pinchinat, 2009]). Unlike the classic definition of diagnosability, $K-$diagnosability requires the quantitative determination of the finite delay (as an integer K). Thus, $K-$diagnosability means that one can determine with certainty the occurrence of a fault in the system after K observations. Hereafter, we recall the original definition of $K-$diagnosability introduced by Dallal et al. [Dallal and Lafortune, 2010, Dallal and Lafortune, 2011].

Définition 10 *($K-$diagnosability* [Dallal and Lafortune, 2011]*)*
An FSA G is $K-$diagnosability with respect to projection mapping P and class of faults Σ_f, if no pair of event sequences $s_1, s_2 \in L(G)$ exists such that:

1. *s_1 has an occurrence of a fault event $f \in \Sigma_f$ and s_2 does not;*
2. *s_1 has at least $K+1$ events after fault event f;*
3. *$P(s_1) = P(s_2)$.* □

This means that, for any two event sequences in the system model which share the same observation, one faulty and the other normal, the system is said to be $K-$diagnosable if and only if the two executions do not have K (or more) successive identical observation after the occurrence of the fault. It is worth noticing that, according to this definition, the bounded k is measured by the number of observable events. However, it can also correspond to the number of unobservable events or the number of reachable states prior to the fault detection.

Example 1 *(from* [Zaytoon and Lafortune, 2013]*)*

This idea is conceptually sketched out in Figure 3.2 for a $6-$diagnosable of fault event f. Figure 3.2 (a) shows trace st of events with prefix s ending with f and suffix t containing 5 observable events. This figure indicates that f is not diagnosable within 5 observable events after its occurrence because one of the other 5 trajectories that are indistinguishable from st (the one that ends in state b) does not contain f. This means that the observation trace $P(st)$ does not allow one to conclude with certainty whether event f has occurred or not.

Figure 3.2(b) represents an extension of t with an observable event e and shows that only four trajectories remain indistinguishable from ste. The occurrence of fault event f, the last event of trace s, can be detected with certainty in this case because each of the 4 indistinguishable trajectories contains f. Fault f will be $6-$diagnosable if each possible occurrence of f in the entire system language can be detected with certainty, in the same way, after 6 observations [Zaytoon and Lafortune, 2013].

Lemma 2 *An FSA G is non-diagnosable $\iff \forall K \in \mathbb{N}$, G is not $K-$diagnosable.*

Generally speaking, there are two main problems on K-diagnosability. The first is to analyze $K-$diagnosability of a system under a given value K, i.e., whether or not any fault (or class of faults) can be detected and identified within K steps (observable events basically) after its occurrence. The second is to find the minimum K for a diagnosable system. This is called $K_{min}-$diagnosability problem [Cassez and Tripakis, 2008, Cabasino

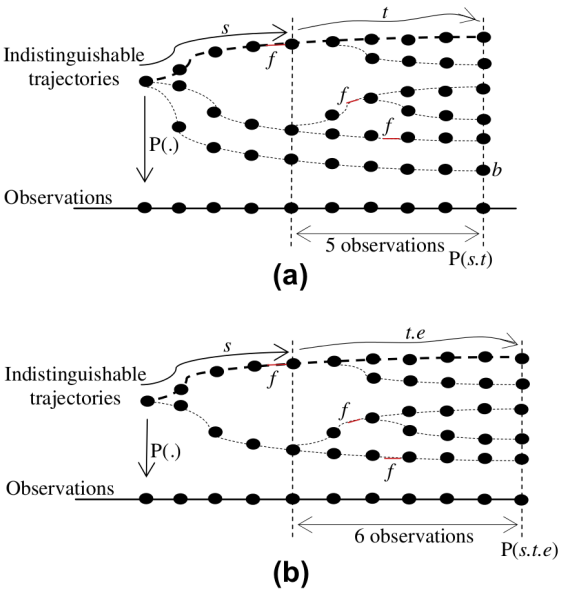


Figure 3.2 – Illustration of $K-$diagnosability

et al., 2012a, Liu, 2014] (it is also called the *bounded-latency* diagnosability [Morvan and Pinchinat, 2009], or the worst case detection delay (WCDD) [Eser Kart and Schmidt, 2015]).

3.4.2 Complexity Analysis

In this section, we summarize the main results in the literature, regarding the complexity pertaining to various diagnosis problems. The classical fault diagnosis problems are the following [Cassez, 2009]:

For a given FSA G and w.r.t. projection mapping P and class of faults Σ_f

Problem 1 : $(K-diagnosability)$
For a given $K \in \mathcal{N}$, is G $K-$diagnosable?

Problem 2 : $(diagnosability)$
Is G diagnosable?

Problem 3 : $(K_{min}-diagnosability)$
What is the minimum value of K s.t. G is Kdiagnosable?

Problem 4 : $(diagnoser\ synthesis)$
If G is diagnosable, synthesis a witness diagnoser $\mathcal{D}$ of G.

Lemma 3 [Cassez, 2009]*:*

- *Problem 1 (3.4.2) can be solved in* ExpTime*. Under some consideration, it can be done in* PTime *($\mathcal{O}(|G|^4)$;*
- *Problem 1 (3.4.2) is in* PSpace

Lemma 4 [Yoo and Lafortune, 2002b, Jiang and Huang, 2001, Cassez and Tripakis, 2008]*:*

- *Problem 2 (3.4.2) can be solved in* PTime *($\mathcal{O}(|G|^2)$;*

It is worth noticing that the non-diagnosability is NLogSpace-*complete* [Rintanen et al., 2007a]. Moreover, using the non-diagnosability algorithm in the context of [Jéron et al., 2006] yields a PTime upper bound and NLogSpace lower bound. Regarding the infinite state system, diagnosability problem is, in general, undecidable [Morvan and Pinchinat, 2009].

Lemma 5 [Cassez, 2009]*:*

- *Problem 3 (3.4.2) can be solved in* PTime *($\mathcal{O}(|G|^4)$). It can also be done in ($\mathcal{O}(|G|^3)$) [Yoo and Garcia, 2003].*

Lemma 6 [Sampath et al., 1995, Sampath et al., 1996]*:*

- *Problem 4 (3.4.2) can be solved in* ExpTime.

In the following section, we recall the two approaches for the analysis of diagnosability of DES, namely, Sampath's diagnoser approach [Sampath et al., 1995] and Twin-plant/verifier approaches [Yoo and Lafortune, 2002b, Jiang and Huang, 2001]. Note that these approaches are the basis of several further approaches which developed adaptations and extensions of the formers.

3.5 The pioneering Approaches

In order to apply these approaches, the system model under investigation assumed to satisfy the following:

1. The system model is an FSA which has a live generated language;
2. No cycle composed of only unobservable event exists in the model;
3. Faults are permanent (i.e., once a fault occurs, the system remains irreparably faulty).

3.5.1 Sampath's Diagnoser Approach

Sampath et al. [Sampath et al., 1995] have proposed a systematic approach for analyzing diagnosability. It consists in building a particular observer, the so-called *diagnoser*. To build the diagnoser, an intermediate model, i.e., the so-called generator (or pre-diagnoser), has to be established, a priori. Such a model is subsequently used to check diagnosability.

In fact, the generator G' of model G is nothing more that the Σ_u-*closure* of G (see Definition 5). It is worth recalling that when, the generator is combined with the tagging function that associates to each state a tag ('N' for normal states and 'F' for faulty ones), then it is called a *pre-diagnoser* or an augmented generator. The diagnoser is then built by the determinization (see Definition 6) of the pre-diagnoser. Hereafter, we give a formal definition of the diagnoser.

Définition 11 *(Diagnoser [Sampath et al., 1995])*

The diagnoser of a system model G is a deterministic FSA $G_d = \langle \mathcal{Q}, \Sigma_d, \delta_d, q_0 \rangle$ associated with a tagging function $Tag : X_o \to 2^{\Delta}$, with $\Delta = \{N, F\}$ (where N means 'normal' and F means 'faulty').

- $\mathcal{Q} = 2^{(X_o \times \Delta)}$ *is the set of diagnoser states;*
- $\Sigma_d = \Sigma_o$ *is the set of (observable) events;*
- $\delta_d : \mathcal{Q} \times \Sigma_d \to \mathcal{Q}$ *is the transition relation;*
- $q_0 = (x_0, N)$ *is the initial diagnoser state.* □

Each diagnoser state q has the form $q = \{(x_1, l_1), \ldots, (x_n, l_n)\}$, with $x_i \in X_o$ and $l_i \in \Delta$. If $\forall i = 1, \ldots, n$, we have $l_i = N$ (resp. $l_i = F$), the diagnoser state q is said to be N-*certain* (resp. F-*certain*), otherwise, i.e., if $\exists i, j$ such that $l_i = N$ and $l_j = F$ in diagnoser state q, it is an F-*uncertain* state. The fault propagation rules in the diagnoser are depicted in Figure 3.3.

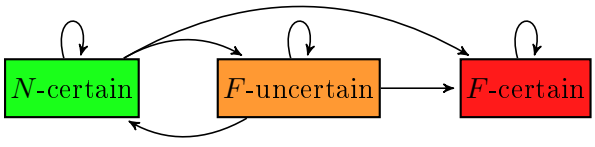


Figure 3.3 – Fault propagation in the classic diagnoser

On the basis of the constructed diagnoser, Sampath et al. [Sampath et al., 1995] have developed a necessary and sufficient condition for analyzing diagnosability. It consists in

tracking some particular cycles in the diagnoser and the pre-diagnoser. Before discussing the analysis of diagnosability, we recall the notions of cycles in the system model G and in the diagnoser G_d.

Définition 12 *(Cycles in G and G_d)*

- *A series of states $x_1, x_2, \ldots, x_n \in X$ is said to form a cycle in G if $\exists s = \sigma_1\sigma_2\ldots,\sigma_n \in L(G)$ such that $\delta(x_i, \sigma_i) = x_{(i+1)mod_n}$ for $i = 1, 2, \ldots, n$.*
- *A series of states $q_1, q_2, \ldots, q_n \in \mathcal{Q}$ is said to form a cycle in G_d if $\exists\, \sigma_i \in \Sigma_d$ such that $\delta_d(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $i = 1, 2, \ldots, n$.*

□

An *F-uncertain* cycle in the diagnoser is a cycle that is composed exclusively by F-uncertain states. An *F-indeterminate* cycle in the diagnoser is defined as an *F-uncertain* cycle, for which two corresponding cycles, sharing the same observable projection as the F-uncertain cycle, exist in the pre-diagnoser such that one cycle involves only faulty states, while the other one involves only normal states.

Such a notion of *F-indeterminate* cycle is crucial, since it helps to give the necessary and sufficient condition for diagnosability.

Theorem 1 *(Necessary & Sufficient Condition)*

An FSA G is diagnosable if and only if no F-indeterminate cycle exists in its diagnoser G_d for any class of faults Σ_f. □

Example 2 *Let us consider FSA G in Figure 3.4 (adapted from [Sampath et al., 1995]). The set of observable events is $\Sigma_o = \{a, b, d, t\}$ and the set of unobservable events is $\Sigma_u = \{u, f\}$ with $f \in \Sigma_f$.*

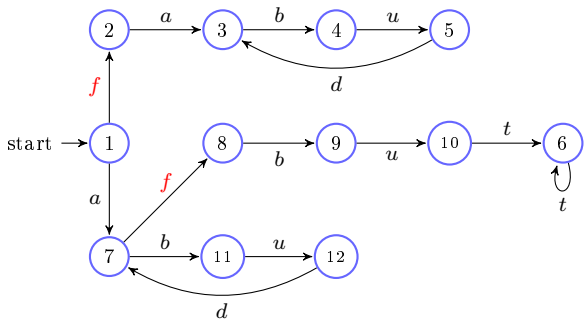


Figure 3.4 – The FSA G

The pre-diagnoser and diagnoser corresponding to G are given in Figure 3.5. There exists an F-uncertain cycle composed of $\{3F, 7N\}$ and $\{4F, 9F, 11N\}$ w.r.t. the observable sequence $(bd)^*$ in the diagnoser. This cycle corresponds to two cycles in the pre-diagnoser (and so in G). The first one is composed of faulty states $3, 4$ w.r.t. sequence $a(bd)^*$. The second cycle is composed of normal states $7, 11$. Thus, one can infer, according to Theorem 1, that an F-indeterminate cycle exists in the diagnoser and, consequently, G is non-diagnosable.

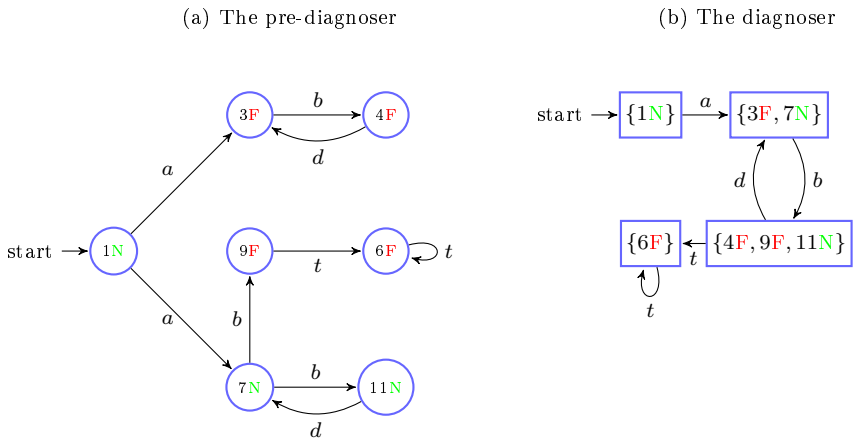


Figure 3.5 – Pre-diagnoser and diagnoser of FSA G

It is worth noticing that, besides its usefulness for analyzing (offline) diagnosability, the diagnoser serves to perform the actual monitoring task, online. In fact, the diagnoser is established offline.

3.5.1.1 The Complexity Analysis

The diagnoser approach suffers from the combinatorial explosion problem due to the determinization operation (Definition 6). In fact, the complexity of constructing the diagnoser and testing the diagnosability is **exponential** in the number of states of the system model and **double-exponential** in the number of fault classes. In [Sampath et al., 1995], it has been proved that a maximum bounded delay n_i for diagnosing faults in Σ_{F_i} exists: $n_i \leq C_i \times n_0 + n_0$, where n_0 is the maximal length of sequences composed exclusively of unobservable events and C_i is the number of states in the diagnoser states pertaining to the F_i-uncertain cycles. Moreover, for a diagnosable model, a fault in Σ_{F_i} can be detected at most after $n_0 + n_i$ events after the fault occurrence.

3.5.2 Twin-Plant/Verifier Approaches

In order to (partially) overcome the issue of the combinatorial explosion problem in Sampath's diagnoser approach, some further approaches aiming to reduce the computing complexity have been proposed [Yoo and Lafortune, 2002b, Jiang and Huang, 2001]. The main idea behind these approaches is to investigate the automata composition, i.e., the parallel composition and the synchronous product, in order to deal with diagnosability. In fact, such compositions allow the analysis of pairs of infinite event-sequences separately. Therefore, according to Lemma 1, it is possible to decide about the diagnosability.

3.5.2.1 The Twin-Plant Approach [Jiang and Huang, 2001]

In such an approach, given a system model G, the pre-diagnoser G' is first constructed and then a structure called *twin-plant* (denoted $\mathcal{G}$ is obtained by the strict parallel composition (i.e, the synchronous product) of the pre-diagnoser with itself, i.e. $\} = G' \times G'$ (see Definition 3), based on the observable events to obtain all the pairs of event-sequences sharing the same observations.

Each twin-plant state q is a pair of the system states, $q = \{(x_1, l_1), (x_2, l_2)\}$, with $x_i \in X_o$ and $l_i \in \{N, F\}$. If $l_i = N$ (resp. $l_i = F$) for $i = 1, 2$, the twin-plant state q is said to be N-*certain* (resp. F-*certain*). Otherwise, state q, it is an F-*ambiguous* state.

An F-confused cycle (called also an infinite critical pair or critical path) in the twin-plant is a cycle which is composed exclusively of F-*ambiguous* states.

According to Lemma 1, the necessary and sufficient condition on the basis of the twin-plant structure, is announced as follows,

Theorem 2 *(Necessary and sufficient condition* [Jiang and Huang, 2001]*)*

An FSA G is diagnosable with respect to projection mapping P and class of faults Σ_f if and only if no F-confused cycle exists in its corresponding twin-plant $\mathcal{G}$. □

Example 3 *Let us take again automaton G of Example 2. Figure 3.6 depicted its corresponding twin-plant $\mathcal{G}$. It is worth noticing that only the live part of $\mathcal{G}$ is constructed. One can observe that $\mathcal{G}$ contains some F-confused cycles (drawn in orange color). Therefore, according to Theorem 2, G is non-diagnosable.*

3.5.2.2 The Complexity Analysis

The complexity of constructing the pre-diagnoser is $\mathcal{O}([X]^2 \times 2^{2|F|} \times |\Sigma_o|)$, whereas, the complexity of constructing the twin plant is $\mathcal{O}([X]^4 \times 2^{4|F|} \times |\Sigma_o|)$. Finally, detecting the presence of F-confused cycle is linear in the number of states/transitions of $\mathcal{G}$. Therefore

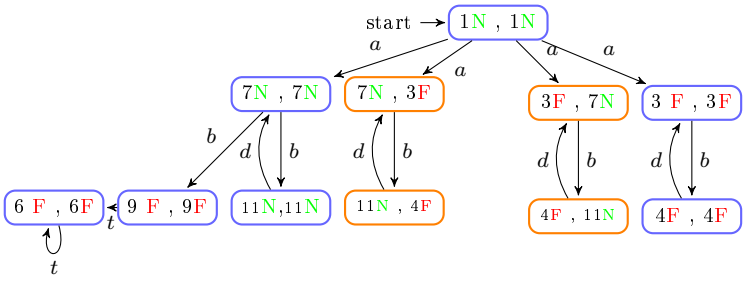


Figure 3.6 – Twin plant $\mathcal{G}$ of (Example 2)

the complexity of the twin-plant approach is **polynomial** (4^{th} order) in the number of states in G and **exponential** in the number of fault classes.

3.5.2.3 The Verifier Approach [Yoo and Lafortune, 2002b]

The verifier approach consists of the construction of a non-deterministic automaton V_{Σ_f} (called Σ_f-*verifier*, with Σ_f is a fault class), by performing the the parallel composition of a system model G with itself augmented with a tagging function that associated to each system state its types (N for normal states, F for faulty ones).

Similarly to the twin-plant structure, each verifier state q is a pair of the system states, $q = \{(x_1, l_1), (x_2, l_2)\}$, with $x_i \in X_o$ and $l_i \in \{N, F\}$. If $l_i = N$ (resp. $l_i = F$) for $i = 1, 2$, the verifier state q is said to be *N-certain* (resp. *F-certain*), otherwise, state q, it is an *F-ambiguous* state.

Verifier V_{Σ_f} is called Σ_f*-confused* if it contains at least one cycle which is composed exclusively of *F-ambiguous* states. Otherwise, it called Σ_f*-confused-free*.

According to Lemma 1, the necessary and sufficient condition on the basis of the verifier structure, is announced as follows,

Theorem 3 *(Necessary and sufficient condition* [Yoo and Lafortune, 2002b]*)*

An FSA G is diagnosable with respect to projection mapping P and class of faults Σ_f if and only if its corresponding verifier V_{Σ_f} is Σ_f-confused-free □

Example 4 *Let us take again automaton G of Example 2. Figure 3.7 depicted its corresponding verifier V_F. It is worth noticing that only a part of V_F is generated. One can observe that V_F is an $V - F$-confused since it contains some F-confused cycles (drawn in orange color). Therefore, according to Theorem 3, G is non-diagnosable.*

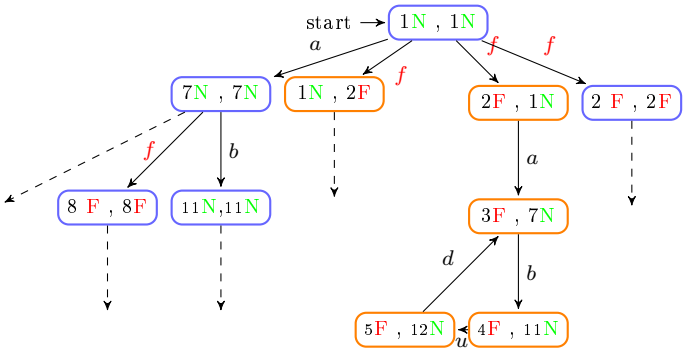


Figure 3.7 – Verifier V_F of (Example 2)

3.5.2.4 The Complexity Analysis

According to [Yoo and Lafortune, 2002b], for a reachable state v in the verifier, the number of feasible transition from v is $3 \times \Sigma$ at most. Since the number of reachable states of is $4 \times |X|^2$ states at most, the complexity of constructing the verifier takes $12 \times |X|^2 \times \Sigma$ time. Therefore, the overall complexity is **polynomial** $(\mathcal{O}([X]^2 \times |\Sigma_o|))$ in the number of the system states. Moreover, it has been proved that for a diagnosable model, any fault occurrence can be detected within $|X|^2$ transitions after the fault occurs.

One can underline that the twin-plant/verifier structures have an interesting feature, which is the symmetric property. It means that each path in the twin-plant/verifier has its symmetric path (e.g., a path containing a state (x_1N, x_2F) then it has its symmetric path which contains (x_2F, x_1N) state, and vice versa). Since the interesting part of the twin-plant/verifier structure for analyzing diagnosability is the ambiguous one (i.e, paths composed of $F_uncertain$ states, then the symmetric property has been exploited in order to reduce the generate state-space. In [Grastien, 2009, Tripakis, 2002], a reduced twin-plant structure is proposed. It consists in keeping the faults behave only on a copy of the model. The other copy, noted G_N, is non-failure, i.e., it contains only the nominal behavior obtained by erasing the faulty transitions and their successors. Therefore, the reduced twin plant is computed by the parallel synchronization $G_N \parallel G$. Recently, Moreira et al. [Moreira et al., 2011], have proposed a reduced verifier with a lower complexity $(\mathcal{O}(|X|^2 \times (|\Sigma| - |\Sigma_f|)))$. In fact, the algorithm in [Moreira et al., 2011] perform the parallel synchronization between the non-failure copy of the model G_N and the co-accessible part from faulty states, noted G, (i.e., $G_N \parallel G_F$). The efficiency of such a construction is due to the fact that in the synchronization step, only the traces that lead to the violation of diagnosability are computed [Hosseini et al., 2013].

3.5.3 A Comparison Between the Diagnoser/Twin-plant/Verifier approaches

In what follow, we provide a comparison between the pioneering approches for fault diagnosis of DESs (discussed approach) regarding various features.

- **Offline/online diagnosis:**
  Regarding the offline diagnosis (i.e., diagnosability analysis), both approaches can deal with this issue, and relatively with $K/K_{min}-$diagnosability. However, regarding the online diagnosis, twin-plant and verifier approaches do not consider such a task due to their nondeterministic structures, contrary to the diagnoser approach, which deals with the online diagnosis (thanks to its determinstic structure). Consequently, the diagnoser approach remains a principal technique to perform both diagnosability analysis and online diagnosis.

- **Intermediate models:**
  The verifier approach has the particularity to be built directly from the system model, contrary to the diagnoser and twin-plant approaches were an intermediate model (the generator or the pre-diagnoser) is necessary for building the diagnoser/twin-plant. Such intermediate constructions increase the memory demanding.

- **The theoretical complexity:**
  As discussed in the above sections, the diagnoser is built in exponential complexity contrary to the twin-plant/verfier were only a polynomial complexity is needed. Nevertheless, comparative studies show that the diagnoser-based approach is more efficient for analyzing diagnosability of some kinds of system models than the twin-plant/verifier approaches, as witnessed by [Liu, 2014].

- **The verification procedure:**
  The procedure for checking diagnosabilily on the basis of the diagnoser approach consists of a *double check* procedure. Firstly, the existence of *F-uncertain* cycle is checked by exploring the diagnoser paths. Secondly, once an *F-uncertain* cycle is found, its corresponding event-trace is used to check the existence of an *F-indeterminate* cycle by executing the event-trace on the pre-diagnoser, from the initial state. In the case of a diagnosable system, this procedure is repeated as many times as there are *F-uncertain* cycles in the diagnoser. In fact, this double check procedure affects drastically the memory/time consumption of the verification algorithm. Despite the need of an intermediate model for constructing the twin-plant, the verification procedure is performed upon one check on the twin-plant structure

(i.e., checking the F-confused cycles). The same procedure is also performed using the verifier approach.

3.6 Conclusion

This chapter proposes a brief overview on the fault diagnosis of DESs under partial observation. The definitions, assumptions, notions and notations, as well as the discussions related the pioneering approaches provided in this chapter will be used in the remainder of this dissertation.

Part II

CONTRIBUTIONS REGARDING THE DIAGNOSER-BASED APPROACH

Chapter 4

A New Variant of the Diagnoser-Based Approach

Sommaire

Summary

In this chapter, we propose a new version of the well-known diagnoser-approach. It consists in separating normal states from faulty ones in each diagnoser node. Such a distinction serves to more efficiently track the faulty and fault-free traces in the diagnoser paths. On the basis of various features that characterize this new diagnoser, we develop a systematic procedure for checking the necessary and sufficient condition for diagnosability without needing to construct any intermediate model (i.e., generator or pre-diagnoser). Finally, we provide an on-the-fly algorithm to simultaneously construct the diagnoser and analyze diagnosability. Therefore, in general, the diagnoser need not be built completely to check diagnosability and perform online diagnosis. Some experimentation are conducted in order to evaluate the effectiveness and the scalability of the proposed approach with respect to the reference approaches in the field, namely, Sampath's diagnoser and the verifier approach. This chapter is enclosed by a comparison discussed between our diagnoser and Sampath's diagnoser regarding various features.

The work presented in this chapter is the subject of publications in VeCOS'15 [Boussif et al., 2015] *and submitted journal papers in IJCCBS* [Boussif et al., 2016b] *and IEEE-TAC* [Boussif and Ghazel, 2016c].

This chapter is structured as follows: In Section 4.2, the system model as well as the preliminary concepts needed in the sequel are introduced. Section 4.3 is devoted to discussing the construction of the new diagnoser variant. Some theoretical results and a systematic procedure to check the necessary and sufficient condition for diagnosability are provided. In Section 4.4, we develop an on-the-fly algorithm for constructing the diagnoser and checking diagnosability simultaneously. Section 4.5 discusses some extensions of the proposed approach. In Section 4.6, the effectiveness and the scalability of the approach are evaluated through some experimentation. A comparison discussion w.r.t. the reference approach (Sampath's diagnoser) is presented in Section 4.7. Finally, conclusion remarks and future research directions are given in Section 4.8.

4.1 Introduction

The fault diagnosis involves (*i*) detecting when a fault has occurred, (*ii*) isolating the true fault from many possible fault candidates, and (*iii*) identifying the true damage to the system. In the context of DES, fault diagnosis is often discussed through two main issues: diagnosability analysis and online diagnosis [Sampath et al., 1995, Lin, 1994]. Online diagnosis consists in inferring the occurrence of predetermined faults from the observed behavior of the system. Diagnosability refers to the capacity of providing a precise diagnosis verdict. Thus, the intention of analyzing diagnosability of a system is to determine accurately whether any predetermined failure can be detected and identified within a finite delay following its occurrence [Sampath et al., 1995].

The pioneering work which deals with these issues was proposed in [Sampath et al., 1995] where a formal definition of diagnosability was introduced. Such a work provided a necessary and sufficient condition for diagnosability as well as a systematic approach, based on the so-called *diagnoser*, with the aim to verify diagnosability and perform the online diagnosis. However, the combinatorial explosion problem is inherent to the defined approach and the state-space of the diagnoser is, in the worst case, exponential w.r.t. the size of the model state-space.

In order to reduce the computing complexity, some further approaches were proposed. In [Yoo and Lafortune, 2002b], a polynomial-time algorithm for checking diagnosability based on the so-called *verifier* is adopted. In [Jiang and Huang, 2001], an algorithm based on the *twin-plant* (a parallel composition of the investigated model with itself) is proposed. Nevertheless, these approaches deal only with diagnosability analysis and do not consider online diagnosis. Moreover, comparative studies show that the diagnoser-based approach is more efficient for analyzing diagnosability of some kinds of system models than these approaches [Liu, 2014]. Consequently, the diagnoser-based approach remains a principal

technique to deal with both diagnosability analysis and online diagnosis.

The diagnoser-based approach has firstly been introduced for systems modeled by centralized automata [Sampath et al., 1995, Zad et al., 2003, Sampath et al., 1996]. The approach is then extended to deal with decentralized architectures [Debouk et al., 1998, Qiu and Kumar, 2006, Debouk et al., 2000, Philippot et al., 2013, Kumar and Takai, 2010, Pencole, 2000, Schumann et al., 2010, Wang et al., 2007, Sayed-Mouchaweh and Carre-Menetrier, 2008, Lafortune et al., 2005, Provan, 2002, Cabasino et al., 2013a, Basilio and Lafortune, 2009, Moreira et al., 2011, Zhou et al., 2008, Nunes et al., 2016, Takai and Kumar, 2016], modular and distributed architectures [Debouk et al., 2002a, García et al., 2005, Contant et al., 2006, Ye and Dague, 2013]. More recently, a series of interesting contributions, inspired from the diagnoser approach, have been proposed in the Petri net framework [Cabasino et al., 2014, Jiroveanu et al., 2008, Ushio et al., 1998, Liu et al., 2014b, Li et al., 2015c].

The main issues related to the diagnoser-based approaches can be outlined as follows:

1. The high complexity of construction, which is exponential in the number of states of the original model, and double-exponential regarding the classes of faults. This consequently hampers the scalability of the approach.

2. The approach is based on the analysis of two graphs. The first graph is a non-deterministic observer (called pre-diagnoser, or generator), while the second one is a deterministic automaton, called diagnoser (or equivalently, generator/diagnoser [Sampath et al., 1995], MBRG/BRD [Cabasino et al., 2009b], FM-graph/FM-set graph [Liu, 2014], etc.).

3. The double-checking procedure, which consists in one verification upon the diagnoser (i.e., the existence of *F-uncertain* cycles) and the other upon the generator or the pre-diagnoser (i.e., checking whether the *F-uncertain* cycle is an *F-indeterminate* one or not). In fact, in general such a double-checking procedure highly increases the verification time.

To partially overcome these limitations, we propose in this chapter a new diagnoser variant with a structure that allows us to check directly the necessary and sufficient condition, without building any intermediate model. Moreover, we provide an on-the-fly algorithm for constructing the diagnoser and checking diagnosability simultaneously, which improves the efficiency in terms of memory and time consumption.

In what follows, we highlight the main features of the proposed approach and we make some comparisons with existing diagnoser-based approaches.

1. The developed approach provides a new structure for representing the diagnoser nodes. Such a structure explicitly separates between the normal and the faulty states in each node. This feature allows us to separately track the normal and the faulty traces directly in the diagnoser.

2. In the same way as for the existing diagnoser-based approaches, our diagnoser serves both to check diagnosability and to perform online diagnosis.

3. In our approach, the diagnoser is directly built from the original system model, without needing to construct any intermediate model, as usually done in the classic diagnoser approaches.

4. On the basis of the proposed structure of the diagnoser, a sufficient condition for the undiagnosability of the model is proposed. Such a condition is used for the on-the-fly verification of diagnosability. Hence, the model is stated to be non-diagnosable as soon as the condition is met, without building or analyzing the whole diagnoser.

5. The approach provides a systematic procedure for checking the necessary and sufficient condition for diagnosabilily (i.e., the existence of F-indeterminate cycle or not) without returning to any intermediate model to check if an F-uncertain cycle corresponds to two cycles, a faulty one and a non-faulty one. Besides, the procedure performs directly on the F-uncertain cycle with no need to start from the initial state. All these aspects allow for significantly speeding up the verification process.

6. An on-the-fly algorithm, based on a depth-first search procedure, for both constructing the diagnoser and verifying diagnosability simultaneously is proposed. The algorithm aims to generate as less state-space as possible, particularly when the system is undiagnosable, which improves the memory/time consumption.

Besides diagnosability analysis, the developed technique allows for checking $\mathcal{K}/\mathcal{K}_{min}-$diagnosability and, when the system is diagnosable, the constructed part of the diagnoser can be directly used as an online diagnoser.

4.2 The System Model

The system to be diagnosed is modeled as a finite state automaton (FSA) $G = \langle X, \Sigma, \delta, x_0 \rangle$. In this chapter, we keep the same notations and notions introduced in Section 3.2.

Regarding the projection mapping, we consider the general setting of the inverse projection P_L^{-1}, which is not restricted to the event-sequences which end with an observable event, i.e., $\forall \omega \in \Sigma_o^*, P_L^{-1}(\omega) = \{s \in L \subseteq \Sigma^* : P(s) = \omega\}$.

4.3 A New Variant of The Diagnoser

In order to depict the structure of the new diagnoser, we firstly introduce the following notations:

- $Enable_{\Sigma}(x) = \{\sigma \in \Sigma | \delta(x,\sigma) \neq \emptyset\}$, is the set of events in Σ that are enabled from state x. The generalization to a subset of states $X' \subseteq X$ and a subset of events $\Sigma' \subseteq \Sigma$, is $Enable_{\Sigma'}(X') = \{\sigma \in \Sigma' | \exists\, x \in X' : \delta(x,\sigma) \neq \emptyset\}$ which denotes the set of enabled events in Σ' from the set of states X', i.e., $Enable_{\Sigma'}(X') = \bigcup_{x \in X'} Enable_{\Sigma'}(x)$.

- $Img(X,\sigma) = \bigcup_{x \in X'} \delta(x,\sigma)$ with $\sigma \in \Sigma$, is the generalization of the transition relation to a subset of states $X' \subseteq X$. The generalization of the transition relation δ to a subset of states $X' \subseteq X$ and a subset of events $\Sigma' \subseteq \Sigma$ is $Img(X',\Sigma') = \bigcup_{x \in X'} \bigcup_{\sigma \in \Sigma'} \delta(x,\sigma)$.

- $Reach_{\Sigma'}(x) = \{x\} \cup \{x' \in X | \exists t \in \Sigma'^{*} : x' \in \delta(x,t)\}$ is the set of states reached by the occurrence of a sequence of events in Σ' from x (will be used particularly for the unobservable reachability). The generalization of this notion for a set of states is $Reach_{\Sigma'}(X') = \bigcup_{x \in X'} Reach_{\Sigma'}(x)$.

4.3.1 The Structure of a Diagnoser Node

In our diagnoser variant, nodes are equivalent to states in the classic diagnoser [Sampath et al., 1995], except that an explicit distinction is made, within each node, between the normal states (denoted by set $\mathcal{X}_N$) and the faulty ones (denoted by set $\mathcal{X}_F$), and we indicate if there exists a (faulty) transition from $\mathcal{X}_N$ to $\mathcal{X}_F$. We will show in the sequel how such a structure can be advantageously used to render diagnosability analysis more efficient than when using the classic diagnoser.

Figure 4.1 (a) depicts the general form of a diagnoser node which contains two sets of states; $\mathcal{X}_N$ represents the set of normal states, while $\mathcal{X}_F$ represents the set of faulty ones. Some faulty states may be reached from normal states in the same node through the occurrence of faulty events. This is depicted by a faulty transition from $\mathcal{X}_N$ to $\mathcal{X}_F$ within the diagnoser node. In order to simplify the notation, we use $a.\mathcal{X}_N$ (resp. $a.\mathcal{X}_F$) to indicate the set of normal states $\mathcal{X}_N$ (resp. the fault states $\mathcal{X}_F$) corresponding to diagnoser node a. Moreover, the notation $x \in a$ means that state $x \in \mathcal{X}_N \cup \mathcal{X}_F$.

One can differentiate between three types of diagnoser nodes, in the same way as in the classic diagnoser:

- **N-certain diagnoser node**: is a diagnoser node of which the set of faulty states is empty ($\mathcal{X}_F = \emptyset$);
- **F-certain diagnoser node**: is a diagnoser node of which the set of normal states is empty ($\mathcal{X}_N = \emptyset$);
- **F-uncertain diagnoser node**: is a diagnoser node of which neither the normal set, nor the faulty set, is empty, i.e., $\mathcal{X}_N \neq \emptyset$ and $\mathcal{X}_F \neq \emptyset$.

Hereafter, we introduce the formal definition of a diagnoser node.

Définition 13 *(Diagnoser node)*
Consider an FSA $G = \langle X, \Sigma, \delta, x_0 \rangle$ *with* $\Sigma = \Sigma_o \biguplus \Sigma_u$ *and* $\Sigma_f \subseteq \Sigma_u$. *We define a diagnoser node* $a = \langle \mathcal{X}_N, \mathcal{X}_F \rangle$ *as a non-empty set of states satisfying:*

1. $\forall x \in X$, $s \in \Sigma^*$, *and* $\Sigma_f \in s$ *such that* $x \in \delta(x_0, s)$ *(i.e., x is reachable by a faulty sequence):* $x \in a \Leftrightarrow Reach_{\Sigma_u}(x) \subseteq a.\mathcal{X}_F$;
2. $\forall x \in X$, $s \in (\Sigma \backslash \Sigma_f)*$ *such that* $x \in \delta(x_0, s)$ *(i.e., x is reachable by a fault-free sequence):* $x \in a \Leftrightarrow \mathcal{X}' = Reach_{\Sigma_u \backslash \Sigma_f}(x) \subseteq a.\mathcal{X}_N \wedge Reach_{\Sigma_u}(Img(\mathcal{X}', \Sigma_f)) \subseteq a.\mathcal{X}_F$.
3. $\forall x, x' \in a$, $\exists s, s' \in \Sigma^*$, *such that* $x \in \delta(x_0, s)$, $x' \in \delta(x_0, s')$, *and* $P(s) = P(s')$.

□

The dashed arrows in Figure 4.1 (a) show the different possibilities that an observable transition from a diagnoser node may correspond to. For instance, in Figure 4.1 (a) observable event σ_2 output by diagnoser node a may be output from the normal or the faulty sets or from both sets. Moreover, a faulty transition f may or may not exist between $\mathcal{X}_N$ and $\mathcal{X}_F$. As for the dashed arrows, only the faulty transitions linking $\mathcal{X}_N$ to $\mathcal{X}_F$ inside a given node are actually encoded (as a single faulty transition) in the diagnoser (cf. Figure 4.1 (a)) using a boolean variable which is *True* when such transitions exist, and *False* if not.

4.3.2 The Diagnoser Construction

For a given FSA G, the new diagnoser variant can be defined as follows.

Définition 14 *(Diagnoser variant)*
Let $G = \langle X, \Sigma, \delta, x_0 \rangle$ *be an FSA to be diagnosed. The diagnoser associated with G is a deterministic FSA* $\mathcal{D} = \langle \Gamma, \Sigma_o, \delta_{\mathcal{D}}, \gamma_0 \rangle$, *where:*

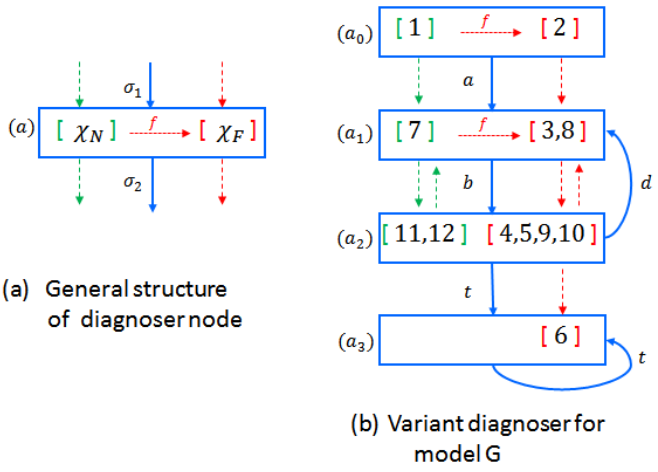


Figure 4.1 – The structure of the diagnoser node and the diagnoser variant of FSA G in Figure 3.4

1. Γ *is a finite set of diagnoser nodes;*

2. γ_0 *is the initial diagnoser node with:*

   a) $\gamma_0.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(x_0)$;

   b) $\gamma_0.\mathcal{X}_F = Reach_{\Sigma_u}(Img(\gamma_0.\mathcal{X}_N, \Sigma_f))$.

3. $\delta_{\mathcal{D}} : \Gamma \times \Sigma_o \to \Gamma$ *is the transition relation, defined as follows:*

$$\forall a, a' \in \Gamma,\ \sigma \in \Sigma_o : a' = \delta_{\mathcal{D}}(a, \sigma)$$

$$\Leftrightarrow \begin{cases} a'.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a.\mathcal{X}_N, \sigma)) \\ \wedge \\ a'.\mathcal{X}_F = Reach_{\Sigma_u}(Img(a'.\mathcal{X}_N, \Sigma_f) \cup Img(a.\mathcal{X}_F, \sigma)) \end{cases}$$

□

To summarize, the diagnoser $\mathcal{D}$ is constructed as follows: let the current node be a, and an observable event σ. The target diagnoser node a' is computed following the rules below:

1. If $\sigma \in Enable(a.\mathcal{X}_N) \cap Enable(a.\mathcal{X}_F)$ then:

   - $a'.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a.\mathcal{X}_N, \sigma))$.
   - $a'.\mathcal{X}_F = Reach_{\Sigma_u}(Img(a'.\mathcal{X}_N, \Sigma_f) \cup Img(a.\mathcal{X}_F, \sigma))$.

2. If $\sigma \in Enable(a.\mathcal{X}_N) \setminus Enable(a.\mathcal{X}_F)$ then:

- $a'.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a.\mathcal{X}_N, \sigma))$.
- $a'.\mathcal{X}_F = Reach_{\Sigma_u}(Img(a'.\mathcal{X}_N, \Sigma_f))$.

3. If $\sigma \in Enable(a.\mathcal{X}_F) \setminus Enable(a.\mathcal{X}_N)$ then:

- $a'.\mathcal{X}_N = \emptyset$.
- $a'.\mathcal{X}_F = Reach_{\Sigma_u}(Img(a.\mathcal{X}_F, \sigma))$.

These aforementioned rules preserve a specific fault propagation scheme regarding the assumption that faults are considered to be permanent. This can be depicted in Figure 4.2, and can be summarized in the three points below:

- From an *N-certain* diagnoser node, either an *N-certain* diagnoser node or an *F-uncertain* one can be reached;
- From an *F-certain* diagnoser node, only *F-certain* diagnoser nodes can be reached;
- From an *F-uncertain* diagnoser node, any of an *F-uncertain*, an *N-certain* or an *F-certain* diagnoser node can be reached.

Since all the successors of an *F-certain* diagnoser node are also *F-certain*, we do not need in our approach to construct them (i.e., the subsequent *F-certain* nodes) because it is unnecessary from the diagnosis point of view. Indeed, as regards diagnosability analysis, only the analysis of F-uncertain cycles is necessary and since faults are permanent, one can be certain that no such cycle can be generated following an F-certain node. As for online diagnosis, once an F-certain node is reached, one can be sure that the system will remain indefinitely faulty.

It is worth noticing that the fault propagation rules of our diagnoser are different from those of the classic diagnoser, since an *F-certain* diagnoser node cannot be reached directly from an *N-certain* diagnoser node. This is due to the fact that in the building procedure of our diagnoser, the unobservable reachability is computed before the current node is left.

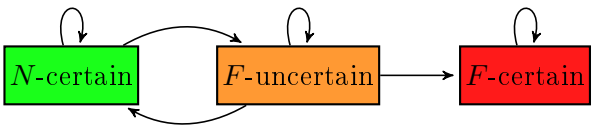


Figure 4.2 – Fault propagation in the diagnoser variant

In order to better illustrate the diagnoser construction procedure, let us again consider FSA G in Figure 3.4, introduced in the previous chapter (Chapter 3). Then, its corresponding diagnoser is depicted in Figure 4.1(b). The initial node (a_0) is composed of the initial state of G (state 1) and state 2 reachable from state 1 by the occurrence of faulty event f. One can also notice that there exists an *F-uncertain* cycle composed of nodes (a_1) and (a_2) by executing the observable event sequence $a(bd)^*$. Diagnoser node (a_3) is reached after the occurrence of event t and it contains only a set of faulty states $(a_3.\mathcal{X}_N = \emptyset)$. Thus, it is an *F-certain* node. As *F-certain* nodes are unnecessary for analyzing diagnosability, and since we deal with permanent faults, the subsequent nodes are not constructed.

4.3.3 Some Properties of the New Diagnoser Variant

In this section, we discuss some main features characterizing the new diagnoser variant that will be used in the sequel to analyze diagnosability.

Définition 15 *Let us consider three successive diagnoser nodes a, a', and a'' such that $a' = \delta_{\mathcal{D}}(a, \sigma)$ and $a'' = \delta_{\mathcal{D}}(a', \sigma')$ $(\sigma, \sigma' \in \Sigma_o)$.*

- ***The set of input normal states*** $\mathcal{I}_N^{a,\sigma}(a')$*: is the set of states in $a'.\mathcal{X}_N$, which are directly reachable from $a.\mathcal{X}_N$ through the occurrence of observable event σ. Formally, $\mathcal{I}_N^{a,\sigma}(a') = \{x' \in a'.\mathcal{X}_N | \exists\, x \in a.\mathcal{X}_N : x' \in \delta(x, \sigma)\}$.*

We note that $\mathcal{I}_N^{a,\sigma}(a')$ can be also written as : $\mathcal{I}_N^{a,\sigma}(a') = Img(a.\mathcal{X}_N, \sigma)$.

- ***The set of output normal states*** $\mathcal{O}_N^{\sigma}(a')$*: is the set of states in $a'.\mathcal{X}_N$, which directly enable observable event σ' to reach some states in $a''.\mathcal{X}_N$. Formally, $\mathcal{O}_N^{\sigma'}(a') = \{y' \in a'.\mathcal{X}_N | \exists\, x'' \in \mathcal{I}_N^{a',\sigma'}(a'') : x'' \in \delta(y', \sigma')\}$.*

□

It is worth noticing that the notions of input (resp. output) normal states are related to a sequence of nodes, which means that many input (resp. output) normal states may exist for node a, i.e., according to the entering (resp. outgoing) transitions.

Property 1 *Let $a, a' \in \Gamma$ be two diagnoser nodes, such that $a' = \delta_{\mathcal{D}}(a, \sigma)$ for $\sigma \in \Sigma_o$. Then, we have:* $(\forall x' \in a'.\mathcal{X}_N)$, $(\exists t \in (\Sigma_u \backslash \Sigma_f)^*, \exists x \in \mathcal{I}_N^{a,\sigma}(a'))$*: $x' \in \delta(x, t)$.*

□

This means that for a diagnoser node a' that is reached from a diagnoser node a by the occurrence of an event σ, all the states in $a'.\mathcal{X}_N$ are reachable from some states in the set of input normal states corresponding to diagnoser node a, i.e., from $\mathcal{I}_n^{a,\sigma}(a')$.

This property is inferred directly from the construction procedure of the diagnoser. Indeed, $\forall a, a' \in \Gamma$, $\sigma \in \Sigma_o$: $a' = \delta_{\mathcal{D}}(a, \sigma) \Rightarrow a'.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a.\mathcal{X}_N, \sigma))$, which means that each state in $a'.\mathcal{X}_N$ is reachable from $\mathcal{I}_N^{a,\sigma}(a')$ (which is nothing other than $Img(a.\mathcal{X}_N, \sigma)$). Therefore, in the case where a' has more than one predecessor node in the diagnoser, each state in $a'.\mathcal{X}_N$ is reachable from some states in any set of input normal states corresponding to any entering transition for a'.

Property 2 *Let us consider three successive diagnoser nodes a, a', a'' such that $a' = \delta_{\mathcal{D}}(a, \sigma)$ and $a'' = \delta_{\mathcal{D}}(a', \sigma')$ $(\sigma, \sigma' \in \Sigma_o)$. We have:*

1. $\mathcal{I}_N^{a',\sigma'}(a'') = Img(\mathcal{O}_N^{\sigma'}(a'), \sigma')$.
2. $\forall\, x'' \in \mathcal{I}_N^{a',\sigma'}(a'')$: $\exists x' \in \mathcal{I}_N^{a,\sigma}(a'), s \in (\Sigma_u \setminus \Sigma_f)^*$ *such that* $x'' \in \delta(x', s.\sigma')$. □

The first point is trivial given the definition of $Img()$ operation. The second one means that each input normal state $x'' \in \mathcal{I}_N^{a',\sigma'}(a'')$ is reachable from some (normal) states $x' \in \mathcal{I}_N^{a,\sigma}(a')$ through a fault-free unobservable sequence (possibly empty) followed by σ'. Hereafter, a sketch of the proof is given.

Proof.

Let us pick $x' \in \mathcal{I}_N^{a,\sigma}(a')$, $y' \in \mathcal{O}_N^{\sigma'}(a')$ and $x'' \in \mathcal{I}_N^{a',\sigma'}(a'')$. From the diagnoser construction, we have $a'.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a.\mathcal{X}_N, \sigma))$. In fact, the states in $a.\mathcal{X}_N$ that enable the output observable event σ are those in $\mathcal{O}_N^{\sigma}(a)$, which means that $a'.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(\mathcal{O}_N^{\sigma}(a), \sigma))$. From the first point, we have $\mathcal{I}_N^{a,\sigma}(a') = Img(\mathcal{O}_N^{\sigma}(a), \sigma)$, then $a'.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(\mathcal{I}_N^{a,\sigma}(a'))$, which means that any state in $a'.\mathcal{X}_N$ is reachable from some states in $\mathcal{I}_N^{a,\sigma}(a')$ through a faultless unobservable sequence. Regarding the set of output normal states, this means that:

$$\forall y' \in \mathcal{O}_N^{\sigma'}(a'), \exists x' \in \mathcal{I}_N^{a,\sigma}(a'),\ \exists s \in (\Sigma_u \setminus \Sigma_f)^* : \quad y' \in \delta(x', s) \qquad (1)$$

In other terms, each state from any set of output normal states is reachable from some states in any set of input normal states. This ensures a chaining between input normal states and output normal states in the same node, regardless of the considered entering and outgoing transitions of the node.

Furthermore, as $\mathcal{I}_N^{a',\sigma'}(a'') = Img(\mathcal{O}_N^{\sigma'}(a'), \sigma')$, then:

$$\forall x'' \in \mathcal{I}_N^{a',\sigma'}(a''), \exists y' \in \mathcal{O}_N^{\sigma'}(a') : x'' \in \delta(y', \sigma') \qquad (2)$$

From (1) and (2), one can infer that:

$$\forall\, x'' \in \mathcal{I}_N^{a',\sigma'}(a''), \exists x' \in \mathcal{I}_N^{a,\sigma}(a'), s \in (\Sigma_u \backslash \Sigma_f)^* : x'' \in \delta(x', s.\sigma')$$

In a similar way to the above, this ensures a chaining between the input normal states in any successive nodes of the diagnoser. Such a feature will be used in the following section to derive necessary and sufficient conditions for diagnosability. ■

Figure 4.3 (A) illustrates the above proof, where successive diagnoser nodes a, a', a'' are considered (only the sets of normal states are represented). $\mathcal{I}_N^{a,\sigma}(a') = \{x'_1, x'_2, x'_3, x'_4\}$, $\mathcal{O}_N^{\sigma'}(a') = \{y'_1, y'_2, y'_3\}$ and $\mathcal{I}_N^{a',\sigma'}(a'') = \{x''_1, x''_2, x''_3\}$. One can observe that each state from the set of input normal states of a'' is reachable from some states in the set of input normal states of a' (e.g. x''_1 is reachable from state x'_1).

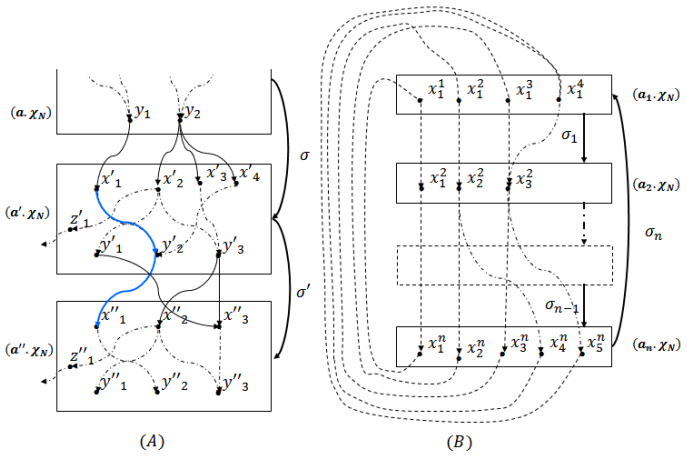


Figure 4.3 – Proofs of Property 2 and Proposition 1

4.3.4 Diagnosability Verification

Diagnosability verification using the classic diagnoser consists in checking the existence of *F-indeterminate* cycles. As mentioned before, this method requires a double-checking process, i.e., to check the existence of an *F-uncertain* cycle in the diagnoser and in the case when such cycles exist, to return to an intermediate model (the generator or the pre-diagnoser) to check if the found *F-uncertain* cycle corresponds to two cycles: a faulty cycle and a non-faulty one. Moreover, the cycle analysis procedure is performed each time, starting from the initial state of the intermediate model. In order to avoid this double-checking process, we develop a systematic procedure to check *F-indeterminate* cycles without needing to construct an intermediate model. Besides, the elaborated procedure does not require starting from the initial node to check whether an F-uncertain cycle is

an F-indeterminate one. Instead, only the nodes of the F-uncertain cycle are handled. This technique relies on some theoretical results that we discuss in what follows:

Proposition 1 *Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$[1] for $1 \leq i \leq n$. Then, there exists, at least, one fault-free cycle in G, that shares the same observation $(\sigma_1, \sigma_2, \ldots, \sigma_n)^*$.* □

Proof.

According to the diagnoser building procedure, $\forall\, i : 1 \leq i \leq n,\ a_{(i+1)mod_n}.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a_i.\mathcal{X}_N, \sigma_i))$. Then, from Property 2, we have : for $1 \leq l < n, \forall\, x_{(l+1)} \in \mathcal{I}_N^{a_l,\sigma_l}(a_{(l+1)})$: $\exists\, x_l \in \mathcal{I}_N^{a_{(l-1)},\sigma_{(l-1)}}(a_l), \exists\, s_l \in (\Sigma_u \setminus \Sigma_f)^*$ s.t. $x_{(l+1)} \in \delta(x_l, s_l.\sigma_l)$, and $\forall\, x_1 \in \mathcal{I}_N^{a_n,\sigma_n}(a_1)$: $\exists x_n \in \mathcal{I}_N^{a_{(n-1)},\sigma_{(n-1)}}(a_n), \exists\, s_n \in (\Sigma_u \setminus \Sigma_f)^*$ s.t. $x_1 \in \delta(x_n, s_n.\sigma_n)$.

Now, let us take an input normal state x_l^i from $\mathcal{I}_N^{a_{(l-1)},\sigma_{(l-1)}}(a_l)$, with $1 < l \leq n$ and $1 \leq i \leq k_l$ (k_l is the number of input normal states in $\mathcal{I}_N^{a_{(l-1)},\sigma_{(l-1)}}(a_l) = \{x_l^1, \ldots, x_l^{k_l}\}$). Then, from above, $\forall\, i : 1 \leq i \leq k_l$, $\exists\, t = s_{(l+1)_{mod_n}}, \sigma_{(l+1)_{mod_n}}, s_{(l+2)_{mod_n}}, \sigma_{(l+2)_{mod_n}}, \cdots, s_{(l+n-1)_{mod_n}}, \sigma_{(l+n-1)_{mod_n}}$, with $s_{i'} \in (\Sigma_u \setminus \Sigma_f)^*$, for $1 \leq i' \leq n$ such that $x_l^i \in \delta(x_l^j, t)$, with $1 \leq j \leq k_l$.

In other words, by applying the result of Property 2 recursively upon the *F-uncertain* cycle $a_1, a_2, \ldots, a_n \in \mathcal{D}$, one can infer that $x_l^i \in \mathcal{I}_N^{a_{(l-1)},\sigma_{(l-1)}}(a_l)$ is reachable from an input state x_l^j also in $\mathcal{I}_N^{a_{(l-1)},\sigma_{(l-1)}}(a_l)$. By repeating this operation while considering x_l^j instead of x_l^i, and so on, at least k times, one can infer that the input state x_l^i is certainly visited twice (See Figure 4.3 (B)). Therefore, it becomes obvious that a cycle exists in the original model G. As all the states are normal, it is consequently a fault-free cycle. ■

Remark 1 *This result is interesting for checking F-indeterminate cycles, using both the classic diagnoser or our diagnoser. It is, in fact, sufficient to check that an F-uncertain cycle in the diagnoser corresponds to a faulty cycle in the original model (or the intermediate model), without checking the existence of the fault-free cycle.*

Proposition 2 *Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$ with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$. Then, if $\forall i : 1 \leq i \leq n$, $Img(a_i.\mathcal{X}_N, \Sigma_f) = \emptyset$, then $c\ell$ is an F-indeterminate cycle.*

□

This result means that if in all the diagnoser nodes of an *F-uncertain* cycle no faulty transitions from the normal set of states to the faulty one (by means of Img operator) exists, therefore this cycle is an *F-indeterminate* cycle.

[1] $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ means the following: $\forall i < n\ :\ \delta_{\mathcal{D}}(a_i, \sigma_i) = a_{i+1}$ and $\delta_{\mathcal{D}}(a_n, \sigma_n) = a_1$.

Proof.

Let us consider $c\ell = a_1, a_2, \ldots, a_n \in \mathcal{D}$ to be an *F-uncertain* cycle. Also, let us assume that $\forall i : 1 \leq i \leq n$, $Img(a_i.\mathcal{X}_N, \Sigma_f) = \emptyset$.

From Proposition 1, a corresponding fault-free cycle exists in the original model G, which shares the same observation $(\sigma_1, \sigma_2, \ldots, \sigma_n)^*$. Then, it only remains to prove that a corresponding faulty cycle exists in model G, which also shares the same observation as $c\ell$. As $\forall i : 1 \leq i \leq n$, $Img(a_i.\mathcal{X}_N, \Sigma_f) = \emptyset$, we have $a_{(i+1)mod_n}.\mathcal{X}_F = Reach_{\Sigma_u}(Img(a_i.\mathcal{X}_F, \sigma_i))$. One can observe that it corresponds exactly to the construction rule of the normal set of states in the diagnoser node (i.e., $a_{(i+1)mod_n}.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a_i.\mathcal{X}_N, \sigma_i))$). Thus, using the same reasoning in Proof 4.3.4, we infer that a corresponding faulty cycle, which also shares the same observation as $c\ell$, exists in the original model. Thus, the *F-uncertain* cycle $c\ell$ is an *F-indeterminate* one as well. ■

Remark 2 *Proposition 2 can be viewed as a sufficient condition for non-diagnosability, since a system model is non-diagnosable if the condition in Proposition 2 is satisfied by the diagnoser. Hence, diagnosability analysis is stopped as soon as the condition in Proposition 2 is satisfied. In this case, the diagnoser will be constructed partially. Indeed, as will be discussed in Section 4.4, diagnosability analysis will be performed simultaneously on the fly as the diagnoser is set up. Such a feature will potentially speed up the diagnosability analysis; in particular when the system is non-diagnosable.*

The above result is used below to introduce two notions of series associated to an *F*-uncertain cycle.

Définition 16 *(Series $S^{c\ell}$)*

Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$. Series $S^{c\ell} = S_1^{c\ell}, S_2^{c\ell}, \ldots$ associated with $c\ell$, is defined as follows:

$$\begin{cases} S^{c\ell} : \mathbb{N}^* \rightarrow 2^X \\ S_1^{c\ell} = a_1.\mathcal{X}_F \qquad \textit{(the first term of } S^{c\ell}\textit{)} \\ S_i^{c\ell} = Reach_{\Sigma_u}(Img(S_{(i-1)}^{c\ell}, \sigma_{(i-1)_{mod_n}})), \forall i > 1. \end{cases} \tag{4.1}$$

□

In fact, series $S^{c\ell}$ tracks the subsets of faulty states in each node along $c\ell$, but does not consider the faulty states generated through the occurrence of some faulty transitions starting form the normal subset in the traversed nodes (except for $S_1^{c\ell}$ which holds all the faulty states of a_1, i.e., $S_1^{c\ell} = a_1.\mathcal{X}_F$). In fact, series $S^{c\ell}$ is introduced with the aim of tracking the actual faulty cycles corresponding to a given *F-uncertain* cycle, if such cycles exist in the original model G.

Définition 17 *(Series $S'^{c\ell}$)*

Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$, for $1 \leq i \leq n$. Series $S'^{c\ell} = S'^{c\ell}_1, S'^{c\ell}_2, S'^{c\ell}_3, \ldots$ corresponding to $c\ell$, is defined as follows:

$$\begin{cases} S'^{c\ell} : \mathbb{N}^* \to 2^X \\ S'^{c\ell}_i = S^{c\ell}_{(1+(i-1)n)}, \forall i \in \mathbb{N}^*. \end{cases} \tag{4.2}$$

□

It is worth noticing that $S'^{c\ell}$ can also be written as follows: $S'^{c\ell} = S^{c\ell}_1, S^{c\ell}_{(1+n)}, \ldots, S^{c\ell}_{(1+kn)}$, $S^{c\ell}_{(1+(k+1)n)}, \ldots$, for $k \in \mathbb{N}$. In other terms, series $S'^{c\ell}$ is a *sub-series* of series $S^{c\ell}$, which is extracted from $S^{c\ell}$ by considering sample terms with n steps (n is the number of nodes in the F-uncertain cycle). Thus, series $S'^{c\ell}$ preserves some properties of series $S^{c\ell}$ (i.e., convergence, limits, etc.).

Proposition 3 *Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$, and let $S'^{c\ell} = S'^{c\ell}_1, S'^{c\ell}_2, S'^{c\ell}_3, \ldots$ be its corresponding series as defined above. Then, the following property holds:*

$$\forall k \in \mathbb{N}^* \ : \ S'^{c\ell}_{k+1} \subseteq S'^{c\ell}_k$$

i.e., $\forall k \in \mathbb{N}^* \ : \ S^{c\ell}_{(1+kn)} \subseteq S^{c\ell}_{(1+(k-1)n)}$. □

Proof.

This property can be straightforwardly proved using mathematical induction. Firstly, for $k = 1$, it is direct that $S^{c\ell}_{n+1} \subseteq S^{c\ell}_1$, since we have: $S^{c\ell}_1 = a_1.\mathcal{X}_F$ and $S^{c\ell}_2 = Reach_{\Sigma_u}(Img(S^{c\ell}_1, \sigma_1)) \subseteq a_2.\mathcal{X}_F = Reach_{\Sigma_u}(Img(a_2.\mathcal{X}_N, \Sigma_f) \cup Img(a_1.\mathcal{X}_F, \sigma_1))$, with the same logical reasoning along the event-sequence $\sigma_1, \ldots, \sigma_n$, we obtain:
$S^{c\ell}_n = Reach_{\Sigma_u}(Img(S^{c\ell}_{(n-1)}, \sigma_n)) \subseteq a_n.\mathcal{X}_F = Reach_{\Sigma_u}(Img(a_n.\mathcal{X}_N, \Sigma_f) \cup Img(a_{(n-1)}.\mathcal{X}_F, \sigma_{(n-1)}))$. Therefore, $S^{c\ell}_{n+1} = Reach_{\Sigma_u}(Img(S^{c\ell}_n, \sigma_n)) \subseteq a_1.\mathcal{X}_F = Reach_{\Sigma_u}(Img(a_1.\mathcal{X}_N, \Sigma_f) \cup Img(a_n.\mathcal{X}_F, \sigma_n)) = S^{c\ell}_1$. Now, let us suppose, as by heredity, that $S^{c\ell}_{(1+kn)} \subseteq S^{c\ell}_{(1+(k-1)n)}$ and we have to prove that $S^{c\ell}_{(1+(k+1)n)} \subseteq S^{c\ell}_{(1+kn)}$.
In fact, we have $S^{c\ell}_{(1+kn)} = Reach_{\Sigma_u}(Img(S^{c\ell}_{kn}, \sigma_n))$ and $S^{c\ell}_{(k+1)n} \subseteq S^{c\ell}_{kn}$ (by following the same logical reasoning as above). Then, $S^{c\ell}_{(1+(k+1)n)} = Reach_{\Sigma_u}(Img(S^{c\ell}_{(k+1)n}, \sigma_n)) \subseteq Reach_{\Sigma_u}(Img(S^{c\ell}_{kn}, \sigma_n)) = S^{c\ell}_{1+kn}$. ■

The above-mentioned property means that, by ignoring the faulty states generated by the faulty transitions from the normal sets of states into the faulty ones within the same node, one can ensure the (non-strict) inclusion relationship between terms of series $S'^{c\ell}$.

Proposition 4 *For a given F-uncertain cycle $c\ell = a_1, a_2, \ldots, a_n$, its corresponding series $S'^{c\ell} := S'^{c\ell}_1 = S^{c\ell}_1, S'^{c\ell}_2 = S^{c\ell}_{(1+n)}, S'^{c\ell}_3 = S^{c\ell}_{(1+2n)}, \ldots$. Then, $S'^{c\ell}$ reaches a fixed-point.*
i.e., $\exists\, k \in \mathbb{N}$ *s.t.* $\forall i \in \mathbb{N} : S^{c\ell}_{(1+(k+i)n)} = S^{c\ell}_{(1+kn)}$. □

Proof.

By considering the (non-strict) inclusion relationship and that $S'^{c\ell} : \mathbb{N} \to 2^X$, Proposition 4 can be derived directly from *the fixed-point theorem (version Banach-Picard* [Farmakis and Moskowitz, 2013]). ■

Corollary 1 *For a given F-uncertain cycle $c\ell = a_1, a_2, \ldots, a_n$, its corresponding series $S^{c\ell} = S_1^{c\ell}, S_2^{c\ell}, S_3^{c\ell}, \ldots$ becomes periodic (with period n) from a certain index.* □

Proof. By Proposition 4 , let k be the index of the fixed-point reached by $S'^{c\ell}$. Then $\forall i \in \mathbb{N}$, $S'^{c\ell}_{(k+i)} = S'^{c\ell}_k$, that is $\forall i \in \mathbb{N} : S^{c\ell}_{1+(k+i)n} = S^{c\ell}_{1+kn}$. Thus, $\forall j : 0 \leq j \leq n \Rightarrow S^{c\ell}_{(1+(k-1)n+j)} = S^{c\ell}_{(1+kn+j)} = \cdots = S^{c\ell}_{(1+(k+i)n+j)} = \cdots$. ■

Corollary 2 *Let $c\ell$ be an F-uncertain cycle, If $c\ell$ is not an F-indeterminate one then, its corresponding series $S^{c\ell}$ reaches an empty fixed-point within a finite delay.* □

Theorem 4 *An F-uncertain cycle $c\ell = a_1, a_2, \ldots, a_n$ in $\mathcal{D}$ is an F-indeterminate cycle if and only if the fixed point reached by its corresponding series $S'^{c\ell}$ is non-empty.* □

Proof. Let $c\ell = a_1, a_2, \ldots, a_n$ be an *F-uncertain* cycle in $\mathcal{D}$ with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$.

($\Rightarrow$) We assume that $c\ell$ is an *F-indeterminate* cycle, and we have to prove that the fixed point reached by series $S'^{c\ell}$ associated with $c\ell$ is non-empty.

As $c\ell$ is an *F-indeterminate* cycle, this means that some faulty cycles exist in the original model G and share the same observation $(\sigma_1, \ldots, \sigma_n)^*$ with some indistinguishable normal cycles. Let us assume that there exist exactly m faulty cycles. This means that $\exists s_i^j \in \Sigma_u^*, \exists x_i^j \in a_i.\mathcal{X}_F$ such that $x^j_{(i+1)} \in \delta(x_i^j, s_i^j.\sigma_i)$ for $(1 \leq i < n)$ and $x_1^j \in \delta(x_n^j, s_n^j.\sigma_n)$, for $1 \leq j \leq m$ (See proof of Proposition 1). Thus, it is plain that $\forall k \in \mathbb{N}^*, \forall i : 1 \leq i \leq n, \forall j : 1 \leq j \leq m : x_i^j \in S^{c\ell}_{i+nk}$. Therefore, all the terms of $S'^{c\ell}$ are non-empty. Hence, obviously the reached fixed point is also non-empty.

($\Leftarrow$) We assume that series $S'^{c\ell}$ associated with $c\ell$ has a non-empty fixed point and let us prove that $c\ell$ is an *F-indeterminate* cycle. Actually, from Proposition 1, one only needs to prove that a faulty cycle which shares the same observation $(\sigma_1, \ldots, \sigma_n)^*$ exists in the original model G.

From Proposition 4, $\exists\, k \in \mathbb{N}^*$ s.t. $S^{c\ell}_{(1+kn)} = S^{c\ell}_{(1+(k-1)n)}$. Moreover, according to our assumption $S^{c\ell}_{(1+kn)} \neq \emptyset$. Let us assume that $S^{c\ell}_{(1+kn)} = S^{c\ell}_{(1+(k-1)n)} = \{x_1, \ldots, x_m\}$, with $m \in \mathbb{N}$.

We will adopt a reasoning analogous to the one used to prove Proposition 1. From the definition of series $S^{c\ell}$ (see Definition 16), we have $\exists x_i, x_j \in S^{c\ell}_{(1+(k-1)n)}$, $\exists\, t = s_1.\sigma_1.s_2.\sigma_2.\ldots.s_{n-1}.\sigma_{n-1}.\ s_n.\sigma_n$, with $s_{i'} \in (\Sigma_u)^*$ (for $1 \leq i' \leq n$) such that $x_i \in \delta(x_j, t)$,

with $1 \leq i, j \leq m$. Thus, by repeating this procedure to x_i and so on at least m times, one can infer that x_j is certainly visited twice, which means that, at least, one faulty cycle exists in G. Therefore, $c\ell$ is an *F-indeterminate* cycle. ■

4.4 On-the-fly Verification

4.4.1 A Systematic Procedure for Checking Diagnosability

It is worth noticing that for the actual verification of diagnosability, *a systematic procedure*, which is derived directly from Theorem 4, can be performed as follows:

When an F-uncertain cycle $c\ell$ is found in $\mathcal{D}$, then:

- generate the successive terms of series $S^{c\ell}$ (starting from S_1), and for each terms $S_i^{c\ell}$ check the following conditions:
  1. if $S_i^{c\ell} = \emptyset$, then cycle $c\ell$ is not an F-indeterminate cycle and stop the procedure;
  2. else, if $S_i^{c\ell} \neq \emptyset$ and $\exists k \in \mathbb{N} : i = 1 + kn$ (with $n = |c\ell|$), then:
     (a) if $S_i^{c\ell} = S_{(i-n)}^{c\ell}$, stop the procedure, since cycle $c\ell$ is an F-indeterminate;
     (b) otherwise continue.

This procedure is repeated as long as there are F-uncertain cycles in diagnoser $\mathcal{D}$.

It should be noticed that, on the basis of Proposition 4, one can be certain that the above procedure terminates well since a fixed-point will be reached (by $S'^{c\ell}$) within a finite delay.

Example 5 *Let us take once again diagnoser $\mathcal{D}$ of model G depicted in Figure 4.1(b). An F-uncertain cycle $c\ell = a_1, a_2$ exists in $\mathcal{D}$. Thus, let us pick sequence $\rho_4 = S_1^{c\ell}, S_2^{c\ell}, S_3^{c\ell}, S_4^{c\ell}$, which contains the successive terms of series $S^{c\ell}$ (see Figure 4.4). One can observe that $S_4^{c\ell} = S_{2=4-2}^{c\ell} = \{4, 5\} \neq \emptyset$, which means that, according to Theorem 4, the F-uncertain cycle $c\ell$ is also an F-indeterminate cycle. Thus, G is non-diagnosable.*

Example 6 *Let us consider another FSA G' in Figure 4.5 (taken from [Sampath et al., 1995]). The set of observable events is $\Sigma_o = \{a, b, c, d, e\}$ and the set of unobservable events is $\Sigma_u = \Sigma_f = \{f\}$. Diagnoser $\mathcal{D}_{G'}$ corresponding to model G' is depicted in Figure 4.6. One can observe that $\mathcal{D}_{G'}$ has an F-uncertain cycle composed of diagnoser nodes a_1, a_2, a_3 with corresponding event sequence $(bce)^*$. Let as pick sequence $\rho = S_1^{c\ell}, S_2^{c\ell}, S_3^{c\ell}, S_4^{c\ell}, S_5^{c\ell}, S_6^{c\ell}, S_7^{c\ell}$ which contains the successive terms of series $S^{c\ell}$. Since $S_7^{c\ell} = S_2'^{c\ell} = \emptyset$, then according to Theorem 4, the F-uncertain cycle in G' is not an F-indeterminate one. Thus, FSA G is diagnosable.*

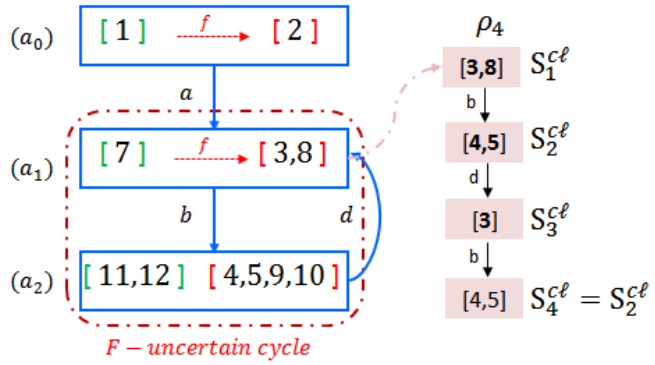


Figure 4.4 – Generation of series $S^{c\ell}$ for analyzing diagnosability

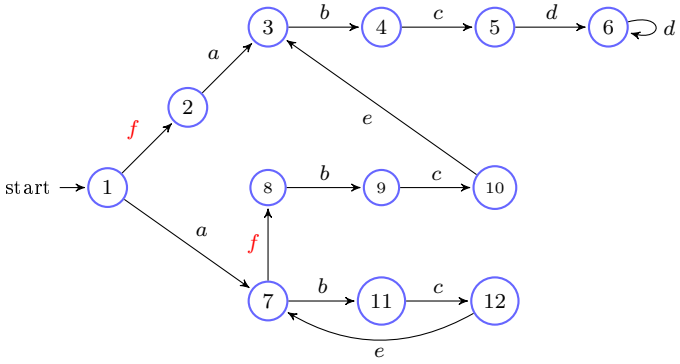


Figure 4.5 – Example of a diagnosable model

4.4.2 Algorithm

In this section, we discuss an on-the-fly algorithm based on a depth-first-search (DFS) procedure for simultaneously constructing the diagnoser and verifying diagnosability. Constructing the diagnoser on the fly serves to avoid the systematic generation of the whole state-space of the diagnoser. On the one hand, if the system is diagnosable, it is unnecessary to construct the part of the diagnoser following F-certain diagnoser nodes, since such a part is unnecessary for analyzing diagnosability and performing online diagnosis. On the other hand, when the system is non-diagnosable, we stop constructing the diagnoser as soon as an F-indeterminate cycle is found. Moreover, the proposed algorithm firstly checks the sufficient condition for non-diagnosability, as proposed in Proposition 2. As soon as this condition is met, the model is stated to be non-diagnosable and the verification process is stopped.

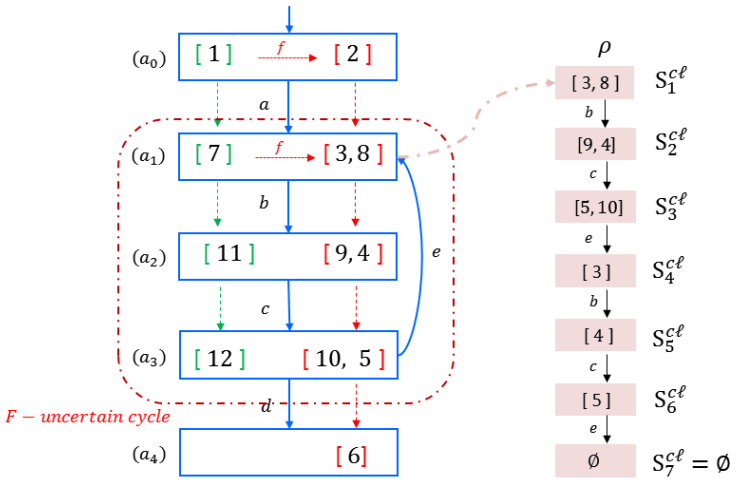


Figure 4.6 – Diagnoser $\mathcal{D}_{G'}$ corresponding to model G' with series $S^{c\ell}$ for analyzing diagnosability

The following functions and data structures will be used in the elaborated algorithm:

- $IsUncertain()$: is a function that returns a Boolean value (*true* if the encountered cycle is composed of only *F-uncertain* diagnoser nodes, and $false$ otherwise).
- List_State, List_State1, Cycle_State: are three finite ordered lists of sets of states. They are used for checking the existence of cycles.
- List_Event, Cycle_Event: are two finite ordered lists of observed events, corresponding respectively to List_State, Cycle_State. They are used to check the existence of cycles.
- Add: an operation that adds an element to an ordered list .
- $RemoveLast(S)$: is an operation that removes the last added element from an ordered list.
- Copy(i, End): is an operation that copies elements from index i to the end of the ordered list, into a new empty ordered list.

The initialization step of the algorithm (cf. Algorithm 1, lines 6-12) serves to compute the initial diagnoser node. Therefore, the construction of the diagnoser nodes is performed by Diagnoser_Construct() function (cf. Algorithm 2). The construction is performed using a depth-first exploration regarding the set of enabled (observable) events from the

Algorithm 1 On-the-fly algorithm to construct the diagnoser and check diagnosability

Input: $G = \langle X, \Sigma, \delta, x_0 \rangle$

Output: Diagnosability verdict

1 Set of states $S_n, S_f, Y_{nf}, Y_f, S'_n, S'_f$
2 Set of Events: Evt_n, Evt_f
3 Boolean: $bool = True$
4 Ordered lists: List_State = {}, List_Event = {}
5 Diagnoser node a, a'
6 **Initialization:**
7 $S_n = Reach_{\Sigma_u \setminus \Sigma_f}(q_0)$
8 $S_f = Reach_{\Sigma_u}(Img(S_n, \Sigma_f))$
9 $\Gamma_0.\mathcal{X}_N = S_n$; $\Gamma_0.\mathcal{X}_F = S_f$; $\Gamma_0.tag = False$
10 $\Gamma = \{\Gamma_0\}$; $a = \Gamma_0$
11 $Evt_n = Enable_{\Sigma_o}(S_n)$; $Evt_f = Enable_{\Sigma_o}(S_f)$
12 List_State = $\{a\}$;
13 **if** *(Diagnoser_ Construct (G, $\mathcal{D}$, Evt_n, Evt_f, S_n, S_f))* **then**
14 **return** Non-diagnosable
15 **return** Diagnosable

current node. With the aim of using the constructed diagnoser to perform the online diagnosis while avoiding the construction of the subsequent F-certain diagnoser nodes, Diagnoser_Construct() function builds only the first encountered F-certain node (if there exists) in the diagnoser path. This is performed by exploring the set of observable events enabled only from the set of faulty states (i.e., $Evt_f \setminus Evt_n$) (Algorithm 2, Lines 2-6). However, for analyzing diagnosability, the exploration is done from the set of observable events enabled from the set of normal states of the current node (i.e., Evt_n). It should be noticed that boolean variable *tag* that is associated with each diagnoser node and initialized to $false$, is used for checking the sufficient condition given in Proposition 2.

The computation of a new node a', reachable through an observable event σ from node a, is completed by the '*Reach*' operation upon the unobservable events (c.f. Algorithm 2, lines 7-14). If diagnoser node a' has already been encountered then the diagnoser is updated by only adding a new transition. In this case, function $IsUncertain()$ checks if node a' belongs to an *F-uncertain* cycle (cf. Algorithm 2, line 20). If so, the sufficient condition (Proposition 2) is firstly checked by analyzing if all the tags of the diagnoser nodes in this F-uncertain cycle are *true* (c.f. Algorithm 2, lines 21 and 22), which means that the F-uncertain cycle is an F-indeterminate one.

Algorithm 2 Diagnoser_Construct() function

Input: G, $\mathcal{D}$, Evt_n, Evt_f, S_n, S_f

Output: Boolean value

Function *Diagnoser_ Construct():*
- **foreach** $(\sigma \in Evt_f \backslash Evt_n)$ **do**
  - $a'.\mathcal{X}_N = \emptyset$
  - $a'.\mathcal{X}_F = Reach_{\Sigma_u}(Img(S_f, \sigma))$
  - **if** $(\exists\, a'' \in \Gamma \mid a'' = a')$; **then** $a'' = \delta(a, \sigma)$
  - **else** $\Gamma = \Gamma \cup \{a'\}$, $a' = \delta(a, \sigma)$
- **foreach** $(\sigma \in Evt_n)$ **do**
  - $S'_n = Reach_{\Sigma_u \backslash \Sigma_f}(Img(S_n, \sigma))$
  - $Y_{nf} = Reach_{\Sigma_u}(Img(S'_n, \Sigma_f))$
  - $Y_f = Reach_{\Sigma_u}(Img(S_f, \sigma))$
  - $a'.\mathcal{X}_N = S'_n$
  - **if** $(Y_{nf} = \emptyset)$ **then** $a'.\mathcal{X}_F = S'_f = Y_f$, $a'.tag = true$
  - **else** $a'.\mathcal{X}_F = S'_f = Y_{nf} \cup Y_f$
  - List_Event.Add(σ)
  - **if** $(\exists\, a'' \in \Gamma \mid a'' = a')$; **then**
    - $a'' = \delta(a, \sigma)$
    - **if** $(a' \in$ List_State$)$ **then**
      - Cycle_State = List_State.Copy($Index(a'), End$)
      - Cycle_Event = List_Event.Copy($Index(a'), End$)
      - **if** *(IsUncertain(Cycle_ State))* **then**
        - **foreach** $(a \in$ Cycle_State **do**
          - **if** *(a.tag = false)* **then** *bool = false*
        - **if** *(bool = true)* **then return** *true*
        - **if** *(Check_ Series(G, a′, Cycle_ Event))* **then**
          - **return** *true*
    - **else** List_State.Add(a')
  - **else**
    - $\Gamma = \Gamma \cup \{a'\}$, $a' = \delta(a, \sigma)$
    - $Evt'_n = Enable_{\Sigma_o}(S'_n)$, $Evt'_f = Enable_{\Sigma_o}(S'_f)$
    - **Diagnoser_Construct** $(\boldsymbol{G}, \boldsymbol{\mathcal{D}}, \boldsymbol{Evt'_n}, \boldsymbol{Evt'_f}, \boldsymbol{S'_n}, \boldsymbol{S'_f})$
  - RemoveLast(List_State)
  - RemoveLast(List_Event)
- return *false*

Thus, Diagnoser_Construct() function outputs boolean value *true* and the verification process is stopped. If the sufficient condition is not satisfied, the necessary and sufficient condition for diagnosability is checked using the procedure proposed in Theorem 4 (cf. Algorithm 3).

Algorithm 3 Check_Series() function

Input: G, a', Cycle_Event, Int i, n

Output: Boolean value

```
1 Function CHECK_SERIES()
2     S_1 = a'.X_F, i = 1, n = |LIST_EVENT|,  LIST_STATE1.ADD(S_1)
3     while (S_i ≠ ∅) do
4         S_(i+1) = Reach_Σu(Img(S_i, σ_((i+1)mod n)))
5         if (i % n = 0) then
6             if ((i ≥ n)&(S_(i+1) = S_(i−n+1))) then return (true)
7             else LIST_STATE1.ADD (S_i), i++
8     return (false)
```

Indeed, function *Check_Series*() (c.f. Algorithm 3) is launched in order to compute the successive terms of series $S^{c\ell}$ associated with the F-uncertain cycle. If a (non-empty) computed term S_i is equivalent to term $S_{(i-n)}$ $(i > n)$ already computed, then boolean value *true* is returned,, otherwise, if the generated term is empty, then boolean value *false* is returned. If the returned value by function *Check_Series*() is '*true*', then the *F-uncertain* cycle is an *F-indeterminate* one. Thus, Algorithm 1 outputs that the model is non-diagnosable and the diagnoser construction is stopped. Otherwise, construction is continued in a recursive manner. Once all the branches of interest are constructed and explored with no F-indeterminate cycle having been met, Algorithm 1 outputs that the model is diagnosable.

4.4.3 A Heuristic Strategy to Improve the Building Algorithm

Our algorithm for constructing the diagnoser (and checking diagnosability) is based on a depth-first search (DFS) to investigate the state-space. However, no rules are defined to select the execution to be investigated first, i.e., the order of exploring executions is arbitrary. In fact, in our case, the diagnoser node structure provides some information that can be exploited to direct the search in such a way as to increase the chances of quickly obtaining a diagnosability verdict by exploring the most "*promising*" executions at first.

When we deal with diagnosability analysis, the interesting executions of the system are those which share the same observed event-sequence such that some of them contain a faulty event and the others are fault-free. This amounts to track the observed event-sequences, in the variant diagnoser, leading to *F-uncertain* nodes. Generally, there exists three types of enabled transitions from any nodes, as depicted in Figure 4.7.

1. Observable events generated only from states in the faulty set (i.e., $a.\mathcal{X}_F$) (Figure 4.7 (a)). In fact, this trace generates only F-certain nodes.
2. Observable events generated only from states in normal set (i.e., $a.\mathcal{X}_F$)) (Figure 4.7 (b)). In this case, we need to continue the construction through this branch since other faults may occur in the future.
3. Observable events generated from both normal and faulty sets of states (Figure 4.7 (c)). In this case, the next reachable diagnoser node will be certainly *F-uncertain*.

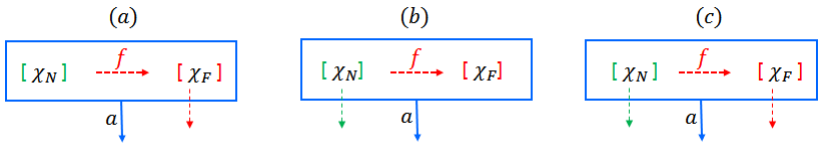


Figure 4.7 – Types of enabled transitions from a node

This last type of transitions is the most-promising as one seeks for F-indeterminate cycles, since it is known, a priori, that the new diagnoser node will be certainly an F-uncertain node, contrary to the other above cases. Thus, such a branch will be the first to be explored in order to direct the construction of the diagnoser and to potentially speed up the verification process.

4.4.4 Complexity Analysis

The construction of the diagnoser variant has the same theoretical complexity as the other diagnoser-based approaches [Sampath et al., 1995, Cabasino et al., 2014, Zad et al., 2003], which means that, in the worst case, such a procedure is *exponential* in the cardinality of state-space of the system model. Actually, this is unavoidable when working with diagnoser-based approaches, since it is due to the deterministic nature of the diagnoser. However, the main underlying idea behind our approach is to avoid as much as possible such a case, i.e., to avoid generating the whole state-space of the diagnoser. In this respect, thanks to the proposed structure of diagnoser nodes and the on-the-fly construction

and verification procedure, our approach shows many advantages regarding the classic diagnoser-based approaches. Firstly, no intermediate model is needed, either for construction the diagnoser or for analyzing diagnosability. This feature reduces significantly memory/time consumption. Secondly, only the necessary part for analyzing diagnosability and performing online diagnosis is generated. Thirdly, diagnosability analysis is performed directly on the F-uncertain cycles with no need to start the investigation form the initial state. Finally, building the diagnoser and checking diagnosability, simultaneously on the fly serves to significantly speed up the verification process, particularly in the case of non-diagnosable models. Indeed, in this case the verification process is stopped immediately after an F-indeterminate cycle is found.

4.5 Extensions

In this section, we discuss some extensions of the proposed approach in order to deal with further fault diagnosis issues, namely online diagnosis, $\mathcal{K}/\mathcal{K}_{min}-$*diagnosability* and the case of multiple fault classes.

4.5.1 Online Diagnosis

Once a system model is checked to be diagnosable, the constructed part of our diagnoser is sufficient to perform online diagnosis. To do so, one keeps only the fault tag information in each diagnoser node (i.e, 'F' for *'faulty'* nodes, 'N' for *'Normal'* ones, and 'U' for *'F-uncertain'* ones). In fact, the online diagnoser is a directed graph where each node carries a fault tag and each transition is tagged with an observable event. Thus, for any sequence of observable events, by following the corresponding path in the diagnoser and according to some predefined rules pertaining to F-uncertain cycles, one can determine, online, the system status.

4.5.2 $\mathcal{K}/\mathcal{K}_{min}-$*diagnosability*

In addition to the analysis of the *'classical'* diagnosability, the proposed diagnoser can be used to deal with the practical concepts of diagnosability, namely $\mathcal{K}-$diagnosability (i.e., diagnosability in $\mathcal{K}$ steps) and $\mathcal{K}_{min}-$*diagnosability* (i.e, the smallest value of $\mathcal{K}$ ensuring diagnosability) [Liu et al., 2014b, Dallal and Lafortune, 2010, Cabasino et al., 2012a, Basile et al., 2012a], in the case where the investigated model is diagnosable.

In fact, we extended the diagnosability analysis procedure in order to cope with $\mathcal{K}/\mathcal{K}_{min}-$ *diagnosability* analysis, in the case where the model is diagnosable. The main idea is to count the maximum number of possible successive F-uncertain nodes in diagnoser $\mathcal{D}$. Three cases can be encountered:

– *Case 1*: a non-cyclic maximal sequence of successive F-uncertain nodes: in this case, the number of these nodes is considered;

– *Case 2*: an F-uncertain cycle $c\ell$ where all the predecessors and successors of its nodes, which do not belong to $c\ell$, are N (or F)-certain: in this case, the index of the earliest empty term (fixed-point) of series $S^{c\ell}$ is considered;

– *Case 3*: an F-uncertain cycle $c\ell$ holding nodes that have some F-uncertain predecessors and/or successors, which do not belong to $c\ell$: in this case, both the index of the earliest empty term of series $S^{c\ell}$ (By Theorem 4 and Corollary 2, since the model is diagnosable, we are certain that $S'^{c\ell}$, so also $S^{c\ell}$, reach an empty fixed-point) and the length of the successive F-uncertain predecessors and successors, are considered.

These different cases are depicted in Figure 4.8, with 'F' for *'faulty'* nodes, 'N' for *'Normal'* ones, and 'U' for *'F-uncertain'* ones.

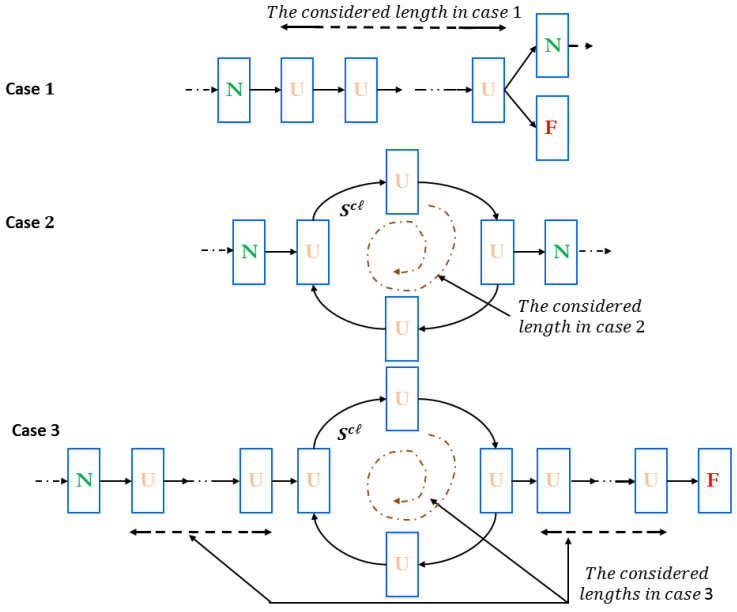


Figure 4.8 – Different cases for analyzing $\mathcal{K}-$diagnosability

4.5.3 Diagnosability of Multiple Fault Classes

In this work, only one class of fault has been considered. However, the proposed approach can be extended to deal with multiple classes of faults. In this case, the set of fault events is partitioned into disjoint fault classes $\Sigma_f = \Sigma_{f_1} \biguplus \Sigma_{f_2} \biguplus \ldots \biguplus \Sigma_{f_m}$, where $\Sigma_{f_i} (i = 1, 2, \ldots, m)$ denotes the i^{th} fault class. Sampath et al. [Sampath et al., 1995] have dealt

with this case of multiple fault classes and proposed a generalization of the necessary and sufficient condition of diagnosability. In fact, a system model is said to be diagnosed if and only if no F_i-indeterminate cycles exist in the diagnoser (with index i corresponds to fault class Σ_{f_i}).

In order to deal with multiple fault classes using the developed diagnoser, one first needs to extend the structure of the diagnoser node. Thus, for a system containing m fault classes, each diagnoser node a contains $m+1$ subsets of states (some of them may be empty): $a.\mathcal{X}_N$ which is the set of normal states, $a.\mathcal{X}_{F_i}$ which is the set of F_i-faulty states reached by event-traces which have experimented, at least, one fault event from fault class Σ_{f_i}, for $i : 1 \leq i \leq m$. These various sets of states may be linked to each other by various faulty transitions, e.g., set of faulty states $\mathcal{X}_{F_2}$ may be linked by faulty transitions (i.e., transitions labeled with fault events in Σ_{f_2}) originated from $a.\mathcal{X}_N$, $a.\mathcal{X}_{F_1}$, $a.\mathcal{X}_{F_3}$,... or $a.\mathcal{X}_{F_m}$. Figure 4.9 depicts the general form of a diagnoser node.

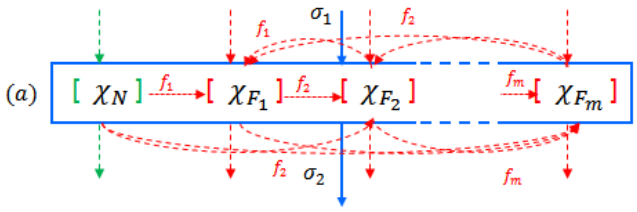


Figure 4.9 – The structure of diagnoser nodes for multiple fault classes

Figure 4.10 illustrates the diagnoser node construction for a part of system model containing, while considering two fault classes ($\Sigma_{f_1} = \{f_1\}$ and $\Sigma_{f_2} = \{f_2\}$). The different sets of states are $a.\mathcal{X}_N = \{1\}, a.\mathcal{X}_{F_1} = \{2,3\}$, and $a.\mathcal{X}_{F_2} = \{3,4\}$. One can observe that state 3 belongs to both $a.\mathcal{X}_{F_1}$ and $a.\mathcal{X}_{F_2}$, which is due to the fact that this state is reached following the occurrence of both f_1 and f_2 in the same event sequence. Thus, it is duplicated with respect to the fault classes involved. Actually, this feature of duplicating states is crucial for the diagnosability investigation, since it allows for analyzing each set of faulty states *separately*.

In fact, regarding the diagnosability analysis, the systematic procedure, proposed in Section 4.4.1, can be extended in order to check the existence of the F_i-indeterminate cycles. The main idea is to generate for each F_i-uncertain cycle $c\ell$, its corresponding series $S_i^{c\ell}$ and $S_i'^{c\ell}$. Then, in the same way as in the case of one fault class, one has to check the fixed-point reached by series $S_i'^{c\ell}$. Moreover, by considering that all the entering transitions to a set of faulty states $\mathcal{X}_{F_i}$ from any other state within the same node are viewed as if they are issued from the normal set of states (with regards to Σ_{f_i}), then the

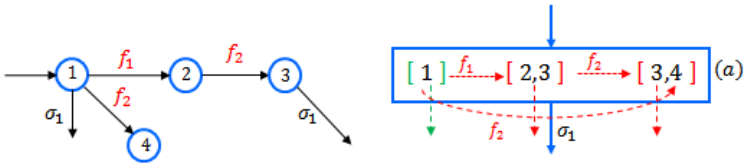


Figure 4.10 – An illustrative example of the diagnoser node with two fault classes

sufficient condition for diagnosability, developed in Proposition 2, can be adapted for the case of multiple fault classes in order to speed up the verification process.

4.6 Experimental Evaluation

In order to assess the effectiveness and the scalability of the proposed approach, the developed algorithm for constructing the diagnoser and verifying diagnosability on the fly (c.f. Section 4.4, has been implemented in C# programming language and tested using a Petri net benchmark. The considered model illustrates the concept of permanent fault concept and fulfills the assumptions considered in this work.

The obtained results are discussed with respect to some reference approaches, namely Sampath's diagnoser [Sampath et al., 1995] and the verifier approach [Yoo and Lafortune, 2002b], which are implemented in the UMDES Library [Lafortune, 2000]. Besides the simulation results, we will point out the similarities and differences between our approach and these reference approaches.

4.6.1 Presentation of the Considered Benchmark

The DES benchmark, depicted in Figure 4.11, is a modified version of the WODES Benchmark [Giua, 2008]. This version of the benchmark was also presented in [Hosseini et al., 2013]. It describes a manufacturing plant characterized by three parameters: m, k and b, where:

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

The faulty transitions are indicated by red boxes, while the other transitions are nominal and can be observable or unobservable depending on the experiments we will carry out.

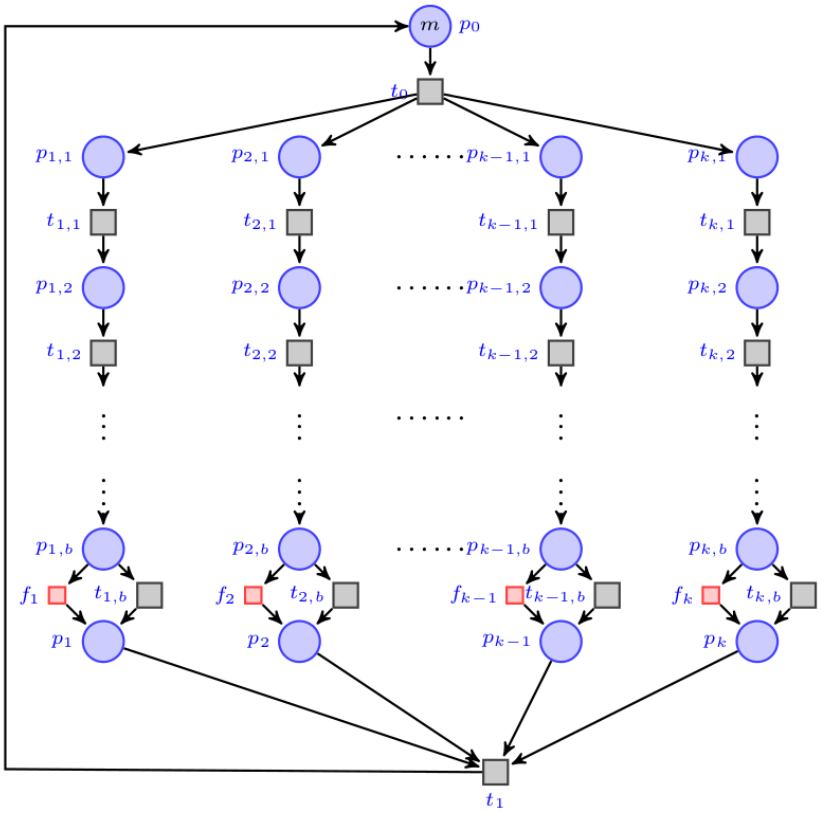


Figure 4.11 – The PN benchamrk

In order to evaluate our approach; two configurations of the benchamrk are considered:

Test 1) in this test, we consider that parameters $m = 1$, $b = 4$, and $k = 3, \ldots, 8$. Moreover, transitions t_0, t_1, and $t_{i,1}$, $t_{i,3}$ for $1 \leq i \leq k$ are observable, and transitions f_i for $1 \leq i \leq k$ are faulty. Regarding the other transitions, two cases are considered :

a) All the other transitions are unobservable (in this case, the model is non-diagnosable).

b) All the other transitions are unobservable except for transitions $t_{i,4}$ for $1 \leq i \leq k$ which are observable (in this case, the model is diagnosable).

Test 2) we evaluate the efficiency of our approach regarding the number of observable and unobservable transitions in the model. In this test, we consider that parameters $m = 1$, $k = 4$, and $b = 2, 3 \ldots, 10, 15, 18$. Transitions t_0, t_1, and $t_{i,b}$ for $1 \leq i \leq 4$ are observable. In addition, transitions f_i for $1 \leq i \leq 4$ are unobservable. Therefore, the model is always diagnosable. Regarding the rest of transitions, two cases are considered:

a) All the other transitions are unobservable.

b) All the other transitions are observable.

In fact, for this test, we only increase the number of unobservable (resp. observable) transitions in the model from one experiment to another. That is, we iteratively add 4 unobservable (or observable) transitions in the model each time.

Before proceeding with the actual experiments, some preparations are necessary. Both approaches, deal with automata models by importing "*.fsm" file. Thus, we first generate the reachability graph (as an automaton) of the considered PN with the help of TINA (TIme petri Net Analyzer) tool [Berthomieu et al., 2007], which is a toolbox for the editing and analysis of Petri Nets. This yields to a "*.aut" file, which is then transformed into a ".fsm" file with the help of a script we have developed. This ensures that the comparative experiments are performed with the same input. It is worth noticing that the experiments have been performed on 64-bit PC (CPU: Intel Core i5, 2.5 GHz, RAM: 6GB). We fix 4 hour as a maximum analysis duration, i.e., if after 4 hours, no results is output by the tool, then we stop the experiment and we consider that the tool fails the analysis.

4.6.2 Results

The experimental results are summarized in Table 4.1 for Test 1, and in Table 4.2 for Test 2, where :

— m, k, and b are the basic structural parameters of the benchmark. They represent respectively the number of tokens in place p_0, the number of production lines, and the number of operations along each production line;

— $|P|$ and $|T|$ are respectively the number of places and transitions of the PN model;

— $|R_S|$ and $|R_T|$, which are the number of states and transitions of the reachability graph respectively, give the scale of the PN reachability graph computed by TINA Tool [Berthomieu et al., 2007]. The reachability graph serves, indeed, as the input automaton model;

— $|D_S|$ and Te_D, are the results obtained by the diagnoser approach [Sampath et al., 1995], and represent respectively: the number of nodes in the diagnoser and the elapsed time for generating the diagnoser and analyzing the diagnosability. Theses results are obtained using *diag.exe, diag_ UR.exe, dcycle.elf* functions of the UMDES Library.

— $|\mathcal{D}_S|$ and $Te_{\mathcal{D}}$ are the results obtained using our approach, and represent respectively: the number of constructed nodes in the diagnoser and the elapsed time for generating the diagnoser and checking the diagnosability;

— V_e is the elapsed time for checking diagnosability using the verifier approach. The result is obtained using *verifier_ dia.exe* function of UMDES Library. Let us note that for this approach, the number of states in the verifier are not provided since the constructed model is not a diagnoser and thus, the comparison in terms of state-space size is not appropriate in this case. Thus, we only give the computation time related to the verifier technique.

— Diag is the diagnosability verdict.

Table 4.1 – Experimental results for Test 1

					Our approach		Sampath's approach		verifier	
k	$\|P\|$	$\|T\|$	$\|R_S\|$	$\|R_T\|$	$\|\mathcal{D}\|$	$Te_{\mathcal{D}}$ (s)	$\|D\|$	Te_D (s)	Te_V (s)	Diag
3	16	17	126	377	16	$\simeq 0$	56	0.3	0.2	non-diagnosable
4	21	22	626	2502	20	$\simeq 0$	164	0.4	4.4	
5	26	27	3126	15627	24	0.1	488	4.4	o.t.	
6	31	32	15626	93752	28	0.3	1460	1893	o.t.	
7	36	37	78126	546877	32	1.5	*	o.t.	o.t.	
8	41	42	390626	3125002	36	9.3	*	o.t.	o.t.	
3	16	17	126	377	65	$\simeq 0$	270	0.2	0.1	diagnosable
4	21	22	626	2502	257	$\simeq 0$	1378	0.3	1.3	
5	26	27	3126	15627	1025	0.2	6686	1	163.4	
6	31	32	15626	93752	4097	1.8	31314	31.4	o.t.	
7	36	37	78126	546877	16385	41.6	143086	1471	o.t.	
8	41	42	390626	3125002	65537	921	*	o.t.	o.t.	

*: No result obtained in 4 hours. o.t.: Out of time (simulation time $\geq$ 4 hours).

4.6.3 Discussion

Regarding the obtained results, presented in Table 4.1, the following remarks can be underlined:

— One can observe that, for all the considered parameters, our approach performs successfully the diagnosability analysis for both diagnosable and non-diagnosable cases while the two other approaches fall starting from a certain range.

— For the non-diagnosable case, the elapsed time for analyzing diagnosability using our approach remains in the order of milliseconds, in spite of the increasing state space of the model. This is particularly due to the on the fly verification technique, which stops constructing the diagnoser as soon as the diagnosability condition is violated. Contrarily, the classic approaches (diagnoser and verifier) firstly build the hole diagnoser/verifier before checking the diagnosability condition, which considerably affects the computation time.

— Our proposed approach remains efficient, in terms of computation time, compared to the two other approaches, when the model is diagnosable. This can be explained by the following three points:

1. in our approach there is no need to construct any intermediate model as in the case of the classic approaches;
2. only the relevant part of the diagnoser for analyzing diagnosability is constructed. That is, the subsequent following F-cetrain nodes (i.e., the certainly faulty part of the diagnoser) is not constructed;
3. the systematic procedure defined for checking diagnosability on the basis of our diagnoser variant allows for enhancing the verification time.

Thus, the generated state-space of the our diagnoser is lower than those of the classic approaches as shown in Table 4.1 (e.g., for $k = 6$ in the diagnosable case, we have 31314 states for the classic diagnoser compared in the 4097 states for our diagnoser).

— The results show that the classic diagnoser approach is more efficient than the verifier approach for the considered benchmark. This does not violate the claim that the verifier approach is more efficient in terms of time complexity (polynomial complexity for the verifier approach versus exponential for the diagnoser approach), since the theoretical complexity is computed while considering the worst case.

For the evaluation results of the approaches regarding the increasing of observable/ unobservable transitions (Table 4.2), the following observation can be made:

— The state-space of the diagnosers are not affected when the number of the unobservable transitions is increased. However, they are very sensitive to the increasing of the observable transitions. This is due to the diagnoser structure.

— The efficiency of Sampath's diagnoser approach and the verifier approach highly decreases when the number of unobservable transitions increases. Regarding the Sampath's diagnoser, this is due to the ε-reduction operation when generating the

Table 4.2 – Experimental results for Test 2

					Our approach		Sampath's approach		verifier
b	$\|P\|$	$\|T\|$	$\|R_S\|$	$\|R_T\|$	$\|\mathcal{D}\|$	$Te_{\mathcal{D}}$ (s)	$\|D\|$	Te_D (s)	Te_V (s)
2	13	14	82	326	17	$\simeq 0$	34	0.1	0.6
3	17	18	257	1026	17	$\simeq 0$	34	3000	46
4	21	22	626	2502	17	$\simeq 0$	*	o.t.	1440
5	25	26	1297	5186	17	0.1	*	o.t.	o.t.
6	29	30	2402	9606	17	0.2	*	o.t.	o.t.
7	33	34	4097	16386	17	0.5	*	o.t.	o.t.
8	37	38	6562	26246	17	0.7	*	o.t.	o.t.
9	41	42	10001	40002	17	1.7	*	o.t.	o.t.
10	45	46	14642	58566	17	3	*	o.t.	o.t.
14	61	62	50626	202502	17	8.4	*	o.t.	o.t.
18	77	78	130322	521286	17	28.8	*	o.t.	o.t.
2	13	14	82	326	82	$\simeq 0$	164	$\simeq 0$	0.4
3	17	18	257	1026	257	$\simeq 0$	514	$\simeq 0$	0.4
4	21	22	626	2502	626	$\simeq 0$	626	0.1	0.5
5	25	26	1297	5186	1297	0.1	0.4	1.2	1.2
6	29	30	2402	9606	2402	0.3	4804	0.7	2.7
7	33	34	4097	16386	4097	1	8194	2.1	9.3
8	37	38	6562	26246	6562	3	13124	3.4	31
9	41	42	10001	40002	10001	8.9	20002	8.5	71.6
10	45	46	14642	58566	14642	15.4	29284	25.3	154.4
14	61	62	50626	202502	50626	206	101252	363	1590
18	77	78	130322	521286	130322	1382	260644	2340	8520

*: No result obtained in 4 hours. o.t.: Out of time (simulation time $\geq$ 4 hours).

intermediate model (the generator), which highly depends on the number of unobservable events in the model. Regarding the verifier approach, this is due to the various rules used for building the verifier, i.e., for one unobservable transition in the system model, three distinct transitions are considered in the verifier (see the construction rules of the verifier in [Yoo and Lafortune, 2002b]).

— It is worthwhile noticing that the UMDES tool (both the diagnoser and the verifier techniques) falls with some models in this test while our tool performs successfully the diagnosability analysis of all the model variants. The efficiency of our technique comparatively to the two UMDES techniques is more noticeable as the part of unobservable transitions becomes more important in the model.

4.7 A Comparison Between Sampath's Diagnoser and our Proposed Diagnoser

The new diagnoser-based approach, we have discussed in this chapter, has some similarities and differences with Sampath's diagnoser approach that are discussed in what follows:

- Both approaches deal with fault diagnosis of DESs in the automata formalism under the same assumptions, (i.e., the model is live and no unobservable cycle exist in it).
- Both approaches are constructed in an exponential complexity, due the determinization nature of the diagnoser. However, in our approach, this high complexity is partially tackled using the partial construction of the diagnoser state-space, due to the on the fly building/verification algorithm.
- The main advantages of our approach w.r.t. to Sampath's diagnoser are twofold:
  1. Only the necessary part for analyzing diagnosability and performing the online diagnosis is generated. Indeed, the diagnoser construction and diagnosability analysis are simultaneously performed on-the-fly using a depth-first-search algorithm. Therefore, the diagnoser need not be built completely for checking diagnosability. This would significantly save memory, particularly in the case of non-diagnosable models.
  2. In our technique, the systematic procedure for checking the necessary and sufficient condition does not require to return to any intermediate model. Besides, for the analysis of F-uncertain cycles, there is no need to start from the initial state and the analysis is performed directly on the cycle. Such a procedure, significantly speed up the verification process (as shown in the experimentation).
- The main drawback of our approach, is the size of nodes (in terms of memory) in the diagnoser, due to the unobservable reachability. In fact, the nodes of our diagnoser may contain a large number of the model states that are reachable through unobservable sequences after the occurrence of an observable event. Such an issue is not encountered in Sampath's diagnoser, since the unobservable reachability is erased by means of the ε-reduction operation when building the generator. However, the fact that two graphs, namely the generator and the diagnoser, are constructed and memorized still raises the memory issues. In order to overcome the issue of memory consumption in our diagnoser, we intend to bring into play symbolic representation techniques (using Binary Decision Diagrams (BDDs)). In fact, BDDs can efficiently encode and manage the sets of states in the diagnoser nodes. Such a technique

represents the main contribution of Chapter 6 where the considered models are labeled Petri nets.

4.8 Conclusion

This chapter discusses an efficient diagnoser-based approach for fault diagnosis of discrete-event system modeled as partially observed finite state automata. A new structure for representing the diagnoser nodes is proposed. It consists in separating normal states and faulty states in each diagnoser node. Therefore, on the basis of this structure and through using various features that characterize it, we propose a systematic procedure for directly checking diagnosability using the diagnoser and without needing to construct any intermediate model. Moreover, the proposed procedure also considers dealing with the quantitative diagnosability properties, namely, $\mathcal{K}-$ and $\mathcal{K}_{min}-diagnosability$, as well as online diagnosis. An on-the-fly algorithm for constructing the proposed diagnoser and verifying diagnosability simultaneously is provided.

Chapter 5

A Diagnoser-Based Approach for Intermittent Fault Diagnosis

Sommaire

Summary

In this chapter, we extend the diagnoser-based approach, introduced in the previous chapter, in order to deal with the fault diagnosis of intermittent faults. First, we discuss two ways for modeling the intermittent faults in finite state automata framework. Then, various definitions of diagnosability from the SED literature are revisited. Such definitions concern the occurrence of the fault events, their recovery, and the identification of the system status within finite delays. In order to analyze such properties, we extend the proposed diagnoser (discussed in Chapter 4) with a new structure, which consists in separating normal states, faulty states and recovered ones in each diagnoser node. Furthermore, necessary and sufficient conditions for checking the various diagnosability properties are derived on the basis of the diagnoser structure and systematic procedures for checking such conditions without needing any intermediate model is proposed.

The work presented in this chapter is the subject of publications in ICPHM'16 [Boussif and Ghazel, 2016b]*, WoDES* [Boussif et al., 2016c] *and a submitted one on ACC'17* [Boussif and Ghazel, 2017].

This chapter is structured as follows: Section 5.1 gives a general introduction regarding intermittent faults and their importance in the fault diagnosis of real-life systems. In Section 5.2, a brief literature review on intermittent fault diagnosis of DESs is provided. Section 5.3 presents an overview on the approach we discuss in the current chapter. Section 5.4 introduces the system modeling and two ways for modeling intermittent faults. In Section 5.5, various notions of diagnosability of intermittent faults are introduced and illustrated. Section 5.6 discusses the construction of the diagnoser and presents some associated features that will be advantageously exploited in the sequel. Section 5.7 is devoted to the reformulation of the necessary and sufficient conditions regarding the 'weak' versions of diagnosability and the development of a systematic method for checking these conditions. Necessary and sufficient conditions regarding the strong versions of diagnosability of intermittent faults are established in Section 5.8. In section 5.9, a discussion regarding the related works is provided. Finally, Section 5.11 draws some concluding remarks.

5.1 Introduction

A fault is defined to be any deviation of a system from its normal or intended behavior and fault diagnosis is the task that consists in detecting/identifying the abnormality in the system behavior and isolating the cause or the source of this abnormality. In the DESs framework, faults are basically depicted as *unobservable/silent*, *indistinguishable* and *uncontrollable* event or states, which means that the failure cannot be detected/identified directly.

Faults can be classified on the basis of their individual behavior into three types [Sharma et al., 2015, Zaytoon and Lafortune, 2013]:

1. **Permanent faults:** fault occurs but does not disappear (such that the system remains in faulty state) until repairing measures are undertaken. Typically a permanent fault is caused by subsystem failures, physical damage or design error.

2. **Incipient faults (also called gradual or drift-like faults):** fault varies gradually and slowly develops into an enormously large value. Diagnosis of such faults is more difficult than the case of permanent faults because they evolve very slowly and their effects can be confused with noise and uncertainty.

3. **Intermittent faults:** fault occurs and then suddenly disappears and this process continues to happen in a repeated manner. Therefore, the system switches between normal and faulty states. Such faults may be activated or deactivated by some external disturbances [Isermann, 2006].

From the diagnosis point of view, it is important to distinguish between these fault types, especially between the permanent and intermittent faults. The intermittent faults can be spontaneously recovered (by the occurrence of uncontrollable and unobservable reset event), therefore the system oscillates between normal and faulty behavior. Permanent faults, on the other hand, may be associated with recovery events (repair/replacement) which are controllable and observable, and the system can not spontaneously move from a fault state to a non-faulty one [Huang, 2003].

In a major part of the literature regarding model-based fault diagnosis (MBD), faults are typically assumed to be permanent. However, experience with fault diagnosis of real-life systems shows that intermittent faults are predominant and are among the most challenging kinds of faults to detect and isolate [Fromherz et al., 2004]. Indeed, according to [Shen et al., 2016], intermittent faults exist in many systems, including those ranging from small components to huge complicated equipment. The frequent occurrence of intermittent faults can bring on serious troubles and result in high safety risk and important maintenance cost. In the late 1960s, Hardie [Ball and Hardie, 1969, Hardie and Suhocki, 1967] indicated that intermittent faults comprised over 30% of pre-delivery failures and almost 90% of field failures in computer systems. Roberts [Roberts, 1989] stated that between 80 and 90% of system faults are caused by intermittent faults. Banerjee [Banerjee and Khilar, 2010] indicated that in wireless sensor networks intermittent faults are the most frequently occurring. Intermittent faults bring on many maintenance problems, such as False Alarms (FAs), No Fault Found (NFF), Can Not Duplicate (CND) and so on [Sorensen et al., 1994]. In 2012, a survey among 80 aerospace organizations ranked intermittent faults as the highest perceived cause of NFF [Syed et al., 2013]. The NFF problem has been the highest cost source in aerospace maintenance. For example, the annual NFF exchange cost of the F-16 avionics boxes due to intermittent faults was over $ 20,000,000 [Steadman et al., 2005, Steadman et al., 2008].

According to [Correcher et al., 2003, Correcher et al., 2010], intermittent faults can be modeled using four parameters. These indicators report useful information to the supervisor regarding preventive maintenance and control reconfiguration.

1. *Duration (D_i):* is defined as the time during which the fault is active in each one of its occurrences;

2. *Pseudo-period (Ps):* is defined as the time between two consecutive fault detections. It does not seem reasonable to talk about period, due to the asynchronous nature of faults;

3. *Persistence (Pr):* is defined as the inverse of the pseudo-period. It gives an estimate of the fault frequency;

4. *Number of fault detection (Nf):* represents how many times the fault was detected.

Examples of intermittent faults are: electrical contacts, overheating, overloads, arc faults in the pantograph of a running train [Aydin et al., 2013], some kinds of interruptions and bugs in software systems, etc. In practice, intermittent faults are viewed as a temporary malfunction of a device that compels the system to switch between faulty and non-faulty behavior at discrete random intervals. Intermittent faults are characterized by repetitive occurrences, often with irregular intervals, and separated by reset action that causes the system to switch back to the normal behavior . Applications in various domains are concerned by intermittent fault diagnosis, such as digital circuit testing [Chang and McCluskey, 1997], aerospace industry [Salvatore et al., 2003], aircraft systems [Anderson and Aylward, 1993], modern industrial and chemical processes [Madden and Nolan, 1999, Yan et al., 2015], transportation systems [Aydin et al., 2013] and machine driven systems [Ismaeel and Bhatnagar, 1997, Kim, 2009].

Many definitions of intermittent faults have been proposed in the literature: Sorensen in [Sorensen et al., 1994] defined intermittent faults as any temporary deviation from nominal operating conditions of a circuit or device. In [Syed et al., 2013], they are defined as temporary malfunction of a device. Among these definitions, the environment induced disturbance may also be regarded as intermittent faults. Pan [Pan et al., 2012] regarded intermittent faults as a hardware error which occurs frequently and irregularly for a period of time. According to IEEE [Prasad, 1990], intermittent faults are defined as "*failures of an item for a limited period of time, following which the item recovers its ability to perform its required function without being subjected to any external corrective action. Moreover, such failures are often recurrent*". In the DESs community, intermittent faults are defined as faults which often occur intermittently, and can be depicted with fault events followed by corresponding reset events for these faults, followed by new occurrences of fault events, and so forth [Contant et al., 2004]. Another definition, which is similar to this one, is given in [Deng et al., 2014]: Intermittent faults are defined as failures that can automatically recover once they have occurred. It may be activated or deactivated by some external disturbance. Then, if the disturbance ends, the failure will disappear. The work presented in this Chapter considers these last two definitions which can be seen as an abstraction of intermittent faults description in the DESs framework.

5.2 A Review of Intermittent Fault Diagnosis in DESs

The DES-based fault diagnosis techniques developed to deal with permanent failures are no longer suitable for the analysis of intermittent faults, since the case of intermittent faults

shows some subtle configurations compared to the case of permanent failures. The same remark can be made for the various methodologies developed in the field of model-based reasoning in Artificial Intelligence, as witnessed in [Contant, 2005]. Consequently, some further DES-based frameworks have been proposed to handle intermittent faults. The earlier works in [Aghasaryan et al., 1998, Benveniste et al., 2003], which are developed to deal with permanent failures, allows considering intermittent faults. Nevertheless, these works do not propose a systematic framework for their detection and isolation.

One of the pioneering contributions in intermittent fault diagnosis was made in [Jiang et al., 2003b], where a *state-based* DES modeling for the so-called "*repeated faults*" was introduced. The idea behind this work is that, instead of only diagnosing the fault occurrences, the developed technique considers determining the number of times the fault has occurred. In order to do so, Jiang et al. have introduced three notions of diagnosability: K-diagnosability (for determining that a fault has been occurred K times), $[1, K]$-diagnosability (for 1 to K faults diagnosability), and $[1, \infty]$-diagnosability (for all fault occurrences diagnosability). For checking such diagnosability properties, some polynomial algorithms were provided. Moreover, a polynomial procedure for online diagnosis of such faults is also presented in this work. Improvement in terms of the polynomial complexity for checking $[1, \infty]-$diagnosability have been proposed in [Yoo and Garcia, 2004, Yoo and Garcia, 2009, Zhou and Kumar, 2009], where the complexity is of $\mathcal{O}(|X|^5 \cdot [\Sigma|^2)$ and $\mathcal{O}(min(|X|^3 \cdot [\Sigma|^2, |X|^5))$ for respectively nondeterministic and deterministic models instead of $\mathcal{O}(|X|^6 \cdot [\Sigma|^2)$ and $\mathcal{O}(|X|^4 \cdot [\Sigma|^2)$ in [Jiang et al., 2003b]. An application of such a theoretical framework for diagnosing routing events in discrete flow networks is discussed in [Garcia and Yoo, 2005]

In [Jiang and Kumar, 2006, Jiang, 2002], the above-mentioned framework (of repeated faults) has been reformulated in a temporal logic-based approach. Indeed, Linear-time temporal logic (LTL) formulas were used to express the above-mentioned diagnosability properties. In these works, notions of prediagnosability and diagnosability for failures, firstly introduced for dealing with permanent failures [Jiang and Kumar, 2004], have been extended to deal with repeated faults. Then, polynomial algorithms for testing the prediagnosability and diagnosability have been provided.

These works, discussed above, focused on diagnosing how many times a fault has occurred. However, they do not take into account determining the status of the system at a given moment (i.e., detecting and identifying which faults are present in the system and which ones have been reset). Dealing with the diagnosability of intermittent faults in this sense was firstly discussed in [Contant, 2005, Contant et al., 2002, Contant et al., 2004]. In these works, an event-based reasoning on the basis of finite state automata (FSA) is adopted, namely, faults and their recovery are considered to be unobservable

events. The purpose of these works is to determine, at certain points, which faults are present in the system and which ones have occurred, or have been recovered. These works can be regarded as an extension of the seminal work on diagnosability analysis of permanent failures based on the diagnoser approach [Sampath et al., 1995]. The structure of the diagnoser, in this case, was enriched by new labels to capture the dynamic nature of intermittent faults in the system model. Four notions of diagnosability have been proposed, regarding the occurrence/recovery of faults and the identification (determination) of the faulty/recovered system status within finite delays.

On the basis of the theoretical framework developed in [Contant et al., 2004], the authors in [Correcher et al., 2003] proposed a methodology to diagnose intermittent faults in industrial processes. The classic pump/valve process [Sampath et al., 1995] is simulated in Matlab, employing Simulink for modeling the continuous behavior and Stateflow for the DES diagnoser. In [Carvalho et al., 2013], the authors addressed the issue of diagnosing intermittent sensor faults. In the same spirit as in [Contant et al., 2004], authors have modified the model of intermittent loss of observation to account for sensor malfunction only. Then, the issue of detecting intermittent sensor faults is transformed into an issue of diagnosing intermittent faults. Therefore, a similar diagnoser to the one developed in [Contant et al., 2004] is used to check diagnosability. In the same scope, robust diagnosability against intermittent sensor faults is discussed in [Carvalho et al., 2010]. In such a work, the assumption that not only all sensors work properly but also all information reported by sensors always reaches the diagnoser, is relaxed. That is, some observed events may not reach the diagnoser, which can be seen as an intermittent loss of the observation. A necessary and sufficient condition for robust diagnosability is provided and tested using the diagnoser [Sampath et al., 1995] and the verifier [Yoo and Lafortune, 2002b] techniques.

An extension of the *state-based* DES framework [Zad et al., 2003] was proposed in [Biswas, 2012] to deal with intermittent faults. Two notions of diagnosability were introduced. One for detecting the occurrence of a fault, and the other for detecting its recovery. The diagnoser is built in a similar way as in [Zad et al., 2003] with the same time complexity. Necessary and sufficient conditions for each notion of diagnosability were developed and an algorithm to verify such conditions was provided.

In [Soldani et al., 2007], intermittent fault diagnosis in an FSA framework was reported. The particularity of such a work is that only the normal behavior of the system is considered, i.e., a fault-free model). Then, faults are modeled as the occurrence of an extra event or as the absence of a specific event. A diagnoser is then established for each event type. An extension to Petri net framework was given in [Soldani et al., 2006].

An overall framework regarding the assessment of intermittent fault probabilities, i.e.,

computing the occurrence probabilities, is developed in [De Kleer et al., 2008, De Kleer, 2009]. The proposed approach aims at developing an efficient general method, referred to as the *General Diagnostic Engine* (GDE), for diagnosing failures due to any number of simultaneous faults. Moreover, the approach computes both the posterior probabilities after some observations are made as well as the additional probes needed to efficiently isolate the fault in the system devices. In [Deng et al., 2014], a new fault model which includes both permanent and intermittent faults is presented. Thereafter, an approach is given to discriminate between the two fault types and diagnoses any current fault in the system. The authors in [Jéron et al., 2006] have proposed a supervision pattern framework for dealing with fault diagnosis. In fact, such patterns are finite state automata that can describe the occurrence of permanent faults, intermittent faults, the repair of a system after the occurrence of a fault, as well as quite complex sequences of events. Then, the problems of diagnosis correctness and bounded diagnosability are discussed.

Recently (and in parallel to our work), Fabre et al. [Fabre et al., 2016] have proposed an approach to deal with repairable faults. Diagnosability in this work means that the occurrence of a fault should always be detected in bounded delay, but also before the fault is repaired. The authors have also showed that checking such a notion of diagnosability is PSPACE-complete and have proposed an augmented diagnoser (by performing the parallel composition of the classic diagnoser and the original system model) in order to conduct the diagnosability verification.

5.3 Overview on the Developed Contribution

Almost all the works pertaining to intermittent fault diagnosis in DESs mentioned above deal with diagnosability on the basis of structural analysis of the so-called "diagnoser". However, we have shown in the previous chapter (Chapter 4) that, besides the fact that the diagnoser construction shows a high complexity level (exponential) in terms of state-space, two other main issues related to diagnosability analysis can be outlined:

(i) the approach is based on the analysis of two graphs. The first graph is a non-deterministic observer (called pre-diagnoser, or generator), while the second one is a deterministic automaton called diagnoser;

(ii) the diagnoser-based approaches use a *double-checking* procedure, which consists in one verification step on the diagnoser to check the existence of F-*uncertain* cycles and the other one on the generator/pre-diagnoser to check whether the F-*uncertain* cycle is an F-*indeterminate* one or not). In fact, in general this double check procedure greatly increases the verification time.

The work presented in this chapter falls in the framework of diagnoser-based approaches for intermittent fault diagnosis of DESs. It consists in an extension of the approach discussed in Chapter 4 where a new diagnoser structure is proposed to partially overcome the above-mentioned limitations of existing diagnoser-based approaches. The main idea consists in separating the normal, the faulty and the recovered sets of states in each diagnoser node. Such a structure allows for performing diagnosability analysis upon the diagnoser, without needing to construct any intermediate model, which improves the efficiency in terms of the memory and time consumption.

In what follows, we recall the main features of the proposed approach,

1. Separating the normal, the faulty and the recovered states in each diagnoser node allows us to separately track the nominal, the faulty and the recovered traces directly in the diagnoser.

2. In our approach, the diagnoser is directly built from the original system model, without needing to construct any intermediate model, as usually done in the classic diagnoser approaches [Sampath et al., 1995, Contant et al., 2004].

3. The approach provides a systematic procedure for checking the necessary and sufficient conditions for diagnosabilily of intermittent faults without returning to any intermediate model. Besides, when an uncertain cycle is encountered the procedure performs directly on the cycle of the diagnoser. All these aspects allow for significantly speeding up the verification process.

5.4 Modeling of the System and Intermittent Faults

5.4.1 System Model

In this chapter, we consider that the system to be diagnosed is modeled by an FSA $G = \langle X, \Sigma, \delta, x_0 \rangle$. We keep the same notations and notions introduced in Section 3.2. However, we add the following notations regarding intermittent faults:

Let us denote by $\Sigma_f \subseteq \Sigma_u$ the set of fault events and $\Sigma_r \subseteq \Sigma_u$ the set of fault reset events. Faults and their recovery are basically represented using unobservable events. Moreover, the set of fault events (resp. the set of reset events) can be partitioned into disjoint fault classes $\Sigma_f = \Sigma_{f_1} \biguplus \Sigma_{f_2} \biguplus \ldots \biguplus \Sigma_{f_m}$, where $\Sigma_{f_i} (i = 1, 2, \ldots, m)$ denotes one class of faults (resp. $\Sigma_r = \Sigma_{r_1} \biguplus \Sigma_{r_2} \biguplus \ldots \biguplus \Sigma_{r_m}$, where $\Sigma_{r_i} (i = 1, 2, \ldots, m)$ denotes the recovering class of faults in Σ_{f_i}) and $\Sigma_f \cap \Sigma_r = \emptyset$. In the sequel, we will use $\psi(\Sigma_{f_i})$ to denote the set of event-traces in L that end with some faulty event in Σ_{f_i}. That is,

$\psi(\Sigma_{f_i}) := \{s.\sigma_{f_i} \in L : \sigma_{f_i} \in \Sigma_{f_i}\}$. Similarly, we will use $\psi(\Sigma_{r_i})$ to denote the set of event-traces in L that end with some reset event in Σ_{r_i}. That is, $\psi(\Sigma_{r_i}) := \{s.\sigma_{r_i} \in L | \sigma_{r_i} \in \Sigma_{r_i}\}$. Finally, we make the following remark:

Remark 1 *We assume that the occurrence of a reset event σ_r without any corresponding fault event σ_f having occurred, does not affect the system status (i.e., in this case, the reset event is seen as a normal unobservable event).*

Remark 2 *It is worth noticing that regarding the projection mapping, the general setting of the inverse projection P_L^{-1} is not restricted to the event-sequences which end with an observable event (i.e., $\forall \omega \in \Sigma_o^*, P_L^{-1}(\omega) = \{s \in L \subseteq \Sigma^* : P(s) = \omega\} \cap \Sigma^*\Sigma^o$), as it is usually considered in fault diagnosis of permanent faults (for instance, see [Sampath et al., 1995, Contant et al., 2004]). Therefore, in this chapter, the general setting of the inverse projection P_L^{-1} is considered, that is, $\forall \omega \in \Sigma_o^*, P_L^{-1}(\omega) = \{s \in L \subseteq \Sigma^* : P(s) = \omega\}$. In other terms, we consider a -somehow- more general setting since we require that the diagnosis shall consider also any unobservable continuation following the last observed event. Hereafter, an example that explains this slight difference between these two definitions, is given.*

Example 1 *Let us consider the part of automaton in Figure 5.1. We are interesting in the generation of the set of event-sequences sharing the same observation $\omega = a$ using the two definitions above.*

1. $P_L^{-1}(\omega) = \{s \in L \subseteq \Sigma^* : P(s) = \omega\} \cap \Sigma^*\Sigma^o = \{a, ua\}$;
2. $P_L^{-1}(\omega) = \{s \in L \subseteq \Sigma^* : P(s) = \omega\} = \{a, au, auf, ua, uau\}$.

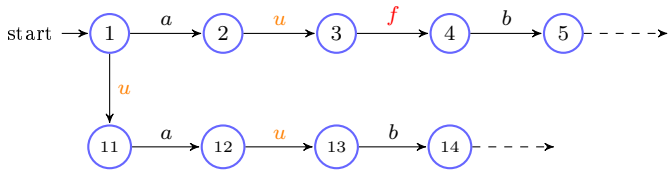


Figure 5.1 – Illustration for the inverse projection mapping in Example 1

Remark 3 *It is worth noticing that this slight difference between the two definitions of the inverse projection operator, does not affect the diagnosis verdict when dealing with permanent faults, i.e., using both definitions, we obtain the same diagnosis verdict. However, in the case of intermittent faults, the choice of the inverse projection operator can have an impact on the diagnosis verdict, as it will be discussed at the end of this chapter.*

For the sake of clarity, only one class of fault event Σ_f and its corresponding class Σ_r of reset events are considered here. Extension to multiple fault classes is performed in the same manner as discussed in 4 (Section 4.5.3).

5.4.2 Intermittent Fault Modeling

In the literature pertaining to the diagnosis of DES, faults are said to be intermittent when they are non-permanent, in the sense that each fault occurrence is followed by its corresponding reset event (i.e., normalization) within a finite delay, which is in turn followed later on by a new occurrences of fault events, and so forth. Such faults may be activated or deactivated by some external disturbances.

Regarding the system status after the occurrence of fault events and their reset events, two modeling formalisms can be distinguished:

5.4.2.1 The Recover formalism

In such a formalism, a distinction is made between the states reached by faulty-free sequences and the states reached by sequences where at least one fault has occurred and reset. Regarding the system status, the occurrence of an intermittent fault occurrence switches the system from a normal state to a faulty state, and thereafter the system is switched again to a recovered state within a finite delay upon the occurrence of the corresponding reset event. Such a recovered state is assumed to be safe. However, it is different from the normal, i.e., it can somehow be seen as a degraded mode. It should be noticed that, in such a formalism, the reset events are also called '*recover*' events.

Such a modeling manner has been used in [Carvalho et al., 2010, Moreira et al., 2011, Carvalho et al., 2013] and also in [Contant, 2005, Contant et al., 2004, Contant et al., 2002] by adding a distinction between the first fault occurrence and the subsequent occurrences.

Due to the various types of events and in order to capture these changes in the system status, we use a supervision pattern $\Omega = \langle \{N, F, R\}, \Sigma, \delta_\Omega, N \rangle$ [Carvalho et al., 2012, Fabre et al., 2016] as shown in Figure 5.2, which is a label automaton that transcribes the system status according to the occurrence of the different event types. Automaton Ω can be seen as a labeling function, which is usually used in fault diagnosis [Sampath et al., 1995].

In fact, when label automaton Ω is in state N (N stands for the normal status), the system executes a normal behavior, which indicates that no event from Σ_f has occurred yet. However, when a fault event occurs, Ω moves to state F (F stands for the faulty status) and remains in this state as long as the system executes a faulty behavior. When the fault is recovered due to the occurrence of a reset event, Ω switches to state R (R stands for the recovered status), where it stays as long as the system continues to execute

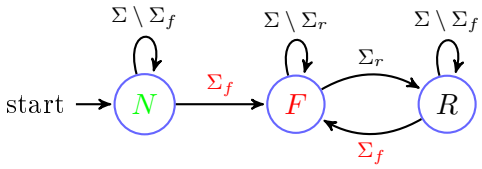


Figure 5.2 – The label automaton Ω in the recover modeling

a non-faulty behavior. As we deal with intermittent faults, the system can again execute a fault event. In this case, the label automaton Ω switches back to state F and so on.

As mentioned in Remark 1, one can notice that a reset event $\sigma_r \in \Sigma_r$ may take place without any fault event $\sigma_f \in \Sigma_f$ having occurred. This is also taken into account by label automaton Ω and the system remains in normal status (state N in Ω) in this case.

In order to keep track of the fault occurrences and their corresponding resets along the system's evolution, we compute automaton G_ℓ as the parallel composition of automata G and Ω ($G_\ell = G \parallel \Omega$). In fact, the parallel composition does not change the language of G, but performs a state augmentation that keeps track of the firing of fault and reset events. That is, the states of G_ℓ are the states of automaton G enriched with labels N, F and R. This technique was generalized in [Jéron et al., 2006] to detect/diagnose some regular pattern of labels, rather than the simple firing of fault/reset events.

Example 7 *Consider automaton G (taken from [Contant et al., 2004]) and shown in Figure 5.3 (a) . The sets of observable and unobservable events are $\Sigma_o = \{a, b, c, d\}$ and $\Sigma_u = \{f, r\}$, respectively. In addition, $\Sigma_f = \{f\}$ and $\Sigma_r = \{r\}$. Automaton $G_\ell = G \parallel \Omega$ is depicted in Figure 5.3 (b).*

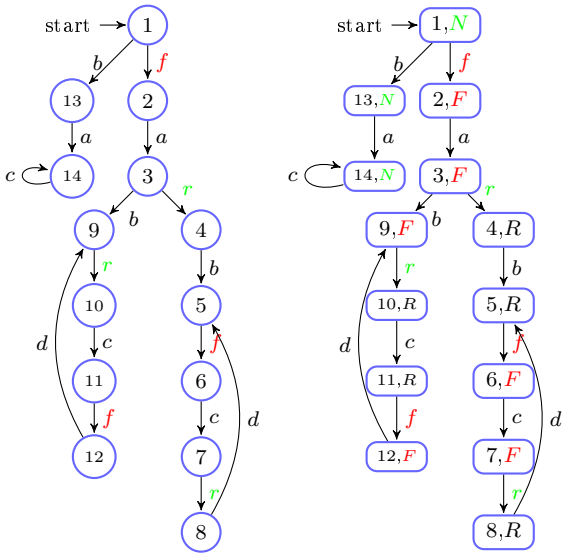
From the monitoring viewpoint, the states of automaton G_ℓ can be partitioned into three subsets: 'Normal', 'Faulty' and 'recovered', which can be identified using *fault-assignment function*:

$$\Psi : X \to \{N, F, R\}.$$

Now, let us define *labeling function* ℓ, $\ell : L(G) \subseteq \Sigma^* \to \{N, F, R\}$ as follows:

Let $s \in L$ be an event-sequence, then:

- $\ell(s) = N \quad if\ (\Sigma_f \notin s)$
- $\ell(s) = F \quad if\ \exists\, s', s'' : (s = s's'') \wedge [s' \in \psi(\Sigma_f)] \wedge\ (\Sigma_r \notin s'')$
- $\ell(s) = R \quad if\ \exists\, s', s'' : (s = s's'') \wedge (\Sigma_f \in s') \wedge [s' \in \psi(\Sigma_r)] \wedge (\Sigma_f \notin s'')$



(a) Automaton G (b) Automaton G_ℓ

Figure 5.3 – Example 7

5.4.2.2 The Normalizing formalism

In such a formalism, a fault event switches the system from a normal state to the faulty one, while its corresponding reset event takes back the system from a faulty state to a normal one. Therefore, only two types of states exist in the system model, since no distinction is made between the normal states reached by faulty-free event sequences and the states reached by event sequences that show a recovered fault, i.e., $s = s's''$ such that $\Sigma_f \in s', s' \in \psi(\Sigma_r)$ and $\Sigma_f \notin s''$.

In fact, in this modeling formalism, it is assumed that the system returns to its nominal behavior as soon as the fault recovers. Typical examples of faults for which such a modeling formalism is appropriate: in the case of circuits, some faults may occur due to high temperatures but disappear once the circuit is cooled. Also, a valve in a flow control system may get stuck closed due to overflow of a liquid. The system may recover from the failure after some repeated efforts by the motor to open the valve which is triggered by the controller when it detects an abnormal pressure rise (because of the closed valve) [Biswas, 2012].

It should be noticed that, in this modeling formalism, the reset events are called '*normalizing*' event and the switch from a faulty state to a normal one is called '*normalization*'.

Figure 5.4 shows the label automaton $\Omega_N = \langle \{N, F\}, \Sigma, \delta_\Omega, N \rangle$ that illustrates the normalizing formalism. In fact, when Ω is in state N, the system executes a normal

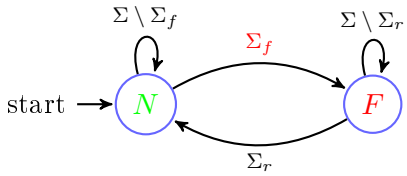
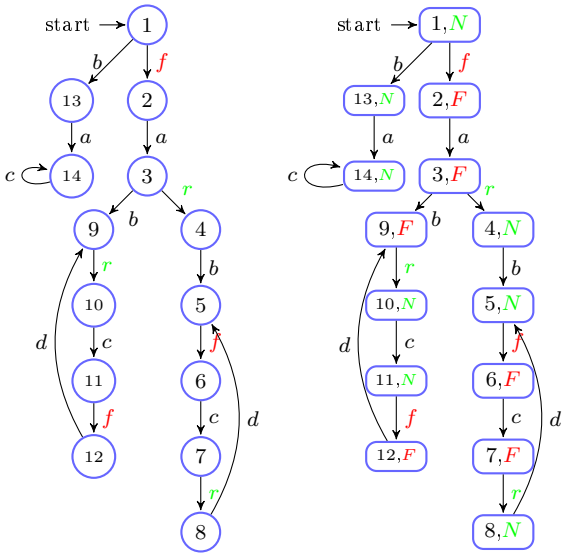


Figure 5.4 – The label automaton Ω_N

behavior, which indicates that no fault event from Σ_f has occurred yet or that any fault that has occurred is normalized, due to the occurrence of a reset (normalizing) event. However, when a fault event occurs, label automaton Ω moves to state F and remains in that state as long as the system executes a faulty behavior (i.e., no reset event occurs). As we deal with intermittent faults, the system switches between these two status indefinitely.

In order to keep tracking of the occurrence of faults and their corresponding resets along the system's evolution, we compute automaton G'_ℓ as the parallel composition of automata G and Ω_N ($G'_\ell = G \parallel \Omega_N$). G'_ℓ is depicted in Figure 5.5.



(a) Automaton G (c) Automaton G'_ℓ

Figure 5.5 – automaton G'_ℓ in normalizing modeling

From monitoring point of view, the states of automaton G_ℓ can be partitioned into two subsets: 'Normal' and 'Faulty', which can be captured using *fault-assignment function*:

$$\Psi : X \to \{N, F\}.$$

Now, let us define *labeling function* ℓ, $\ell : L(G) \subseteq \Sigma^* \to \{N, F\}$ as follows:

Let $s \in L$ be an event-sequence, then:

- $\ell(s) = N \quad if: \ (\Sigma_f \notin s) \vee \exists\, s', s'' : (s = s's'') \wedge (\Sigma_f \in s') \wedge [s' \in \psi(\Sigma_r)] \wedge (\Sigma_f \notin s'')$
- $\ell(s) = F \quad if: \ \exists\, s', s'' : (s = s's'') \wedge [s' \in \psi(\Sigma_f)] \wedge \ \ (\Sigma_r \notin s'')$

It should be underlined that in the remainder of our thesis the recover formalism is adopted. However, a discussion regarding the normalizing formalism will be given as necessary.

5.5 Notions of Diagnosability

5.5.1 Assumptions

Besides the well-known assumptions considered in the diagnosis of permanent faults [Sampath et al., 1995], that is language $L(G)$ is live and no cycle of unobservable events exists in G, the following assumptions are considered:

(A1) Each class of fault events Σ_f has its corresponding class of reset event Σ_r. Recall that both faulty and reset events are unobservable.

(A2) At least one observable event exists between the occurrence of a fault event $\sigma_f \in \Sigma_f$ and its corresponding reset event $\sigma_r \in \Sigma_r$ and between the occurrence of a reset event $\sigma_r \in \Sigma_r$ and a new occurrence of fault event $\sigma_f \in \Sigma_f$.

(A3) Each occurrence of fault event $\sigma_f \in \Sigma_f$ is followed by the occurrence of a reset event $\sigma_r \in \Sigma_r$ within a finite delay. Similarly, each occurrence of a reset event $\sigma_r \in \Sigma_r$ is followed by a new occurrence of a fault event $\sigma_f \in \Sigma_f$ within a finite delay. This assumption implies that fault and reset events occur with some regularity (pseudo-periodicity). These notions are called the $\Sigma_f - recurrence$ and $\Sigma_r - recurrence$, as introduced in [Contant, 2005].

5.5.2 Definitions of Diagnosability

Intermittent faults are dynamic [Contant, 2005]; that is, they can repeatedly occur and reset. Thus, the fault status evolves along the system evolution. Consequently, several notions of diagnosability can be introduced, according to the properties and the specifications one may need to investigate. For example, one may want to ensure the detection of any fault occurrence or its corresponding recovery. Another definition would require checking the presence of each fault before its recovery or checking the recovery of a fault before a new occurrence of this fault. Determining accurately the finite delays in which the fault or its recovery can be diagnosed can also be of interest in practice. Obviously,

the choice between these considerations greatly depends on the application nature of the system and the objectives assigned to the diagnosis activity.

In this section, we discuss and adapt various definitions of diagnosability to our modeling formalism. The first two notions of diagnosability deal with the detection of fault occurrences and that of their recovery [Contant, 2005] without necessarily identifying at any moment whether or not the current status of the system (fault is present or not) is precisely known. We call these definitions the *weak* diagnosability properties. Then, we discuss some restrictive versions of these definitions, for the purpose of assessing the ability to identify with certainty the status of the system after the occurrence of an intermittent fault or its recovery. These last two properties are called *strong* diagnosability. These properties are firstly introduced in [Contant, 2005] with slight differences.

Définition 18 *(WF-diagnosability)*

An FSA G is said to be WF-diagnosable w.r.t. projection P, fault class Σ_f and its corresponding reset event class Σ_r, if the following holds:

$$(\exists\, n \in \mathbb{N}) \quad [\forall s \in \psi(\Sigma_f)] \quad (\forall t \in L/s) \quad [\| t \| \geq n \Rightarrow D_{WF}]$$

where diagnosability condition D_{WF} is:

$$\omega \in [P_L^{-1}(P(s.t))] \Rightarrow (\Sigma_f \in \omega)$$

□

WF-diagnosability, where 'W' stands for 'weak' and 'F' for 'fault occurrence', has the following meaning: for any event-trace s ending with a fault event in Σ_f, and t any continuation of s, then, $n \in \mathbb{N}$ exists such that, after the occurrence of at most n events, it is possible to detect that a fault has occurred based on the captured observation. This implies that all the event-traces that are indistinguishable from $s.t$ contain at least one fault from Σ_f. It is worth noticing here that there is no constraint regarding the system status at the time when the fault occurrence can be detected. Phrased differently, according to the above definition, it is possible that the system has left the faulty status when the fault occurrence is diagnosed.

As we deal with intermittent faults, each fault occurrence is followed later by its corresponding reset event (recover). Therefore, it is also interesting to discuss the diagnosability of the recovery occurrence. Namely, this consists in checking whether we can detect within a finite delay that the system has moved to its recovered behavior after the fault has been recovered. In what follows, we introduce WR-diagnosability which represents the dual notion of WF-diagnosability.

Définition 19 *(WR-diagnosability)*

An FSA G is said to be WR-diagnosable w.r.t. projection P, fault class Σ_f and its corresponding reset event class Σ_r, if the following holds:

$$(\exists\, n \in \mathbb{N}) \quad [\forall s \in \psi(\Sigma_r)] \quad (\forall t \in L/s) \quad [\| t \| \geq n \Rightarrow D_{WR}]$$

where diagnosability condition D_{WR} is:

$$\omega \in [P_L^{-1}(P(s.t))] \Rightarrow (\Sigma_r \in \omega)$$

□

Where 'R' stands for reset event occurrences, WR-diagnosability has the following meaning: there exists $n \in \mathbb{N}$, such that for any event-trace s ending with a reset event in Σ_r (which means that at least one fault has occurred and recovered) and t any continuation of s which is arbitrarily long, then, after at most n events it is possible to detect the fault recovery based on the captured observation. This implies that all the event-traces that are indistinguishable from $s.t$ have experienced a fault occurrence and its recovery. In the same way as for WF-diagnosability, there is no constraint regarding the system status when the recovery is diagnosed.

Example 8 *Let us take automaton G in Example 7 (Figure 5.3). G shows one type of faults (event f) with its corresponding reset class (event r). Consider execution $\rho = 1, f, 2, a, 3, r, 4, b\ \ (5, f, 6, c, 7,\ r, 8, d)^*$. Let the infinite event-trace, corresponding to this execution, be noted $s.t$ with, $s = (f, a, r, b, f)$ (one can see that $s \in \psi(\Sigma_f)$), and $t = (c, r, d, f)^*$. The resulting observed event-trace is then $P(s.t) = (a, b, (c, d)^*)$. The only event-trace in G which shares the same observable event-trace with ρ is $\omega = (f, a, b, (r, c, f, d)^*)$. One can see that once 3 events after executing the faulty event-trace s (2 observable events) have occurred, it becomes possible to infer accurately the occurrence of fault f (since f occurs in all the event-traces which share the same observation with $s.t$). Thus, according to Definition 18, G is WF-diagnosable ($n \geq 3$). The same reasoning can be done to show that G is also WR-diagnosable.*

As mentioned earlier in the chapter, the above definitions serve only to detect the occurrence of the fault (or its recovery) without any guarantee regarding the determination of the system status at any moment. Throughout this thesis, the above definitions are referred to as the weak diagnosability.

In what follows, we introduce strong versions of the above notions, in order to consider the identification of the system status.

Définition 20 *(SF-diagnosability)*

An FSA G is said to be SF-diagnosable w.r.t. projection P, fault class Σ_f and corresponding reset event class Σ_r, if the following holds:

$$(\exists\, n \in \mathbb{N}) \quad [\forall s \in \psi(\Sigma_f)] \quad (\forall t \in L/s) \quad [\| t \| \geq n \Rightarrow D_{SF}]$$

where diagnosability condition D_{SF} is:

$$\exists\, t' \leq t : \forall\, \omega \in [P_L^{-1}(P(s.t'))] \Rightarrow \ell(\omega) = F$$

□

In SF-diagnosability, 'S' stands for '*strong*' and 'F' for '*fault occurrence*'.
SF-diagnosability states that for any event-trace s ending with a fault event in Σ_f, and t any continuation of s, it is possible to detect the occurrence of the fault and to determine, with certainty, the faulty status of the system upon the occurrence of at most n events following the fault event. This implies that all the event-traces that are indistinguishable from $s.t$ lead the system to faulty states at the same observation point, within a finite delay after the occurrence of the fault.

Hereafter, SR-diagnosability, which represents the dual notion of SF-diagnosability, is stated.

Définition 21 *(SR-diagnosability)*

An FSA G is said to be SR-diagnosable w.r.t. projection P, fault class Σ_f and corresponding reset event class Σ_r, if the following holds:

$$(\exists\, n \in \mathbb{N}) \quad [\forall s \in \psi(\Sigma_r)] \quad (\forall t \in L/s) \quad [\| t \| \geq n \Rightarrow D_{SR}]$$

where diagnosability condition D_{SR} is:

$$\exists\, t' \leq t : \forall\, \omega \in [P_L^{-1}(P(s.t'))] \Rightarrow \ell(\omega) = R$$

□

It means that for any event-sequence s ending with a reset event in Σ_r, and t any continuation of s, it is possible, based on the captured observations, to detect the reset of the fault and to determine, with certainty, the recovered status of the system. This implies that all the event-sequences that are indistinguishable from $s.t$ lead to recovered states from reset fault class F at the same observable point, within a finite delay after the fault recovery.

Example 9 *Let us take again automaton G (cf. Figure 5.3), execution $\rho = 1, f, 2, a, 3, r, 4, b(5, f, 6, c, 7, r, 8, d)^*$. There exists, in automaton G, one event-sequence $\omega = f, a, b, (r, c, f, d)^*$ which shares the same observed event-sequence with ρ, i.e., $P(\omega) = P(\rho) = a, b, (c, d)^*$. However, according to Definition 20, there is no bound n such that, after this limit, both event-sequences ρ and ω lead to faulty states at the same time, which means that it is not possible to identify accurately the faulty status of the system. Therefore, G is non-SF-diagnosable. The same reasoning can be applied to show that G is non-SR-diagnosable.*

It should be noticed that (SF, SR)-diagnosability are stronger notions compared to (WF, WR)-diagnosability. Indeed, SF-(resp. SR)-diagnosability requires the detection of any fault (resp. any recovery) and the certain determination of the system status (faulty or recovered). However, WF-(resp. WR)-diagnosability only requires the detection of the fault (resp. its recovery) within a finite delay. Therefore, it is straightforward to infer the following,

Proposition 5 [Contant et al., 2004]

- *SF-diagnosability $\Rightarrow$ WF-diagnosability*
- *SR-diagnosability $\Rightarrow$ WR-diagnosability*

5.5.2.1 Relationship Between Properties

Earlier in this chapter, it was assumed that the models under investigation satisfy the $\Sigma_f - recurrence$ and $\Sigma_r - recurrence$ (See assumption (A3)). That is, the fault and reset events occur with some regularity along any possible event sequence in the system behavior. Such an assumption ensures the following properties [Contant et al., 2004]:

Property 3 *Let $x_1, x_2, \ldots, x_n$ be a state-trace associated with event-sequence $s = \sigma_1, \sigma_2, \ldots, \sigma_n$ and that forms a cycle in G, i.e., $x_{(i+1)_{mod_n}} = \delta(x_i, \sigma_i)$ for $1 \leq i \leq n$. Then, fault event $\sigma_f \in s$ if and only if its corresponding reset event $\sigma_r \in s$.*

This property means that each cycle that contains an event fault σ_f, contains also its corresponding reset event σ_r.

Corollary 3 *Let us consider a system model G that satisfies assumptions (A1), (A2) and (A3). Then, G is WF-diagnosable if and only if it is WR-diagnosable.* □

Proof. A formal proof of this corollary will be given in Chapter 8 (see Section 8.2.2).

As WF-diagnosability is equivalent to WR-diagnosability (under assumptions (A1), (A2), and (A3)), the necessary and sufficient conditions will be discussed only regarding WF-diagnosability in the sequel.

Remark 3 *It is worth noticing that, unlike the weak diagnosability, there is no equivalence between the strong version of diagnosability. Therefore, a system model may be SF-diagnosable while it is non-SR-diagnosable, or conversely.*

The following Example illustrates this remark,

Example 10 *Consider automaton G_1, shown in Figure 5.6. The sets of observable and unobservable events are $\Sigma_o = \{a, b, c, d, e\}$ and $\Sigma_u = \{f, r\}$, respectively. In addition, $\Sigma_f = \{f\}$ and $\Sigma_r = \{r\}$. Executions $\pi_1 = 1, a, 2, f, 3, b, (4, c, 5, r, 6, d, 7, f, 8, e)^*$ and $\pi_2 = 1, f, 11, a, 12, r, 13, b, (14, f, 15, c, 16, d, 17, r, 18, e)^*$ share the same observation $\omega = ab(cde)^*$. one can observe that both executions reach faulty states (respectively, state 5 and state 16) after executing observable sequence 'abc', and then periodically upon any observable sequence $abc(dec)^*$. Therefore, model G_1 is SF-diagnosable. However, π_1 and π_2 never reach recovered states at the same time upon the execution of a same observable sequence. Therefore, model G_1 is non-SR-diagnosable.*

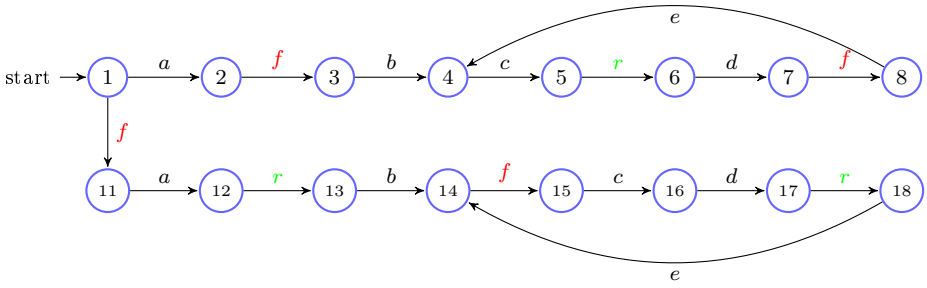


Figure 5.6 – A system model which is SF-diagnosable and non-SR-diagnosable

5.6 Construction of the Diagnoser

In this section, we extend the diagnoser-based approach proposed in the previous chapter, in order to deal with intermittent faults. In what follow, we discuss the diagnoser construction, while considering the recover formalism for modeling intermittent faults

5.6.1 The Structure of the Diagnoser Node

In our proposed diagnoser, the nodes are equivalent to the states in the classic diagnosers [Sampath et al., 1995, Contant et al., 2004, Zad et al., 2003, Cabasino et al., 2009a], except that an explicit distinction is made, within each node, between the normal

states (denoted by set $\mathcal{X}_N$), the faulty states (denoted by set $\mathcal{X}_F$), and the recovered states (denoted by set $\mathcal{X}_R$). Moreover, we indicate if there exists some (faulty) transitions from $\mathcal{X}_N$ or $\mathcal{X}_R$ to $\mathcal{X}_F$ and similarly, if there exists some reset transitions from $\mathcal{X}_F$ to $\mathcal{X}_R$.

Figure 5.7 depicts the general form of a diagnoser node. In particular, three subsets of states can be discussed; $\mathcal{X}_N$ is the set of normal states and $\mathcal{X}_F$ the set of faulty ones, while $\mathcal{X}_R$ is the set of recovered states. Some faulty states may be reached from normal (resp. recovered states) in the same node through the occurrence (resp. reoccurrence) of faulty events. This is depicted by a faulty transition from $\mathcal{X}_N$ or $\mathcal{X}_R$ to $\mathcal{X}_F$ within the diagnoser node. Similarly, some recovered states may be reached from faulty states in the same node through the occurrence of reset events.

In Figure 5.7, the dashed arrows entering in and outgoing from each node show the different possibilities an entering/outgoing observable transition may correspond to. For instance, observable event σ_2 output by diagnoser node a may be output from the normal, the faulty, or the recovered set. That is, σ_2 is generated from at least one of these three sets. Note that these transition (entering/outgoing dashed arrows) are depicted here for the sake of explanation but are not actually encoded in the diagnoser. Moreover, some faulty transitions f may or may not exist between $\mathcal{X}_N$ (or $\mathcal{X}_R$) and $\mathcal{X}_F$. Similarly, some reset transitions r may or may not exist between $\mathcal{X}_F$ and $\mathcal{X}_R$. This is depicted by the dotted arrows (inside the diagnoser node) linking $\mathcal{X}_N$, $\mathcal{X}_F$ and $\mathcal{X}_R$ within the same node. It is worth noticing that the existing of such transitions between the same node subsets is actually encoded in the diagnoser structure using boolean variables that are set to *True* when such transitions exist, and to *False* if not.

These information will be useful while investigating diagnosability. In order to simplify the notation, we use $a.\mathcal{X}_N$ (resp. $a.\mathcal{X}_F$, $a.\mathcal{X}_R$) to indicate the set of normal states $\mathcal{X}_N$ (resp. the faulty states $\mathcal{X}_F$ and the recovered states $a.\mathcal{X}_R$) corresponding to diagnoser node a.

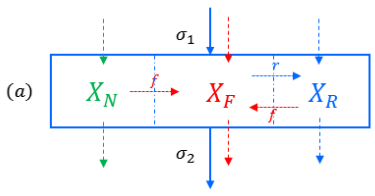


Figure 5.7 – The diagnoser node structure

5.6.2 The Diagnoser Construction

For a given FSA G, the diagnoser can be defined as follows.

Définition 22 *(Diagnoser)*
Let $G = \langle X, \Sigma, \delta, x_0 \rangle$ be an FSA to be diagnosed. The diagnoser associated with G is a deterministic FSA $\mathcal{D} = \langle \Gamma, \Sigma_o, \delta_{\mathcal{D}}, \Gamma_0 \rangle$, where:

1. *Γ is a finite set of diagnoser nodes;*
2. *Γ_0 is the initial diagnoser node with:*
   a) $\Gamma_0.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(x_0)$;
   b) $\Gamma_0.\mathcal{X}_F = Reach_{\Sigma_u \setminus \Sigma_r}(Img(\Gamma_0.\mathcal{X}_N, \Sigma_f))$.
   b) $\Gamma_0.\mathcal{X}_R = Reach_{\Sigma_u \setminus \Sigma_f}(Img(\Gamma_0.\mathcal{X}_F, \Sigma_r))$.
3. *$\delta_{\mathcal{D}} : \Gamma \times \Sigma_o \to \Gamma$ is the transition relation, defined as follows:*

$\forall a, a' \in \Gamma$, $\sigma \in \Sigma_o$: $a' = \delta_{\mathcal{D}}(a, \sigma)$

$$\Leftrightarrow \begin{cases} a'.\mathcal{X}_N = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a.\mathcal{X}_N, \sigma)) \\ a'.\mathcal{X}_F = Reach_{\Sigma_u \setminus \Sigma_r}(Img(a.\mathcal{X}_F, \sigma) \cup Img(a'.\mathcal{X}_N, \Sigma_f) \cup Img(a'.\mathcal{X}_R, \Sigma_f)) \\ a'.\mathcal{X}_R = Reach_{\Sigma_u \setminus \Sigma_f}(Img(a.\mathcal{X}_R, \sigma) \cup Img(a'.\mathcal{X}_F, \Sigma_r)) \end{cases} \quad (5.1)$$

□

It should be noticed that one may think that a conflict arises when constructing $a'.\mathcal{X}_F$ and $a'.\mathcal{X}_R$. However, this is only a fictitious one. More precisely $a'.\mathcal{X}_F$ and $a'.\mathcal{X}_R$ can be actually constructed as follows:

Let $\mathcal{Q}_F = Img(a.\mathcal{X}_F, \sigma) \cup Img(a'.\mathcal{X}_N, \Sigma_f)$ and $\mathcal{Q}_R = Img(a.\mathcal{X}_R, \sigma)$. Hence, according to assumption (A2):

$$\begin{cases} a'.\mathcal{X}_F = Reach_{\Sigma_u \setminus \Sigma_r}(\mathcal{Q}_F \cup Img(Reach_{\Sigma_u \setminus \Sigma_f}(\mathcal{Q}_R), \Sigma_f)) \\ a'.\mathcal{X}_R = Reach_{\Sigma_u \setminus \Sigma_f}(\mathcal{Q}_F \cup Img(Reach_{\Sigma_u \setminus \Sigma_r}(\mathcal{Q}_F), \Sigma_r)) \end{cases} \quad (5.2)$$

Property 4 *(Equivalence between nodes)*
Two diagnoser nodes a and a' in $\mathcal{D}$ are equivalent if $a.\mathcal{X}_N = a'.\mathcal{X}_N, a.\mathcal{X}_F = a'.\mathcal{X}_F and a.\mathcal{X}_R = a'.\mathcal{X}_R$

5.6.3 The Various Types of Nodes in the Diagnoser

According to the diagnoser construction, one can differentiate between various types of nodes int the diagnoser variant, in the same way as in the classic diagnoser:

- **N-node**: is a diagnoser node of which the sets of faulty and recovered states are empty ($\mathcal{X}_F = \mathcal{X}_R = \emptyset$);
- **F-node**: is a diagnoser node of which the sets of normal and recovered states are empty ($\mathcal{X}_N = \mathcal{X}_R = \emptyset$);
- **R-node**: is a diagnoser node of which the sets of normal and faulty states are empty ($\mathcal{X}_N = \mathcal{X}_F = \emptyset$);
- **NF-node**: is a diagnoser node of which the sets of recovered states is empty, while $\mathcal{X}_N$ and $\mathcal{X}_F$ are not;
- **NR-node**: is a diagnoser node of which the sets of faulty states is empty ($\mathcal{X}_F = \emptyset$), while $\mathcal{X}_N$ and $\mathcal{X}_R$ are not;
- **FR-node**: is a diagnoser node of which the sets of normal states is empty ($\mathcal{X}_N = \emptyset$), while $\mathcal{X}_F$ and $\mathcal{X}_R$ are not;
- **NFR-node**: is a diagnoser node where $\mathcal{X}_N \neq \emptyset$, $\mathcal{X}_F \neq \emptyset$ and $\mathcal{X}_R \neq \emptyset$.

Besides the above mentioned types, we introduce the following types that will be used to state the necessary and sufficient condition for diagnosability in the sequel.

- **N-uncertain** node, if it is an NF-(or NFR- or NR-)node;
- **F-uncertain** node, if it is an NF-(or NFR- or FR-)node;
- **R-uncertain** node, if it is an NR-(or NFR- or FR-)node;
- **non-N-certain** node, if it is not N-node;
- **non-F-certain** node, if it is not F-node;
- **non-R-certain** node, if it is not R-node;

It is worth noticing that a non-N-certain (resp. non-F-certain, non-R-certain) node is not necessarily an N-uncertain (resp. F-uncertain, R-uncertain).

5.6.3.1 Intermittent Fault Propagation in the Diagnoser

The diagnoser construction preserves a specific fault propagation scheme regarding the fact that faults are intermittent and based on the assumptions (A1), (A2) and (A3). These propagation rules are depicted in Figure 5.8, and outlined below.

- From an N-node, either an N-node or an NF-node can be reached;
- From an F-node, either an F-node or an FR-node can be reached;
- From an R-node, an R-node or an FR-node can be reached;
- From an NF-node, an NF-node, an N-node, an F-node, an FR-node or NFR-node can be reached;
- From an NR-node, an NR-node, an N-node, an R-node, an FR-node or NFR-node can be reached;
- From an FR-node, an FR-node, an F-node, an FR-node or NFR-node can be reached;
- From NFR-node, all the node types can be reached.

It should be noticed that the dashed self loop on a node type (e.g. NF-node) means that the system cannot remains indefinitely in the same node type (by (A2) and (A3)). on the contrary, the system may remain in the same type of nodes where self loops are depicted in continued lines (e.g. NFR-node).

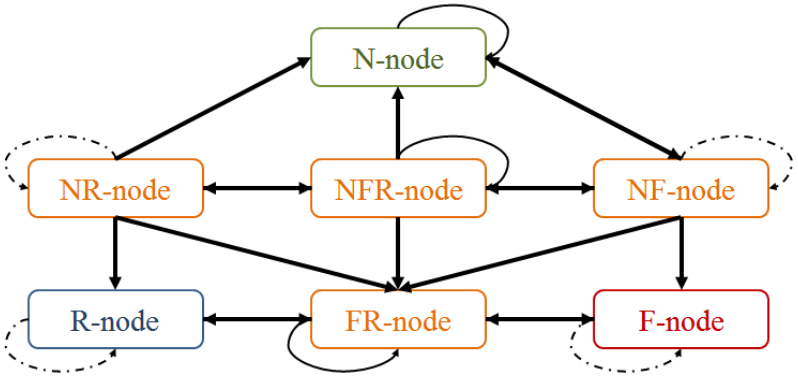


Figure 5.8 – Intermittent fault propagation in the diagnoser

Example 2 *The diagnoser $\mathcal{D}$ corresponding to FSA G (c.f., Figure 7) is shown in Figure 5.9. The initial node (a_0) is composed of initial state 1 and state 2 reachable from state*

1 by the occurrence of faulty event f. Thus, $a_0.\mathcal{X}_N = \{1\}$, $a_0.\mathcal{X}_F = \{2\}$ and $a_0.\mathcal{X}_R = \emptyset$, which means that node a_0 is an NF-node. One can observe that event a is only enabled from the faulty set of node (a_0). Therefore, node a_1 reached from a_0 through event a contains an empty set of normal states, i.e. $a_1.\mathcal{X}_N = \emptyset$. Moreover, $a_1.\mathcal{X}_F = \{3\}$ and, since state 4 is reachable from state 3 through the occurrence of a reset event r and no unobservable event is enabled from state 4, then $a_1.\mathcal{X}_R = \{4\}$. Hence, node a_1 is an FR-node. The rest of nodes are constructed according to the fault propagation scheme in Figure 5.8. Finally, one can observe that two cycles exist in $\mathcal{D}$: (1) a cycle composed of FR-nodes a_2 and a_3 and (2) a cycle composed of N-node a_5.

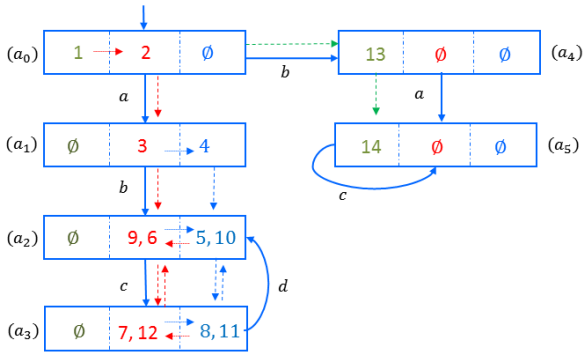


Figure 5.9 – The diagnoser of system model G (of Example 7)

5.7 Analysis of WF-Diagnosability

This section is dedicated to the analysis of WF-diagnosability (and WR-diagnosability according to Corollary 3) by adapting the necessary and sufficient conditions, introduced by Contant et al. [Contant et al., 2004], to our intermittent fault modeling (i.e., the recover modeling formalism). Then, we establish a systematic method for checking such a condition on the basis of our diagnoser variant. In fact, the developed method is an extension of the systematic procedure developed in Chapter 4 for the analysis of diagnosability of permanent faults.

5.7.1 Necessary and Sufficient Condition for WF-diagnosability

In order to state the necessary and sufficient condition for WF-diagnosability, we first introduce the notions of WF-uncertain and WF-indeterminate cycle. In fact, the notion

of WF-indeterminate cycle is introduced in [Contant et al., 2004] as F_i^O-indeterminate cycle (with a slight difference regarding the modeling of intermittent faults).

Définition 23 *(WF-uncertain cycle)*

A cycle $c\ell = a_1, a_2, \ldots, a_n$ in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$, $\sigma_i \in \Sigma_o$ for $1 \leq i \leq n$, is said to be WN-uncertain cycle if all nodes in $c\ell$ are N-uncertain ones.

Définition 24 *(WF-indeterminate cycle)*

Let $c\ell = a_1, a_2, \ldots, a_n$ be a WF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$, $\sigma_i \in \Sigma_o$ for $1 \leq i \leq n$.

$c\ell$ is said to be a WF-indeterminate cycle if $\exists m, m' \in \mathbb{N}^*$ such that the following condition holds:

(C1) Considering nodes $a_1, a_2, \ldots, a_n \in \mathcal{D}$, $\exists\, x_i^k, y_i^{k'} \in a_i, i = 1, \ldots, n$, and $k = 1, \ldots, m$ and $k' = 1, \ldots, m'$, with $x_i^k \in a_i.\mathcal{X}_F$ for some i and k and:

$\delta(x_i^k, s_i^k\sigma_i) = x_{(i+1)}^k$, *for* $i = 1, \ldots, n-1$, $k = 1, \ldots, m$

$\delta(y_i^{k'}, t_i^{k'}\sigma_i) = y_{(i+1)}^{k'}$, *for* $i = 1, \ldots, n-1$, $k' = 1, \ldots, m'$

$\delta(x_n^k, s_n^m\sigma_n) = x_1^{(k+1)}$, *for* $k = 1, \ldots, m-1$

$\delta(y_n^{k'}, t_n^{m'}\sigma_n) = y_1^{(k'+1)}$, *for* $k' = 1, \ldots, m'-1$

$\delta(x_n^m, s_n^m\sigma_n) = x_1^1$

$\delta(y_n^{m'}, t_n^{m'}\sigma_n) = y_1^1$

with $s_i^k \in \Sigma_u^*\ \forall 1 \leq i \leq n$ and $1 \leq k \leq m$ and $\exists 1 \leq i_1 \leq n, \exists k_1 \leq m$ s.t. $\Sigma_f \in s_{i_1}^{k_1}$. $t_i^{k'} \in (\Sigma_u \backslash \Sigma_f)^*\ \forall 1 \leq i \leq n$ and $1leqk' \leq m'$

In other words, condition (C1) indicates that two cycles $c\ell_G^1$ and $c\ell_G^2$ exist in G such that:

- the event-sequence associated with $c\ell_G^1$ contains (at least) one fault event from Σ_f and generates an observable sequence $(\sigma_1, \sigma_2, \ldots, \sigma_n)^m$ with $m \in \mathbb{N}^*$;
- the event-sequence associated with $c\ell_G^2$ is fault-free and generates an observable sequence $(\sigma_1, \sigma_2, \ldots, \sigma_n)^{m'}$ with $m' \in \mathbb{N}^*$.

Here, m (resp. m') denote the number of times cycle $c\ell$ in $\mathcal{D}$ is completed before cycle $c\ell_G^1$ (resp. $c\ell_G^2$) in G is completed. That is, nm (resp. nm') is the length of $c\ell_G^1$ (resp. $c\ell_G^2$) in terms of number of states, i.e. $|c\ell_G^1| = nm$ (resp. $|c\ell_G^2| = nm'$) (the readers can refers to [Sampath et al., 1995] for more details regarding this point).

Hereafter, we recall the necessary and sufficient condition for a system model G to be WF-diagnosable as stated in [Contant et al., 2004].

Theorem 5 *(Necessary & sufficient condition for WF-diagnosability [Contant et al., 2004])*
A system model G is WF-diagnosable w.r.t projection P, class of fault events Σ_f and its corresponding class of reset events Σ_r, if and only if no WF-indeterminate cycle exists in its corresponding diagnoser $\mathcal{D}$. □

5.7.2 Verification of WF-Diagnosability

In what follows, we first reformulate the necessary and sufficient condition for WF-diagnosability on the basis of the new diagnoser variant. Then, we propose a systematic method for the actual verification, without needing to construct any intermediate model.

Proposition 6 *Let $c\ell = a_1, a_2, \ldots, a_n$ be a WF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$. Then, there exists, at least, one fault-free cycle in G, which shares the same observation $(\sigma_1, \sigma_2, \ldots, \sigma_n)^*$.* □

Proof. The proof of Proposition 6 is omitted as it is similar to that of Proposition 1 in Chapter 4.

This result is interesting for checking *WF-indeterminate* cycles. It is, in fact, sufficient to check that a cycle $c\ell_G$ exists in model G such that it shares the same observation with *WF-uncertain* cycle $c\ell$ in the diagnoser, and has experienced at least one fault occurrence. In order terms, one does not need to check the existence of a corresponding faulty-free cycle.

In order to check the existence of *WF-indeterminate* cycles in $\mathcal{D}$, we introduce the notion of $c\ell_{WF}$-indicating sequence associated to WF-uncertain cycle $c\ell$.

Définition 25 *($c\ell_{WF}$-indicating sequence)*
Let $c\ell = a_1, a_2, \ldots, a_n$ be a WF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ $\forall 1 \leq i \leq n$. Also, let us chose the first node a_1 in $c\ell$ in such a way as to fulfill: $a_i.\mathcal{X}_F \cup a_i.\mathcal{X}_R \neq \emptyset$. Since $c\ell$ is a WF-uncertain cycle, such a node necessarily exists in $c\ell$.

$c\ell_{WF}$-indicating sequence $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots,$ associated with $c\ell$ is an infinite sequence of sets of states, such that:

- $\mathcal{S}_1 = a_1.\mathcal{X}_F \cup a_1.\mathcal{X}_R$;
- $\mathcal{S}_j = Reach_{\Sigma_u}(Img(\mathcal{S}_{j-1}, \sigma_{(j-1)_{mod_n}}))$ *for* $j > 1$.

□

In fact, the $c\ell_{WF}$-indicating sequence tracks the subsets of faulty and recovered states in each node of $c\ell$ without considering the faulty states generated through the occurrence of some faulty transitions outgoing from the normal set of states in the traversed nodes

(except for $\mathcal{S}_1$ which holds all the faulty and recovered states of a_1). Actually, the $c\ell_{WF}$-indicating sequence is introduced with the aim of tracking the existence of cycles in G that share the same observation with some corresponding WF-uncertain cycle $c\ell$, and which have experienced at least one fault occurrence.

Définition 26 *(Series $\mathbb{S}^{c\ell}$)*

Let $c\ell = a_1, a_2, \ldots, a_n$ be a WF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$, for $1 \leq i \leq n$. $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ is the $c\ell_{WF}$-indicating sequence. Series $\mathbb{S}^{c\ell} = \mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \ldots$ corresponding to $\rho^{c\ell}$, is defined as follows:

$$\begin{cases} \mathbb{S}^{c\ell} : \mathbb{N}^* \rightarrow 2^X \\ \mathbb{S}_i = \mathcal{S}_{(1+(i-1)n)}, \forall i \in \mathbb{N}^*. \end{cases} \tag{5.3}$$

□

It is worth noticing that $\mathbb{S}^{c\ell}$ can also be written as follows: $\mathbb{S}^{c\ell} = \mathcal{S}_1, \mathcal{S}_{(1+n)}, \ldots, \mathcal{S}_{(1+kn)}, \mathcal{S}_{(1+(k+1)n)}, \ldots$, for $k \in \mathbb{N}$. In other terms, series $\mathbb{S}^{c\ell}$ is extracted from $c\ell_{WF}$-indicating sequence $\rho^{c\ell}$ by considering sample elements with n steps (n is the number of nodes in the WF-uncertain cycle).

Proposition 7 *Let $c\ell = a_1, a_2, \ldots, a_n$ be a WF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ ($\sigma_i \in \Sigma_o$), for $1 \leq i \leq n$. Let $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ be the $c\ell_{WF}$-indicating sequence and $\mathbb{S}^{c\ell} = \mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \ldots$ be the corresponding series as defined above. Then, the following property holds true:*

$$\forall k \in \mathbb{N}^* \ : \ \mathbb{S}_{k+1} \subseteq \mathbb{S}_k$$

i.e., $\forall k \in \mathbb{N}^* \ : \ \mathcal{S}_{1+nk} \subseteq \mathcal{S}_{1+n(k-1)}$ □

Proof. The proof of Proposition 7 is omitted as it is similar to that of Proposition 3 in Chapter 4.

The above-mentioned property means that, by ignoring the faulty states generated by the faulty transitions from the normal sets of states into the faulty ones within the same node, one can ensure the (non-strict) inclusion relationship between terms $\mathcal{S}_{1+ni} \forall i \in \mathbb{N}$.

Proposition 8 *Let $c\ell = a_1, a_2, \ldots, a_n$ be a WF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ ($\sigma_i \in \Sigma_o$), for $1 \leq i \leq n$. Let $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ be the $c\ell_{WF}$-indicating sequence and $\mathbb{S}^{c\ell} = \mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \ldots$ be its corresponding series. Then, $\mathbb{S}^{c\ell}$ reaches a fixed-point.*
i.e., $\exists\, k \in \mathbb{N}^*$ *s.t.* $\forall i \in \mathbb{N} : \mathbb{S}_{k+i} = \mathbb{S}_k$, *which means that* $\mathcal{S}_{(1+(k+i-1)n)} = \mathcal{S}_{(1+(k-1)n)}$. □

Proof. The proof of Proposition 8 is omitted as it is similar to that of Proposition 4 in Chapter 4.

Proposition 8, establishes the fact that there exists an index i from which $c\ell_{WF}$-indicating sequence $\rho^{c\ell}$ shows a repetitive bloc of length n $[\mathcal{S}_{1+kn}, \mathcal{S}_{2+kn}, \ldots, \mathcal{S}_{n-1+kn}\mathcal{S}_{(k+1)n}]$ with $\mathcal{S}_{(1+(k+1)n} = \mathcal{S}_{1+kn}$. Phrased differently, if we take $i = k+1$ then $\rho^{c\ell}$ necessarily takes one of the following two forms:

1. *A prime sequence:* a non-cyclic elementary sequence (possibly empty) $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{i-1}$, connected to an elementary cycle $(\mathcal{S}_{(i+1)}, \mathcal{S}_{(i+2)}, \ldots, \mathcal{S}_{(i-1+n)}, \mathcal{S}_{(i+n)})^*$, with $\mathcal{S}_{(i+1+n)} = \mathcal{S}_i$;
2. *A finite sequence of non-empty elements $\mathcal{S}_1$, $\mathcal{S}_2, \ldots, \mathcal{S}_j$ (for $j \in \mathbb{N}^*$) followed by an infinite number of empty elements, i.e., $\mathcal{S}_{(j+k)} = \emptyset$, $\forall k \in \mathbb{N}^*$;*

In fact, the first case, i.e., a prime sequence, reveals the presence of at least one actual cycle in the system model, which has experienced at least one fault occurrence and corresponds to the WF-uncertain cycle $c\ell$. This is depicted in the $c\ell_{WF}$-indicating sequence by the presence of the elementary cycle. In contrast, in the second case, the WF-uncertain cycle in the diagnoser does not correspond to an actual *faulty* cycle in the system model. In what follows, these features will be used to determine whether an WF-uncertain cycle is an WF-indeterminate one or not.

Theorem 6 *For an WF-uncertain cycle $c\ell = a_1, a_2, \ldots, a_n$ in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ $\forall 1 \leq i \leq n$. $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ is the $c\ell_{WF}$-indicating sequence associated with $c\ell$ and $\mathbb{S}^{c\ell} = \mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \ldots$ is its corresponding series. Therefore, $c\ell$ is a WF-indeterminate cycle if and only if the fixed-point reached by series $\mathbb{S}^{c\ell}$ is non-empty.* □

In other terms, it means that $\forall i \in \mathbb{N}^* : \mathcal{S}_i \neq \emptyset$.

Actually, this theorem states that a WF-uncertain cycle is a WF-indeterminate one if and only if the $c\ell_{WF}$-indicating sequence does not reach an empty fixed-point. In other words, it takes the form of a prime sequence.

Proof.

The proof of Theorem 6 is omitted as it is similar to that of Theorem 4 in Chapter 4.

5.7.3 A Procedure for Checking WF-diagnosability

It is worth noticing that for the actual verification of WF-diagnosability, *a systematic procedure*, derived directly from Theorem 6, can be performed as follows:

When a WF-uncertain cycle $c\ell$ is encountered in diagnoser $\mathcal{D}$, one has to proceed as follows:

Generate the successive elements of $c\ell_{WF}$-indicating sequence $\rho^{c\ell}$ (starting from $\mathcal{S}_1$), and for each element $\mathcal{S}_i$ check the following conditions:

1. if $\mathcal{S}_i = \emptyset$: cycle $c\ell$ is not a WF-indeterminate cycle and therefore the procedure is stopped;

2. else: if $\exists k \in \mathbb{N} : i = 1 + kn$ (with $n = |c\ell|$), then:

   (a) if $\mathcal{S}_i = \mathcal{S}_{(i-n)} \neq \emptyset$, $c\ell$ is an WF-indeterminate cycle and the procedure is stopped;

   (b) otherwise continue.

This procedure is repeated on each WF-uncertain cycle generated in $\mathcal{D}$.

It is worth underlining that, according to the Corollary 3 (i.e., equivalence between WF-diagnosability and WR-diagnosability), the same procedure can be used for checking WR-diagnosability.

5.8 Analysis of SF-Diagnosability

In this section, we first reformulate the necessary and sufficient condition for SF-diagnosability on the basis of our diagnoser structure (by adapting the necessary and sufficient conditions, introduced by Contant et al. [Contant et al., 2004], to our diagnoser variant). Then, in the same manner as in the section above, we discuss the actual verification of SF-diagnosability, without needing to construct any intermediate model.

5.8.1 Necessary and Sufficient Condition for SF-diagnosability

Firstly, according to Proposition 5, it is easy to infer that the necessary and sufficient condition for WF-diagnosability represents a necessary condition for SF-diagnosability. That is, the presence of an WF-indeterminate cycle in diagnoser $\mathcal{D}$ implies the non-WF-diagnosability and, therefore the non-SF-diagnosability of the system model. However, it is only a necessary condition, which means that the absence of WF-indeterminate cycle does not imply the SF-diagnosability, as witnessed by Example 7.

In order to state the necessary and sufficient condition for SF-diagnosability, we first introduce the notions of SF-uncertain cycle and SF-indeterminate cycle (which is presented in [Contant et al., 2004] as F_i^P-indeterminate cycle).

Définition 27 *(SF-uncertain cycle)*

A cycle $c\ell = a_1, a_2, \ldots, a_n$ in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$, $\sigma_i \in \Sigma_o$, for $1 \leq i \leq n$, is said to be SF-uncertain if all nodes in $c\ell$ are non-F-certain and at least one node is an F-uncertain one. □

Définition 28 *(SF-indeterminate cycle)*

Let $c\ell = a_1, a_2, \ldots, a_n$ be an SF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$, $\sigma_i \in \Sigma_o$, for $1 \leq i \leq n$.

$c\ell$ is said to be an SF-indeterminate cycle if the following condition holds:

(C2) Considering nodes $a_1, a_2, \ldots, a_n \in \mathcal{D}$, $\exists\, x_i^k \in a_i, i = 1, \ldots, n$ and $k = 1, \ldots, m$, with $x_i^k \in a_i.\mathcal{X}_F$ for some i and k and:

$\delta(x_i^k, s_i^k\sigma_i) = x_{(i+1)}^k$, *for* $i = 1, \ldots, n-1$, $k = 1, \ldots, m$

$\delta(x_n^k, s_n^k\sigma_n) = x_1^{(k+1)}$, *for* $k = 1, \ldots, m-1$

$\delta(x_n^m, s_n^m\sigma_n) = x_1^1$.

with $s_i \in \Sigma_u^*$ and for some i, $\exists\sigma_f \in \Sigma_f$ s.t. $\sigma_f \in s_i$.

□

In other words, condition (C2) indicates that a cycle $c\ell_G$ exists in G such that the event-trace associated with it contains (at least) one fault event ($\sigma_f \in \Sigma_f$) and depicts the same observable sequence $(\sigma_1, \sigma_2, \ldots, \sigma_n)^m$ with $m \in \mathbb{N}$.

Hereafter, we recall the necessary and sufficient condition for a model G to be SF-diagnosable.

Theorem 7 *(Necessary & sufficient condition for SF-diagnosability)*

A system model G is SF-diagnosable, w.r.t projection P, class of fault events Σ_f and its corresponding class of reset events Σ_r, if and only if no SF-indeterminate cycle exists in its corresponding diagnoser $\mathcal{D}$.

5.8.2 Verification of SF-Diagnosability

In fact, checking the existence of an SF-indeterminate cycle can be performed in the same way as for WF-diagnosability, i.e., by generating an indicating sequence (which is associated with the SF-uncertain cycle this time) and then checking if all the element of the indicating sequence are non-empty. Hereafter, we recall the main results.

Définition 29 *($c\ell_{SF}$-indicating sequence)*

Let $c\ell = a_1, a_2, \ldots, a_n$ be an SF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ ($\sigma_i \in \Sigma_o$), for $1 \leq i \leq n$. Also, let us chose the first node a_1 in $c\ell$ in such a way as to fulfill: $a_i.\mathcal{X}_F \cup a_i.\mathcal{X}_R \neq \emptyset$.

$c\ell_{SF}$-indicating sequence $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$, associated with $c\ell$ is an infinite sequence of sets of states, such that:

- $\mathcal{S}_1 = a_1.\mathcal{X}_F \cup a_1.\mathcal{X}_R$;
- $\mathcal{S}_j = Reach_{\Sigma_u}(Img(\mathcal{S}_{j-1}, \sigma_{(j-1)_{mod_n}}))$, *for* $j > 1$.

□

Définition 30 *(Series $\mathbb{S}^{c\ell}$)*

Let $c\ell = a_1, a_2, \ldots, a_n$ be an SF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$, for $1 \leq i \leq n$. $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ is the $c\ell_{SF}$-indicating sequence. Series $\mathbb{S}^{c\ell} = \mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \ldots$ corresponding to $\rho^{c\ell}$, is defined as follows:

$$\begin{cases} \mathbb{S}^{c\ell} : \mathbb{N}^* \to 2^X \\ \mathbb{S}_i = \mathcal{S}_{(1+(i-1)n)}, \forall i \in \mathbb{N}^*. \end{cases} \tag{5.4}$$

□

Proposition 9 *Let $c\ell = a_1, a_2, \ldots, a_n$ be an SF-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ ($\sigma_i \in \Sigma_o$), for $1 \leq i \leq n$. Let $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ be the $c\ell_{SF}$-indicating sequence and $\mathbb{S}^{c\ell} = \mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \ldots$ be its corresponding series. Then, $\mathbb{S}^{c\ell}$ reaches a fixed-point. i.e., $\exists\, k \in \mathbb{N}^*$ s.t. $\forall i \in \mathbb{N} : \mathbb{S}_{k+i} = \mathbb{S}_k$, which means that $\mathcal{S}_{(1+(k+i-1)n)} = \mathcal{S}_{(1+(k-1)n)}$.*

□

Proof. The proof of Proposition 9 is omitted as it is similar to that of Proposition 4 in Chapter 4.

Theorem 8 *For an SF-uncertain cycle $c\ell = a_1, a_2, \ldots, a_n$ in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ ($\sigma_i \in \Sigma_o$), for $1 \leq i \leq n$. $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ is the $c\ell_{SF}$-indicating sequence associated with $c\ell$ and $\mathbb{S}^{c\ell} = \mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3, \ldots$ be its corresponding series. Then, $c\ell$ is an SF-indeterminate cycle if and only if the fixed-point reached by series $\mathbb{S}^{c\ell}$ is non-empty.* □

In other terms, this means that $\forall i \in \mathbb{N}^* : \mathcal{S}_i \neq \emptyset$.

Proof. The proof of Theorem 8 is omitted as it is similar to that of Theorem 4 in Chapter 4.

5.8.3 A Procedure for Checking SF-diagnosability

It is worth noticing that for the actual verification of SF-diagnosability, *a systematic procedure*, derived directly from Theorem 8, can be performed as follows:

When an SF-uncertain cycle $c\ell$ is encountered in diagnoser $\mathcal{D}$, we proceed as follows:

Generate the successive elements of $c\ell_{SF}$-indicating sequence $\rho^{c\ell}$ (starting from $\mathcal{S}_1$), and for each element $\mathcal{S}_i$ check the following conditions:

1. if $\mathcal{S}_i = \emptyset$, cycle $c\ell$ is not an SF-indeterminate cycle and therefore the procedure is stopped;
2. else, if $\exists k \in \mathbb{N} : i = 1 + kn$ (with $n = |c\ell|$), then:
   (a) if $\mathcal{S}_i = \mathcal{S}_{(i-n)} \neq \emptyset$, then cycle $c\ell$ is an SF-indeterminate cycle and the procedure is stopped;

(b) otherwise continue.

This procedure is repeated on each SF-uncertain cycle generated in $\mathcal{D}$.

Example 11 *Let us take once again model G (Figure 7) and its diagnoser $\mathcal{D}$ (Figure 5.9). Cycle $c\ell = a_2, a_3$ is an SF-uncertain one since a_2 and a_3 are both non-F-certain nodes and also F-uncertain ones. The $c\ell_{SN}$-indicating sequence associated with $c\ell$ is $\rho = \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \ldots$, such that $\mathcal{S}_1 = \{9, 6, 5, 10\} = \mathcal{S}_3 = \mathcal{S}_5 = \ldots$ and $\mathcal{S}_2 = \{7, 12, 8, 11\} = \mathcal{S}_4 = \mathcal{S}_6 = \ldots$. One can observe that ρ is a cycle composed of two (non-empty) sets of states. Therefore, according to Theorem 8 $c\ell$ is an SF-indeterminate cycle, which means that model G is non-SF-diagnosable.*

Remark 4 *Regarding the analysis of SR-diagnosability, it can be performed in an analogous manner to the analysis of SF-diagnosability. Namely, we first generate a $c\ell_{SR}$-indicating sequence starting from the faulty and recovered sets of states in the SR-uncertain cycle (i.e., a cycle in which all its nodes are non-R-certain and at least one node is an R-uncertain one). Then, one checks if the $c\ell_{SR}$-indicating sequence takes the form of a prime sequence. If it is the case, then the SR-uncertain cycle is an SR-indeterminate one, and therefore, the model G is non-SR-diagnosable. In the case where the $c\ell_{SR}$-indicating sequence reaches an empty fixed -point, then the SR-uncertain cycle is not an SR-indeterminate one.*

5.9 Discussion

In this section, we will point out the main features of the proposed approach and some comparative remarks regarding some existing works strictly related to the issues discussed in the current chapter.

- Regarding the intermittent fault modeling, in our work we do not make a difference between the first fault occurrence and the other occurrences as done in [Contant et al., 2004], where the states reached by the first occurrences of faults have a label (F_i) different from the other states reached after more than one fault occurrence have occurred, which are labeled with F_i^p. However, this feature does not affect the diagnosability verdict, since assumption (A3) ensures that each fault event occurs indefinitely. Thus, if the the model is SF-diagnosable, this means that each fault occurrence can be identified within a finite delay.

- Regarding the diagnoser construction, the initial node of our diagnoser can be an N- (or NF-node) contrarily to Contant's diagnoser, where the initial node is always N-node. This is due to the fact that in the building procedure of our diagnoser variant,

the unobservable reachability is computed before the current node is left, whereas in Contant's diagnoser the unobservable reachability is computed (in the observer) once the current node is left. This feature does not affect the diagnosability verdict, since the necessary and sufficient condition is established regarding the existence of indeterminate cycles, i.e., SF-indeterminate cycle in our work and F_i^p-indeterminate cycle in [Contant et al., 2004].

- The extension of our approach to deal with multiple fault classes can be performed in the same manner as usually done in the classic diagnoser-based approaches [Sampath et al., 1995]. That is, we can either build one specific diagnoser for each fault class. Alternatively, on can build a single diagnoser that simultaneously tracks all the fault types. This means to extend the diagnoser node structure; i.e., for instance, for a system containing m fault classes, each diagnoser node a contains $2m+1$ subsets of states (some of them may be empty): $\mathcal{X}_N, \mathcal{X}_{F_i}$, and $\mathcal{X}_{R_i}$, for $= 1, \ldots, m$. It should be noticed that in such cases, these state subsets are not necessarily disjoint.

- In our approach, the diagnoser is constructed by considering the unobservable reachability to the left (see the construction procedure in Section 5.6). Therefore, the general setting of the inverse projection mapping P_L^{-1} is considered, i.e., without restriction to the event-sequences which terminate with an observable event. This is different from the diagnoser approaches proposed in [Sampath et al., 1995, Contant et al., 2004] where the $\varepsilon-$reduction is considered when building the diagnoser. That is, the inverse projection mapping P_L^{-1} is restricted to the event-sequences that terminate with an observable event. Consequently, regarding the diagnosability of intermittent faults (for example the SF-diagnosability), it is possible that an $SF-$indeterminate cycle (called F^P-indeterminate cycle in [Contant et al., 2004]) exists in our diagnoser variant, while it does not exist in the diagnoser proposed in [Contant et al., 2004]. In other terms, the diagnosability property in [Contant et al., 2004, Sampath et al., 1995] refers to the ability to discriminate between the sequences that share the same observation and that all end with an observable event. However, in our case, we also consider any unobservable continuation following the last observed event. Therefore, the diagnosability decision may be different between the two approaches. Namely, it is possible that a system that is diagnosable in the sense of [Contant et al., 2004] is not diagnosable considering our projection mapping. For instance, the diagnosability decision can be different between the two approaches when only one observable event exists between the fault occurrence and its reset event. In this case and to comply with the context of [Contant et al., 2004], it is sufficient to add a condition that checks this particular case. Let us explain

such a particular case using the following example.

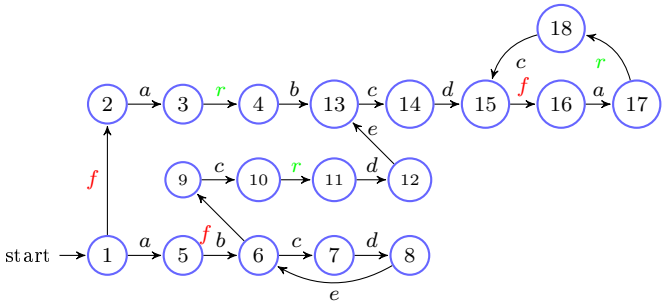


Figure 5.10 – The FSA model in Example 12

Example 12 *Consider automaton G shown in Figure 5.10 and taken from [Contant et al., 2004]. The sets of observable and unobservable events are $\Sigma_o = \{a, b, c, d, e\}$ and $\Sigma_u = \{f, r\}$, respectively. In addition, $\Sigma_f = \{f\}$ and $\Sigma_r = \{r\}$. Diagnoser $\mathcal{D}$ corresponding to model G is depicted in Figure 5.11. One can observe that it contains two SF-uncertain cycles: $c\ell_1 = a_2, a_3, a_4$ and $c\ell_2 = a_5, a_6$. Let us denote by ρ_1 and ρ_2 are the indicating sequence corresponding to $c\ell_1$ and $c\ell_2$ respectively. It is plain that $c\ell_1$ is non-SF-indeterminate cycle, since for $i = 7 : \mathcal{S}_i = \emptyset$. However, $c\ell_2$ is an SF-indeterminate cycle since $\forall i \in \mathbb{N} : \mathcal{S}_i \neq \emptyset$. therefore, model G' is non-SF-diagnosable. In [Contant et al., 2004], this model is assumed to be SF-diagnosable (more precisely Type-F^p-diagnosable, according to the definition in [Contant et al., 2004]). In fact, there is no contradictory between these two results, since the two results are obtained under a different assumption regarding the projection mapping P. Let us consider the observable sequence $s = abcda$. In one hand, according to [Contant et al., 2004] $P^{-1}(s) = \{farbcdfa\}$, which is only one event sequence that reaches a faulty states (i.e., state 17). Therefore, the model is SF-diagnosable. In the other hand, according to our assumption, $P^{-1}(s) = \{farbcdfa, farbcdfar\}$, which consists in two event sequences that respectively reach a faulty state (state 17) and recovered state (state 18). Therefore, it is not possible to decide about the recovery status of the system.*

5.10 A Still Open Issue

In this chapter, we have discussed various notions of diagnosability of intermittent faults, regarding the detection/identification of the fault event occurrences and their reset event occurrences within finite delays. However, we have not taken into account the multiplicity

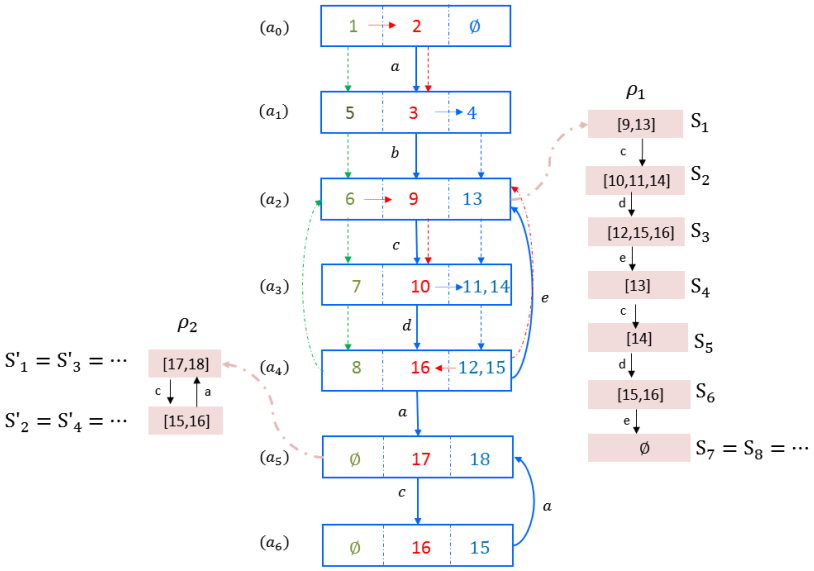


Figure 5.11 – The diagnoser of model G' (Example 12)

of fault/reset occurrences. That is, a fault can occur and recover many times before its detection/identification. In other words, these definitions can not give a verdict on how many times a fault has occurred/recovered.

A stronger version of the diagnosability property consists in detecting each fault event occurrence in a finite delay, but also before that it resets. Hereafter, we give the formal definition of such a property.

Définition 31 *(F_r-diagnosability)*
An FSA G is said to be F_r-diagnosable w.r.t. projection P, fault class Σ_f and corresponding reset event class Σ_r, if the following holds:

$$[\forall s \in \psi(\Sigma_f)] \quad (\forall t \in L/s \wedge\ t \in \overline{\psi}(\Sigma_r)) \Rightarrow D_{F_r}$$

where diagnosability condition D_{F_r} is:

$$\exists\, t' < t : \forall\, \omega \in [P_L^{-1}(P(s.t'))] \Rightarrow \ell(\omega) = F$$

with $\overline{\psi}(\Sigma_r) = \{s = (\sigma_1, \sigma_2, \ldots, \sigma_n) \in L \,|\, \forall 1 \leq i < n : \sigma_i \notin \Sigma_r \wedge \sigma_n \in \Sigma_r\}$ is the set of finite event-traces in L that have only the last event in Σ_r.

The above definition means the following: let s be a finite event-trace in L that ends with a faulty event, t be any finite continuation of s that ends with a reset event but does

not hold any reset event before this last event. Condition D_{F_r} then requires that any finite event-trace that shares the same observation with $s.t$, shall experiment a faulty behavior between the moment of the fault occurrence and its recovery. In simple words, when a fault event occurs, one needs to be able to detect it and identify the faulty status of the system before its recovering. In the same way, we can define its deal version R_f-diagnosability, which consists in detecting and identifying that the system reaches a recovered status after each occurrence of reset event and before any new occurrence of the corresponding fault event.

Remark 4 *F_r-diagnosability allow us to detect each occurrence of a fault event before its recovery, which means that we can also determine how many times a fault has occurred. Using such a definition, it becomes possible to link our fault diagnosis framework with the ones discussed in* [Jiang et al., 2003b, Yoo and Garcia, 2004, Yoo and Garcia, 2009, Zhou and Kumar, 2009, Garcia and Yoo, 2005]*, which deal with fault diagnosis of repeated faults, i.e., the ability to determine how many times a fault has occurred.*

We have introduced and discussed the concept of F_r-diagnosability in our work [Boussif and Ghazel, 2016d]. In parallel, Fabre et al. [Fabre et al., 2016] have introduced a similar definition called $T-$diagnosability, where the only difference is that faults are considered to be repairable, i.e., the recovery of the fault is a controllable (and then observable) action which can be generated by the controller. In [Fabre et al., 2016], the authors propose a necessary and sufficient condition for the analysis of F_r-diagnosability (or $T-$diagnosability) on the basis of an augmented diagnoser, i.e., a parallel composition of the diagnoser and the system model, and they show that checking such a notion of diagnosability is PSPACE-complete. However, this technique is only applied to deterministic model. In addition, the authors argue that the twin-plant technique is not sufficient to check such a diagnosability property since it cannot be characterized by pairs of equivalent paths, i.e., diagnosability is rather a global property on classes of observable-equivalent traces. For the best of our knowledge, there is no work that proposes a necessary and sufficient condition for F_r-diagnosability derived directly from the well-known diagnoser-based or twin-plant based approaches.

5.11 Conclusion

In this chapter, we have extended the diagnoser-based approach, which we have developed in Chapter 4, in order to deal with diagnosability of intermittent faults of DESs. The system modeling, intermittent fault modeling and various diagnosability properties are firstly discussed. An extended diagnoser with a new structure that allows checking the

necessary and sufficient conditions, without needing to construct any intermediate model, is established. Moreover, systematic procedures for the actual verification of the various diagnosability properties considered are established.

Chapter 6

Fault Diagnosis of LPNs Using a Symbolic Reachability Diagnoser

Sommaire

6.1 Summary

The present chapter represents an improvement of the diagnoser-based approach introduced in Chapter 4 while dealing with Petri nets. It consists in building a symbolic diagnoser called Symbolic Reachability diagnoser *(SRD) for both analyzing diagnosability and performing the online diagnosis. In particular, to obtain a compact representation of the diagnoser state-space, the nodes are encoded using a symbolic representation, i.e., Binary Decision Diagrams. However, the arcs linking the diagnoser nodes remain in an explicit representation.Furthermore, a necessary and sufficient condition for diagnosability analysis of PNs is derived on the basis of the proposed structure, and a systematic procedure for checking such a condition is proposed. Finally, we provide an on-the-fly algorithm to simultaneously construct the symbolic diagnoser and analyze diagnosability. In order to evaluate the efficiency and the scalability of the proposed approach, an experimental and comparative analysis relative to other existing technique is presented and discussed, on the basis of a Petri net benchmark. The work presented in this chapter is the subject of publications at VeCOS'15* [Boussif et al., 2015] *and two submitted journal papers to IJCCBS* [Boussif et al., 2016b] *and IEEE-SMC* [Boussif et al., 2016a].

This chapter is structured as follows: in Section 6.2, a brief survey of fault diagnosis in Petri net (PN) framework is presented. In Section 6.4, basic PN notations and the definition of diagnosability are introduced. In Section 6.5, the concept of BDDs and the construction of the symbolic observation graph is discussed. Section 6.6 is devoted to discussing the construction of SRD. In Section 6.7, we discuss the diagnosability analysis using the SRD structure. In Section 6.8, an on-the-fly algorithm for simultaneously constructing the SRD and checking diagnosability is proposed. Section 6.9 presents some experimentations to illustrate the effectiveness and the scalability of our approach while taking the MBRG/BRD approach [Cabasino et al., 2009b] as a reference. Finally, Section 6.10 draws some concluding remarks.

6.2 Petri Net Based Fault Diagnosis

The early works that addressed fault diagnosis issues mostly considered finite state automaton models [Lin, 1994, Sampath et al., 1995, Zad et al., 2003, Jiang and Huang, 2001, Yoo and Lafortune, 2002b, Cimatti et al., 2003]. Afterwards, fault diagnosis issues have also been dealt with within the Petri nets (PNs) framework [Lefebvre and Leclercq, 2015, Lefebvre et al., 2013, Wen et al., 2005, Ramírez-Treviño et al., 2007, Basile et al., 2008, Basile et al., 2009, Basile et al., 2010, Lefebvre, 2014, Lefebvre and Delherm, 2007, Basile, 2014, Jiroveanu and Boel, 2010, Cabasino et al., 2009a, Germanos et al., 2015, Madalinski and Khomenko, 2010, Ushio et al., 1998, Chung, 2005, Jiroveanu and Boel, 2004, Cabasino et al., 2010, Cabasino et al., 2009b, Liu et al., 2014b, Li et al., 2015c]. PNs were created by German mathematician Carl Adam Petri for the purpose of describing chemical processes [Petri, 1966]. PNs are a graph-based mathematical formalism that allow the analysis and verification of concurrent system behavior, based on their graphical and mathematical representations [Murata, 1989].

The diagnosis approaches based on PN models can be classified into two main classes:
1) Methods based on the structural representation of PNs

The mathematical foundation underlying PNs, allows the use of standard techniques, such as Integer Linear-Programming (ILP) to perform online diagnosis and investigate diagnosability. Regarding the diagnosability investigation, various sufficient conditions have been proposed. Authors in [Wen et al., 2005] have proposed an approach for analyzing diagnosability by checking the structural properties of T-invariants under the assumption that the net marking and the transitions are partially observable. In [Ramírez-Treviño et al., 2007], an *interpreted diagnoser* has been developed for fault diagnosis of interpreted Petri nets (IPNs), where a sufficient condition has also been developed on the basis of T-invariant properties. A linear-programming-based polynomial algorithm has been es-

tablished in [Wen et al., 2005] for computing a sufficient condition of diagnosability. In a series of works [Basile et al., 2008, Basile et al., 2009, Basile et al., 2010], Basile et al. have developed two conditions regarding diagnosability: the first is a necessary condition, while the second is a sufficient one. The elaborated technique uses the concept of *g-marking* introduced for online fault detection and used to derive an interpreted diagnosser, i.e., an algorithm based on the online solution of ILP problems [Basile et al., 2009]. More recently, a necessary and sufficient condition for analyzing K-diagnosability (i.e., diagnosability in k steps) was established in [Basile et al., 2012a] on the basis of an ILP optimization tool. The general idea of these approaches based on structural representations of PNs, is to develop an interpreted diagnoser that, based on the observable transitions, infers the current status of the system, by using an ILP algorithm [Basile et al., 2007, Dotoli et al., 2009, Basile et al., 2009].

2) Methods based on the behavioral representation of PNs

The behavioral representation can be advantageously used to extend the automata-based approaches so as to deal with fault diagnosis of Petri nets by considering its reachability grah. In fact, this consists in building some particular graphs to deal with diagnosis issues, which lead to reduce sizes with respect to the reachability graph [Basile, 2014].

Some interesting works were inspired from the *twin-plant/verifier* approaches [Jiang and Huang, 2001, Yoo and Lafortune, 2002b, Cimatti et al., 2003]. In [Jiroveanu and Boel, 2010], an automaton called '*ROF-automaton*' is used for checking the diagnosability of bounded nets by adapting the verifier approach [Yoo and Lafortune, 2002b]. *ROF*-automaton is efficiently constructed on the basis of the computation of the minimal explanations of faults, which serves to generate a state-space that is significantly smaller than the reachability graph. The authors in [Cabasino et al., 2009a] have developed an approach for analyzing diaganosability of unbounded Petri nets on the basis of a net called '*verifier net*' and the corresponding coverability graph. In fact, the *verifier net* is obtained by a (parallel) composition of the PN model and a copy that depicts only the normal behavior. Similar approaches, in the sens of considering parallel composition of PNs, have been proposed in [Germanos et al., 2015, Madalinski and Khomenko, 2010], with the particularity that the diagnosabilily issue was reduced to an LTL-X model-checking problem and solved using parallel model-checking based on Petri net unfoldings. As mentioned earlier in this thesis, the main drawback of the approaches based on *verifiers/twin-plant* approaches, is that they deal only with diagnosability analysis and do not consider online diagnosis.

Actually, the *diagnoser-based* approaches [Sampath et al., 1995, Zad et al., 2003] (which consist in constructing observers/diagnosers) remain the principal techniques which deal with both diagnosability analysis and online diagnosis. Consequently, these approaches

have been extended in order to cope with PN models. In [Ushio et al., 1998], two kinds of diagnosers (ω- and ω-refined diagnosers) have been proposed and a sufficient condition for diagnosability of unbounded PNs, under the assumption that the net marking is observable and all transitions are unobservable is established. The authors in [Chung, 2005] have discussed a similar approach with the assumption that only a subset of transitions is observable. In [Jiroveanu and Boel, 2004], an approach for fault detection of bounded PNs was proposed using a reduced observer and a backward analysis on the net structure. Also, an approach for online diagnosis of bounded PNs was proposed in [Cabasino et al., 2010]. It consists in constructing a compiled diagnoser, called *Basis Reachability Graph* (BRG), by using the concept of minimal explanations. Actually, the minimal explanations are firing count vectors associated with the set of minimal sequences of unobservable transitions that explain the firing of an observable transition.

Recently, Lefebvre and Leclercq [Lefebvre and Leclercq, 2015, Lefebvre et al., 2013] have proposed an approach to deal with diagnosability of both bounded and unbounded Petri nets under partial observation (regarding both transitions and places). The main idea behind this work is to transform the coverability graph into an observation graph that encodes all the observation sequences with respect to a sensor configuration. The aim of such graphs is to represent all observation sequences collected with a given measurement function and also to encode all firing sequences consistent with the observation sequences. On the basis of the analysis of paths and circuit in the observation graph, a necessary and sufficient condition is established.

The main approach which deals with both online diagnosis and offline diagnosability analysis has been proposed in [Cabasino et al., 2009b]. In this approach, two graphs are presented: an observer called *'modified basis reachability graph'* (MBRG) and a diagnoser called *'the basis reachability diagnoser'* (BRD). Although in most of the cases these two graphs are in general smaller than the reachability graph, the proposed procedure to build the MBRG can require a number of steps equal to the cardinality of the reachability graph. Furthermore, the proposed diagnosability test requires to check the existence of cycles in the BRD, which, in the worst case, implies an exponential complexity in time [Basile et al., 2012a].

Recently, an on-the-fly approach for analyzing diagnosability and $K/K_{min}-$diagnosability was proposed in [Liu, 2014, Liu et al., 2014b]. The approach consists in constructing, on the fly, two graphs, called FM-graph and FM-set graph, and simultaneously analyzing diagnosability. Moreover, the generated FM-set graph can be used for performing online diagnosis when the PN model is diagnosable. The key point of this approach is that, in general, only a part of the reachability graph is generated to investigate diagnosability and perform online diagnosis. Some improvements of this approach, using the minimal

explanation concept and T-invariant properties were proposed in [Li et al., 2015c, Li et al., 2015b].

To get a general overview on the literature pertaining to the fault diagnosis of DESs, the reader can refer to the recent surveys [Zaytoon and Lafortune, 2013] (a general survey), [Basile, 2014] (a survey on PN-based approaches) where theoretical and practical issues are discussed.

6.3 Motivation of the Approach

On the basis of the behavioral representation, *diagnoser-based* approaches remain the principal techniques which deal with both diagnosability analysis and online diagnosis. Nevertheless, as discussed in Section 4.1, these approaches mainly suffer from some difficulties. Namely, the high complexity of constructing the diagnoser, the need of an intermediate model and the double-checking procedure for analyzing diagnosability. In the previous chapters (Chapters 4 and 5), we have presented an efficient diagnoser variant approach, which partially overcomes these issues, as witnessed by the experimental results in Section 4.6. Nevertheless, we have pointed that the main drawback of our approach is the number of nodes (in terms of memory) in the diagnoser, due to the unobservable reachability. In fact, the nodes of our diagnoser may contain a large number of the model states that are reachable through unobservable sequences after the occurrence of an observable event. To deal with this issue of memory consumption in the diagnoser, we want to investigate the symbolic representation using binary decision diagrams (BDDs). In fact, BDDs can efficiently encode and manage the sets of states in the diagnoser nodes.

The proposed technique is inspired from the Symbolic Observation Graph (SOG) [Haddad et al., 2004, Klai and Petrucci, 2008] which combines symbolic and enumarative representations in order to build a deterministic observer from a partially observed model. The symbolic observation graph was firstly used for the formal verification using event-based model-checking as an efficient alternative to the Kripke structure.

In fact, the approach we develop allows for analyzing the diagnosability of bounded labeled PNs and for performing the online diagnosis on the basis of a deterministic symbolic automaton called SRD, for '*Symbolic Reachability Diagnoser*', derived directly from the PN model. The SRD has a particular structure, which consists in separating normal markings from faulty ones in each diagnoser node and then encode the two obtained subsets using BDDs. This aims in one hand at reducing the memory requirements and on the other hand at speeding up the verification process. On the basis of this structure, we propose a systematic procedure to check the necessary and sufficient condition for diagnosability using only the SRD (as presented in Chapter 4). An on-the-fly algorithm

is then developed for simultaneously constructing the SRD and analyzing diagnosability.

In what follow, we point out the main features of the proposed approach:

1. The approach provides a new structure for representing the diagnoser nodes. Such a structure explicitly separates the normal markings from the faulty ones in each node. This feature serves to track normal and faulty traces more efficiently.

2. The approach combines enumerative and symbolic representations to construct the diagnoser. The main idea consists in:

   - using binary decision diagrams (BDDs) to compact and handle the sets of markings in the diagnoser nodes, which serves to reduce the memory consumption;
   - using an explicit representation for the (observable) transitions that link the diagnoser nodes. Such a representation allows for an easy exploration of the SRD paths.

3. In the same way as for existing diagnoser-based approaches, the SRD serves to both check diagnosability and perform online diagnosis.

4. In the developed approach, the SRD is directly built from the original LPN without requiring to construct any intermediate model as usually done in the diagnoser-based approaches, e.g., generator or pre-diagnoser in [Sampath et al., 1995], MBRG in [Cabasino et al., 2014], FM-graph [Liu, 2014].

5. On the basis of the SRD structure, a sufficient condition for checking the undiagnosability of LPNs is derived. Such a condition is used for the on-the-fly verification of diagnosability. Therefore, the model is stated to be non-diagnosable as soon as such a condition is met, without needing to build and analyze the whole state-space of the SRD.

6. The developed approach provides a systematic procedure for checking the necessary and sufficient condition for diagnosabilily (i.e., the existence or not of F-indeterminate cycle) without using any intermediate model. This procedure serves to improve the verification time.

7. An on-the-fly depth-first search algorithm for both constructing the SRD and verifying diagnosability simultaneously is proposed. Such an algorithm serves to generate as small state-space as possible and then, improves the memory/time consumption.

In addition to the above-mentioned features, a dedicated tool, called 'DPN-SOG' tool (for **D**iagnosability analysis of **P**etri **N**ets using **S**ymbolic **O**bservation **G**raphs) implementing the proposed approach has been developed. In order to assess the efficiency and

the scalability of the approach, some experimentations have been conducted through a PN benchmark.

In the section below, we briefly recall the syntax and semantics of the bounded labeled Petri nets (LPNs), as well as the diagnosability analysis of such models.

6.4 Preliminaries

6.4.1 Labeled Petri Net Modeling

A Petri net is a structure $N = (P, T, Pre, Post)$, where P is a finite set of places; T is a finite set of transitions; Pre and $Post$ are the pre- and post-incidence mappings, respectively. $C = Post - Pre$ is the incidence matrix. A marking is a vector $m \in \mathbb{N}^{|P|}$ that assigns a non-negative integer to each place. We denote by $m(p)$ the marking of a place p. A marked PN (N, m_0) is a PN N with a given initial marking m_0. For short, a marked PN will be called PN afterward.

A transition t_i is enabled by marking m, denoted by $m[t_i >$, if $m(p) \geq Pre(p, t_i), \forall p \in P$. A transition t_i enabled by marking m can fire, yielding to a marking $m' = m + C \cdot \vec{t_i}$, where $\vec{t_i} \in \{0,1\}^{|T|}$ is a vector in which only the entry associated with transition t_i is equal to 1, the other entries are 0. Then, marking m' is said to be reachable from marking m by firing the transition t_i, also denoted by $m \,[\, t_i > m'$.

A sequence of transitions $s = t_1 t_2 \dots t_k$ is firable at marking m, denoted by $m \,[\, s >$, if $\exists m_1, m_2, m_{k-1}$ s.t. $m \,[\, t_1 > m_1 \,[\, t_2 > \cdots \; m_{k-1} \,[\, t_k >$. The reached marking m' is computed by $m' = m + C \cdot \pi(s)$, and denoted by $m \,[\, s > m'$, where $\pi(s) = \sum_{i=1}^{k} \vec{t_i}$ is the firing vector relative to s.

A marking m is reachable in (N, m_0) if and only if there exists a firing sequence s such that $m_0 [s > m$. The set of all markings reachable from m_0 defines the *reachability set* of (N, m_0) and is denoted by $R(N, m_0)$.

A PN (N, m_0) is bounded if the number of tokens in each place does not exceed a finite number $b \in \mathbb{N}$ for any marking reachable from m_0. Formally, $\exists\, b \in \mathcal{N}$ s.t. $\forall m \in R(N, m_0), \forall p \in P : m(p) \leq b$. A PN is live if, no matter what marking has been reached from m_0, it is possible to ultimately fire any transition of the net by progressing through some further firing sequence [Murata, 1989]. Formally, $\forall m \in R(N, m_0), \forall t \in T, \exists s \in T^* : m \,[\, s.t >$.

A labeled Petri net (LPN) is a tuple $N_L = (N, m_0, \Sigma, \varphi)$, where (N, m_0) is a marked PN, Σ is a finite set of events (i.e., labels) and $\varphi \colon T \to \Sigma$ is the transition labeling function. φ is also extended to sequences of transitions, $\varphi \colon T^* \to \Sigma^*$. The language generated by N_L is $\mathcal{L}(N_L) = \{\varphi(s) \in \Sigma^* \mid s \in T^*,\ m_0 \,[\, s >\}$. For short, we write $\mathcal{L}$ instead of $\mathcal{L}(N_L)$.

Also, one should notice that various transitions can share the same event label, i.e., φ is not bijective. We denote by T_σ the set of transitions sharing the same event σ, i.e., $T_\sigma = \{t \in T : \varphi(t) = \sigma\}$.

6.4.2 Diagnosability of LPNs

Due to the partial observability, the set of transitions is partitioned as $T = T_o \uplus T_u$, where T_o is the set of observable transitions, and T_u is the set of unobservable transitions. The set of unobservable transitions is also partitioned into two subsets $T_u = T_f \uplus T_{reg}$ where T_f is the set of fault transitions while T_{reg} includes the regular (i.e., non-faulty) unobservable transitions. As we deal with labeled Petri nets, the event set Σ can also be partitioned into two disjoint sets, $\Sigma = \Sigma_o \uplus \Sigma_u$, where Σ_o is a finite set of observable events and Σ_u is a finite set of unobservable events. Fault events denoted by set Σ_f are unobservable, thus $\Sigma_f \subseteq \Sigma_u$. Moreover, the set of unobservable events can be partitioned into two disjoint sets, $\Sigma_u = \Sigma_f \uplus \Sigma_{reg}$, where $\Sigma_{reg} = \Sigma_u \backslash \Sigma_f$ is the set of regular unobservable events, i.e., non-faulty unobservable events. In addition, the set of fault events Σ_f can be further partitioned into various fault classes, i.e., $\Sigma_f = \biguplus_{i=1}^{m} \Sigma_{f_i}$, where $\Sigma_{f_i} (i = 1, 2, \ldots, m)$ denotes the i^{th} class of faults.

Let $P_o \colon \Sigma^* \to \Sigma_o^*$ be the projection mapping which *erases* the unobservable events in any given sequence $u \in \Sigma^*$. The inverse projection operator P_o^{-1} is defined as $P_o^{-1}(v) = \{u \in \mathcal{L} \mid P_o(u) = v\}$ for $v \in \Sigma_o^*$. Given a live and prefix-closed language $\mathcal{L} \subseteq \Sigma^*$ and an event-sequence $u \in \mathcal{L}$, the post-language of $\mathcal{L}$ upon u denoted by $\mathcal{L}/u$ is $\mathcal{L}/u = \{v \in \Sigma^* | \ uv \in \mathcal{L}\}$. We denote by $|u|$ the length of event sequence u, and the i^{th} event of u by u^i. Also, for $a \in \Sigma$ and $u \in \Sigma^*$, we write $a \in u$ if $\exists\, i$ s.t. $u^i = a$. By abuse of notation, we note $\Sigma_{f_i} \in u$ to indicate that $\exists f_i \in \Sigma_{f_i}$ s.t. $f_i \in u$.

Without loss of generality, in the sequel we will consider one single fault class Σ_f. The generalization of the technique to multiple fault classes is performed in the same way as discussed in Section 4.5.3.

Définition 32 *(Diagnosability of LPNs)*

A given LPN N_L is diagnosable w.r.t. fault class Σ_f and projection P_o if:

$(\exists\, n \in \mathbb{N})\ (\forall u \in \mathcal{L}, u^{|u|} \in \Sigma_f)\ (\forall v \in \mathcal{L}/u)$*:*

$$|P_o(v)| \geq n \Rightarrow [\forall \omega \in P_o^{-1}(P_o(uv)) : \Sigma_f \in \omega]. \qquad \square$$

The above definition means that an LPN is diagnosable if for every trace u ending with a fault event (which corresponds to a fault transition) of type Σ_f, and for any sufficiently long continuation v of u, all traces ω having the same observable projection of uv contain at least one fault event. In other words, diagnosability of an LPN implies that each occurrence of a fault can be detected after a finite number of transition firings.

In what follows, we consider the following assumptions that the LPN model under investigation has to fulfill:

- The LPN is deadlock-free, i.e., every reachable marking enables at least one transition;
- The LPN is bounded with an upper bound $b \in \mathbb{N}^+$, i.e., $\forall p \in P, M(p) \leq b$;
- The LPN has a known structure as well as a known initial marking m_0;
- The LPN has no executable cycle of unobservable transitions.
- The faults are permanent, i.e., the model remains infinitely faulty after the occurrence of a fault.

Example 3 *Let us consider the LPN N_L in Figure 6.1, where the set of observable transitions is $T_o = \{t_2, t_5, t_7\}$ and the set of unobservable transitions is $T_u = \{t_1, t_3, t_4, t_6\}$. The labeling function is $\varphi(t_1) = \varphi(t_3) = \varepsilon$, $\varphi(t_2) = a$, $\varphi(t_5) = \varphi(t_7) = b$, and $\varphi(t_4) = \varphi(t_6) = f$. In Figure 6.1, the observable transitions are depicted with grey boxes, regular transitions are in white boxes, while faulty transitions are depicted with red boxes.*

Let us consider observable sequence $\omega = (ab)^*$. One can observe that the infinite firing sequences $s_1 = (t_1 t_2 t_5)^*$ and $s_2 = (t_1 t_4 t_2 t_5)^*$ share the same observation ω and that s_1 is fault-free while s_2 contains faulty transition t_5. Consequently, there is no finite delay upon which one can infer the fault occurrence with certainty. Thus, according to Definition 32, LPN N_L is non-diagnosable.

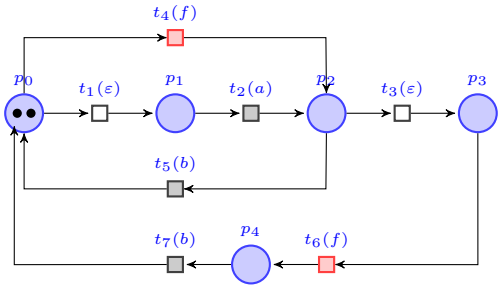


Figure 6.1 – A labeled Petri net (Example 3)

6.5 The Symbolic Observation Graph (SOG)

In this section, we recall the concept of *symbolic observability graph* (SOG) introduced in [Haddad et al., 2004], as an abstraction of the partially observed labeled transition

systems (and by extension, the reachability graph of Petri nets) for the purpose of the verification using LTL-X model-checking. Firstly, we recall the concept of binary decision diagrams (BDDs).

6.5.1 Binary Decision Diagrams

Binary decision diagrams (BDDs), introduced by [Akers, 1978] and improved in [Bryant, 1992], are compact (i.e., symbolic) representations of Boolean functions that enable the encoding and implicit manipulation of sets of states and/or transitions, with no need for explicit enumeration. Nowadays, BDDs are used in a wide range of research areas, such as formal verification, AI planning, etc.

Définition 33 *(binary decision diagram (BDD))*

A BDD is a rooted, directed acyclic graph with:

- *one or two terminal nodes of out-degree zero labeled false or true;*
- *a set of variable nodes u of out-degree. The two outgoing edges are given by two functions: low(u) and high(u);*
- *a variable var(u) is associated with each node.* □

In order to express operations on Boolean functions in terms of efficient graph algorithms, the BDD needs to be reduced and ordered.

Définition 34 *(Ordered BDD)*

A BDD is ordered (OBDD), if on all paths through the graph, the variable respect a given linear order $x_1 < x_2 < \cdots < x_n$. □

Définition 35 *(Reduced OBDD)*

An OBDD is reduced (ROBDD), if the following properties hold:

1. ***Uniqueness**: no two distinct nodes u and v have the same variable name and low- and high-successor, i.e.,* $var(u) = var(v) \wedge low(u) = low(v) \wedge high(u) = high(v)) \Rightarrow u = v$;
2. ***Non-redundant tests**: no variable node u has identical low- and high-successors, i.e.,* $low(u) = high(u)$. □

For instance, for safe Petri nets (where every place is marked with at most one token), one can consider each marking as a Boolean vector $v = \{0, 1\}^m$, m is the number of places

in the net. Then, a set of markings $S \subseteq R(N, m_0)$ can be represented within a BDD by means of the Boolean characteristic function $F_R : R(N, m_0) \to \{0, 1\}$ with:

$$F_R(s) := \begin{cases} 1, & \text{if} \quad s \in S \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

Example 4 *Let us consider a safe Petri net containing four places p_0, p_1 , p_2 , p_3, and let S be the set of markings where only one place is marked. The truth table of the corresponding characteristic function is shown in Table 6.1.*

In order to construct the BDD graph associated with this example, we define $V = \{v_0, v_1, v_2, v_3\}$ as the set of totally ordered variables (e.g., $v_0 < v_1 < v_2 < v_3$). The constructed BDD, depicted in Figure 6.2, is actually an ROBDD, since there are no isomorphic sub-graphs and no redundant nodes. Dotted (resp. solid) outgoing arc of a node u represents the successor $low(u)$ (resp. $high(u)$). A path leading to a true leaf corresponds to a marking in S.

Table 6.1 – The truth table of the corresponding characteristic function

p_0	p_1	p_2	p_3	f_R
1	0	0	0	1
0	1	0	0	1
0	0	1	0	1
0	0	0	1	1
–	...		...	0

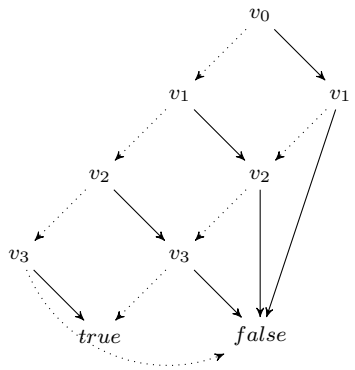


Figure 6.2 – A BDD representing the set of marking S (Example 4)

6.5.2 The Construction of the SOG

In [Haddad et al., 2004], the authors introduced the SOG as an abstraction of the reachability graph of concurrent systems and proved that the verification of an event-based formula of LTL-X on the SOG is equivalent to the verification on the original reachability graph. The main idea is to combine the on-the-fly construction and the compact representation (using BDD techniques) of the model, in order to check LTL-X state-based properties over finite models. Thus, instead of composing the whole system with the Buchi automaton representing the negation of the formula to be checked, the authors propose to make the synchronization of the automaton with an abstraction of the original reachability graph of the system.

In order to define the SOG and then define *symbolic reachability diagnoser* (SRD), we introduce the following notations:

- Given a subset of transitions $T' \subseteq T$, we define $Enable_{T'}(m) = \{t \in T' \mid m\,[\,t>\}$, as the set of transitions in T' that are enabled at marking m.

The extension to a subset of markings $M' \subseteq R(N, m_0)$, is $Enable_{T'}(M') = \{t \in T' \mid \exists\, m \in M' : m\,[\,t>\}$ which denotes the set of transitions in T' enabled at a marking belonging to M', i.e., $Enable_{T'}(M') = \bigcup_{m \in M'} Enable_{T'}(m)$.

- Given a subset of markings $M \subseteq R(N, m_0)$ and a transition $t \in T$, we define $Img(M, t) = \{m' \in R(N, m_0) \mid \exists m \in M : m\,[\,t>\,m'\}$ as the set of markings reachable from the markings in M by firing transition t.

The generalization to a subset of transitions $T' \subseteq T$ is $Img(M', T') = \bigcup_{t \in T'} Img(M', t)$.

- Given a marking $m \in R(N, m_0)$ and a subset of transition $T' \subseteq T$, we define $Reach_{T'}(m) = \{m\} \cup \{m' \in R(N, m_0) | (\exists s \in T'^*) : m\,[\,s>\,m'\}$ as the set of markings reached by firing a sequence of transitions in T' from marking m (will be used particularly for the unobservable reachability).

The generalization of this notion to a subset of markings $M \subseteq R(N, m_0)$ is $Reach_{T'}(M) = \bigcup_{m \in M} Reach_{T'}(m)$.

The construction of the SOG is guided by the set of events occurring in the formula to be checked. Such events are said to be observable while the other events are unobservable. The SOG is then represented as a graph where each node (called an aggregate) is a set of states (reachable by firing unobservable events) handled efficiently using BDD techniques [Bryant, 1992]. Aggregates of the SOG are linked by edges which are labeled with observable events. The SOG is said to be hybrid, since it is both an explicit and a

symbolic structure: the graph is represented explicitly while the nodes are sets of states encoded and managed symbolically.

Despite the theoretical exponential complexity of the size of a SOG, it has a very moderate size in practice (see [Haddad et al., 2004] for the theoretic details and [Klai and Petrucci, 2008] for experimental results). In the following, we first define what an aggregate formally is, before providing a formal definition of a SOG associated with an LPN.

Définition 36 *(aggregate)*

Consider an LPN $N_L = (N, m_0, \Sigma, \varphi)$*, with* $N = (P, T, Pre, Post)$ *and* $T = T_o \biguplus T_u$*. An aggregate* a *is a non-empty set of markings satisfying:* $m \in a \Leftrightarrow Reach_{T_u}(m) \subseteq a$*.* □

In fact, each aggregate in the SOG contains the markings reachable through the firing of an input observable transition and all the successor markings reachable through the firing of a sequence (possibly empty) of unobservable transitions.

Définition 37 *(Symbolic Observation Graph)*

The symbolic observation graph $\mathcal{G}$ *associated with an LPN* $N_L = (N, m_0, \Sigma, \varphi)$ *is a deterministic graph,* $\mathcal{G} = \langle \mathcal{A}, \Sigma_o, \rightarrow_{\Sigma_o}, a_0 \rangle$*, with* Σ_o *is the set of observable events and:*

1. $\mathcal{A}$ *is a finite set of aggregates such that:*
   a) $a_0 \in \mathcal{A}$ *is the initial aggregate and* $a_0 = Reach_{T_u}(m_0)$*.*
   b) $\forall\, a \in \mathcal{A}, \forall\, \sigma \in \Sigma_o$*, if* $\exists\, m \in a, \exists\, t \in T_\sigma$*,* $\exists\, m' \in R(N, m_0)$*:* $m\,[t > m'$ *then,* $Reach_{T_u}(m') \subseteq a'$ *for some aggregate* a' *and* $(a, \sigma, a') \in \rightarrow_{\Sigma_o}$*;*
2. $\rightarrow_{\Sigma_o} \subseteq \mathcal{A} \times \Sigma_o \times \mathcal{A}$ *is the transition relation, obtained by applying* 1.*b*)*.*

The SOG can be constructed by starting with the initial aggregate a_0 and iteratively adding new aggregates as long as the condition of 1.*b*) holds (see [Haddad et al., 2004] for the construction algorithm). In fact, the SOG can be viewed as an observer [Cassandras and Lafortune, 2009] whose macro-states are represented symbolically.

Example 13 *Let us consider the LPN* N_1 *depicted in Figure 6.3 with its reachability graph. The set of observable transitions is* $T_o = \{t_2, t_5\}$ *and the set of unobservable transitions is* $T_u = \{t_1, t_4\}$*. The labeling function is* $\varphi(t_1) = \varphi(t_4) = \varepsilon$*,* $\varphi(t_2) = a$*,* $\varphi(t_5) = b$*.*

The SOG corresponding to LPN N_1 is depicted in Figure 6.4. It is composed of three aggregates, each one contains a set of markings, with $m_0 = 2\ 0\ 0, m_1 = 1\ 1\ 0, m_2 =$

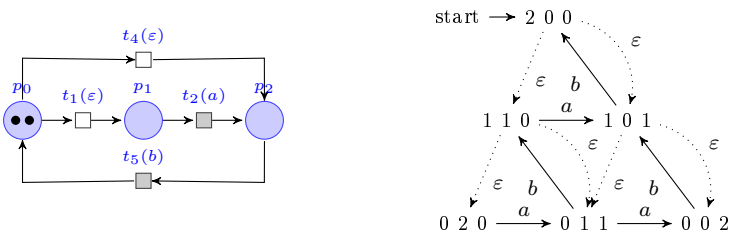


Figure 6.3 – LPN N_1 and its reachability graph (Example 13)

$1\,0\,1, m_3 = 0\,2\,0, m_4 = 0\,1\,1, m_5 = 0\,0\,2$. In this example, the markings in each aggregate are enumerated. However, in the actual representation such sets of marking are encoded as BDDs, in the same manner as shown in Example 4.

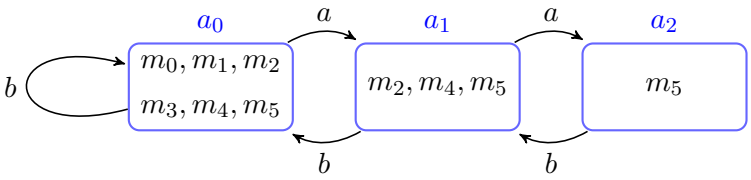


Figure 6.4 – The SOG corresponding to LPN N_1 in Example 13

6.6 The Symbolic Reachability Diagnoser (SRD)

In this section, we discuss how the SOG can be used in order to build a symbolic diagnoser, which we call '*symbolic reachability diagnoser (SRD)*.

6.6.1 The Structure of the Diagnoser Node

In order to capture the main feature for analyzing diagnosability, which is to keep tracking the ambiguous behavior of the system, i.e., normal and faulty executions that share the same observable event sequence, we modify the structure of the aggregates of the SOG. Therefore, each node in our diagnoser (SRD) is partitioned into two distinct subsets of markings, each of them is encoded using a BDD.

1. *the set of normal markings* (denoted by $\mathcal{M}_N$), which is the subset of markings in the node that are reachable by firing faulty-free sequences.

2. *the set of faulty markings* (denoted by $\mathcal{M}_F$), which is the subset of markings in the node that are reachable by firing faulty sequences.

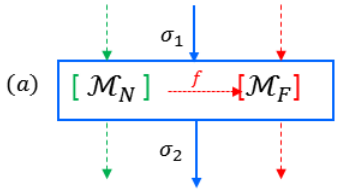


Figure 6.5 – The general structure of the SRD node

Moreover, there may exist some faulty transitions that link some markings in $\mathcal{M}_N$ to some others in $\mathcal{M}_F$ within the same node. The existence of such transitions is also encoded within each node using a Boolean variable. The general structure of the diagnoser node is depicted in Figure 6.5.

We will show in the sequel how such a structure of the SRD nodes can be advantageously explored for rendering diagnosability analysis more efficiently than using the classic structure of diagnosers [Sampath et al., 1995, Cabasino et al., 2014].

According to this structure of nodes, one can differentiate between three types of diagnoser nodes, in the same way as in the classic diagnoser approaches:

- *N-certain diagnoser node*: is a diagnoser node of which the set of faulty markings is empty ($\mathcal{M}_F = \emptyset$);
- *F-certain diagnoser node*: is a diagnoser node of which the set of normal marking is empty ($\mathcal{M}_N = \emptyset$);
- *F-uncertain diagnoser node*: is a diagnoser node of which neither the normal set, nor the faulty set of markings, is empty, i.e., $\mathcal{M}_N \neq \emptyset$ and $\mathcal{M}_F \neq \emptyset$.

In order to simplify the notation, we use $a.\mathcal{M}_N$ (resp. $a.\mathcal{M}_F$) to indicate the set of normal markings $\mathcal{M}_N$ (resp. set of faulty markings $\mathcal{M}_F$) of a given diagnoser node a.

6.6.2 Definition of the SRD

The SRD can be defined as a directed deterministic graph, where each node is composed of two BDDs encoding the subsets of normal and faulty markings, while the arcs are labeled by observable events.

Définition 38 *(Symbolic Reachability Diagnoser)*
The SRD associated with an LPN N_L is a directed deterministic graph $\mathcal{D} = \langle \Gamma, \Sigma_o, \delta_{\mathcal{D}}, \Gamma_0 \rangle$, with $N = (P, T, Pre, Post)$ and $T = T_o \biguplus T_u$, where:

1. Γ *is a finite set of diagnoser nodes;*
2. Σ_o *is a finite set of events associated with a finite set of observable transitions* T_o*;*
3. Γ_0 *is the initial diagnoser node with:*
   a) $\Gamma_0.\mathcal{M}_N = Reach_{T_{reg}}(m_0)$*;*
   b) $\Gamma_0.\mathcal{M}_F = Reach_{T_u}(Img(\Gamma_0.\mathcal{M}_N, T_f))$.
4. $\delta_{\mathcal{D}} : \Gamma \times \Sigma_o \to \Gamma$ *is the transition relation, defined as follows:*
   $\forall a, a' \in \Gamma,\ \sigma \in \Sigma_o$: $a' = \delta_{\mathcal{D}}(a, \sigma) \Leftrightarrow a'.\mathcal{M}_N = Reach_{T_{reg}}(Img(a.\mathcal{M}_N, T_\sigma)) \wedge a'.\mathcal{M}_F = Reach_{T_u}(Img(a'.\mathcal{M}_N, T_f) \cup Img(a.\mathcal{M}_F, T_\sigma))$

where $T_\sigma = \{t \in T : \varphi(t) = \sigma\}$. □

In summary, SRD $\mathcal{D}$ is constructed as follows: let the current node be a and let σ be an observable event, such that:

$$\exists\, t \in T_\sigma; \exists\, m \in \mathcal{M}_N \cup \mathcal{M}_F \ : m\,[\,t>$$

The target diagnoser node a' reachable from a by occurrence of σ is computed following the rules below:

1. If $Enable_T(a.\mathcal{M}_N) \cap Enable_T(a.\mathcal{M}_F) \cap T_\sigma \neq \emptyset$, then:
   - $a'.\mathcal{M}_N = Reach_{T_{reg}}(Img(a.\mathcal{M}_N, T_\sigma))$
   - $a'.\mathcal{M}_F = Reach_{T_u}(Img(a'.\mathcal{M}_N, T_f) \cup Img(a.\mathcal{M}_F, T_\sigma))$
2. If $Enable_T(a.\mathcal{M}_N) \backslash Enable_T(a.\mathcal{M}_F) \cap T_\sigma \neq \emptyset$, then:
   - $a'.\mathcal{M}_N = Reach_{T_{reg}}(Img(a.\mathcal{M}_N, T_\sigma))$
   - $a'.\mathcal{M}_F = Reach_{T_u}(Img(a'.\mathcal{M}_N, T_f))$
3. If $Enable_T(a.\mathcal{M}_F) \backslash Enable_T(a.\mathcal{M}_N) \cap T_\sigma \neq \emptyset$, then:
   - $a'.\mathcal{M}_N = \emptyset$
   - $a'.\mathcal{M}_F = Reach_{T_u}(Img(a.\mathcal{M}_F, T_\sigma))$

These aforementioned rules preserve a specific fault propagation scheme regarding the assumption that faults are assumed to be permanent. Such rules can be summarized in three points as depicted in Figure 6.6:

- From an *N-certain* diagnoser node, either an *N-certain* diagnoser node or an *F-uncertain* one can be reached;
- From an *F-certain* diagnoser node, only *F-certain* diagnoser nodes can be reached;
- From an *F-uncertain* diagnoser node, either an *F-uncertain*, an *N-certain* or an *F-certain* diagnoser node can be reached.

Since all the successors of an *F-certain* node are also *F-certain*, it is unnecessary to build them (i.e., the subsequent *F-certain* nodes) because they do not bring new information from the diagnosis point of view. Indeed, as regards diagnosability analysis, and given the necessary and sufficient condition of diagnosability established in [Sampath et al., 1995], only the analysis of *F*-uncertain cycles is necessary. Thus, as faults are permanent, one can be certain that no such cycles can exist subsequently to an *F-certain* node. As for online diagnosis, based on the SRD, once an *F-certain* node is reached, one can be sure that the system will remain indefinitely faulty.

It should be noticed here that the fault propagation rules of the SRD are different from those of the classic diagnosers [Sampath et al., 1995, Cabasino et al., 2014]. Indeed, an *F-certain* diagnoser node cannot be reached directly from an *N-certain* diagnoser node. This is due to the fact that in the building procedure of the SRD, the unobservable reachability is computed before the current node is left, i.e., before considering the occurrence of any further observable event.

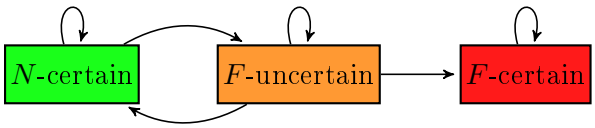


Figure 6.6 – Fault propagation on the SRD

In the same way as in [Sampath et al., 1995], we define the *F*-uncertain and the *F*-indeterminate cycles.

Définition 39 *(F-uncertain cycle)*
A cycle $c\ell = a_1, a_2, \ldots, a_n$, *with* $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ *for* $1 \leq i \leq n$, *in SRD* $\mathcal{D}$, *is said to be an F-uncertain cycle, if* $\forall i : 1 \leq i \leq n : a_i$ *is an F-uncertain diagnoser node.* □

Définition 40 *(F-indeterminate cycle)*
An F-indeterminate cycle in the SRD is an F-uncertain cycle that corresponds to at least, one faulty-free cycle in LPN N_L *and at least, one faulty cycle.* □

Authors in [Sampath et al., 1995] have established a necessary and sufficient condition for diagnosability using this particular cycles. In fact, it has been established that a system model is diagnosable if and only if no F-indeterminate cycle exists in its corresponding diagnoser.

Example 14 *A part of the SRD corresponding to the LPN in Figure 6.1 (introduced in Example 3), which is constructed on the fly, is shown in Figure 6.7. The generated markings are enumerated in Table 6.2. The initial node (a_0) is composed of the initial marking m_0 of the LPN and markings m_1 and m_2 reachable from m_0 by firing unobservable transition t_1. These three markings represent the set of normal markings of node (a_0). As faulty transition t_4 can be fired from the set of normal markings then (a_0) also contains a set of faulty markings reachable by firing unobservable transitions. Node (a_1) is the successor node of (a_0) by considering the firing of observable transition t_2 (labeled by event a).*

One can observe that observable transition t_2 is enabled from both sets of normal markings $a_0.\mathcal{M}_N$ and faulty markings $a_0.\mathcal{M}_F$ of node (a_0). Therefore, according to the fault propagation rules, node (a_1) obviously contains both sets of normal and faulty markings. The set of normal markings of (a_1) contains markings reachable from the set of normal markings of (a_0) by the occurrence of observable transition t_2 and the regular transitions enabled directly after t_2. The set of faulty markings of (a_1) is composed of markings reachable from the set of faulty markings of (a_0) by the occurrence of the observable transition (t_2) and the unobservable transitions enabled after t_2. In addition to that, it also contains markings reachable from the set of normal markings in (a_1) by the firing of faulty transitions t_1 and t_6 and the unobservable transitions enabled after these faulty transitions.

Diagnoser node (a_2) is reached after the firing of transition t_7 (labeled with event b) and it contains only a set of faulty markings ($a_2.\mathcal{M}_N = \emptyset$). Thus, it is an F-certain node. As F-certain nodes are unnecessary for analyzing diagnosability, and since we deal with permanent faults, the subsequent nodes, are not constructed.

The rest of nodes are constructed on the fly using the same reasoning. Regarding the diagnosability analysis, once a cycle appears, it will be checked whether it is an F-indeterminate one or not. If it is, the constructing procedure of the SRD is stopped. We underline that the dotted lines are optionally used for guiding the construction and tracking the normal (in green) and faulty (in red) traces. However, in the actual construction of the SRD, these lines are not built. Moreover, the dashed line, representing the faulty transitions can be replaced by a Boolean variable associated with each diagnoser node (true if faulty transitions exist and $false$ otherwise).

It is worth noticing that, in this example, the nodes are enumerated (for illustration).

Actually, they are compacted and managed using BDDs as illustrated in Section 6.5.2. Thus, each node is represented using two BDDs: the first BDD encapsulates the set of normal markings and the second one corresponds to the set of faulty markings.

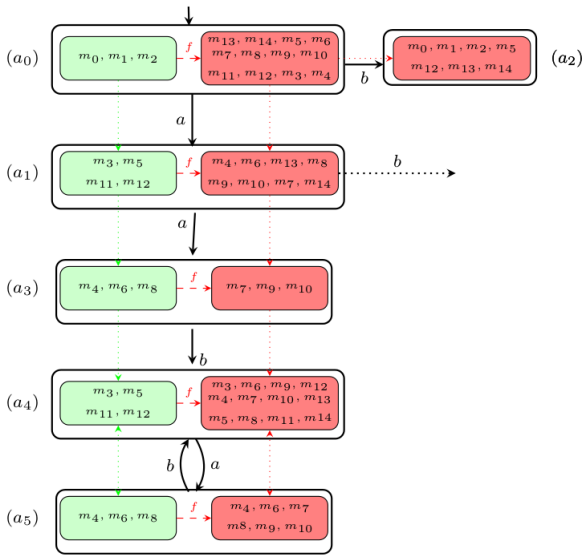


Figure 6.7 – A part of the symbolic reachability diagnoser of Example 3

In the same way, as for the existing diagnoser-based approaches, the SRD serves to both solving the diagnosability issue and performing online diagnosis. In what follows, we show how the SRD is used for performing the online diagnosis.

6.6.3 Diagnosis Using SRD

Définition 41 *The diagnosis function (corresponding to the SRD) is* $\triangle : P_o(\varphi(T^*)) \times T_f \rightarrow \{N, F, U\}$, *which associates to each observation* $\omega \in \mathcal{L}$, *w.r.t. the fault class* T_f, *a diagnosis state:*

- $\triangle(\omega, T_f) = N$ *if* $\forall s \in P_o^{-1}(\omega), \forall \sigma_f \in \Sigma_f : \sigma_f \notin s$, *which means that no fault transitions exist in all the firing sequences consistent with the observation* ω. *Thus, the reached node in the SRD, after the occurrence of* ω *is an* N*-certain node (i.e.,* $\mathcal{M}_F = \emptyset$*) ;*

- $\triangle(\omega, T_f) = F$ *if* $\forall s \in P_o^{-1}(\omega), \exists \sigma_f \in \Sigma_f : \sigma_f \in s$, *which means that at least one fault transition exists in each firing sequences consistent with the observation* ω. *Thus, the reached node in the SRD, after the occurrence of* ω *is an* F*-certain node (i.e.,* $\mathcal{M}_N = \emptyset$*);*

Table 6.2 – The reachable markings of Example 14

i	m_i (enumerative)	m_i (symbolic)
0	$[2\,0\,0\,0\,0]^\tau$	$[10\,00\,00\,00\,00]^\tau$
1	$[1\,1\,0\,0\,0]^\tau$	$[01\,01\,00\,00\,00]^\tau$
2	$[0\,2\,0\,0\,0]^\tau$	$[00\,10\,00\,00\,00]^\tau$
3	$[1\,0\,1\,0\,0]^\tau$	$[01\,00\,01\,00\,00]^\tau$
4	$[0\,0\,2\,0\,0]^\tau$	$[00\,00\,10\,00\,00]^\tau$
5	$[1\,0\,0\,1\,0]^\tau$	$[01\,00\,00\,01\,00]^\tau$
6	$[0\,0\,1\,1\,0]^\tau$	$[00\,00\,01\,01\,00]^\tau$
7	$[0\,0\,1\,0\,1]^\tau$	$[00\,00\,01\,00\,01]^\tau$
8	$[0\,0\,0\,2\,0]^\tau$	$[00\,00\,00\,10\,00]^\tau$
9	$[0\,0\,0\,1\,1]^\tau$	$[00\,00\,00\,01\,01]^\tau$
10	$[0\,0\,0\,0\,2]^\tau$	$[00\,00\,00\,00\,10]^\tau$
11	$[0\,1\,1\,0\,0]^\tau$	$[00\,01\,01\,00\,00]^\tau$
12	$[0\,1\,0\,1\,0]^\tau$	$[00\,01\,00\,01\,00]^\tau$
13	$[0\,1\,0\,0\,1]^\tau$	$[00\,01\,00\,00\,01]^\tau$
14	$[1\,0\,0\,0\,1]^\tau$	$[01\,00\,00\,00\,01]^\tau$

- $\triangle(\omega, T_f) = U$ *if* $\exists$ *(at least)* $s_1, s_2 \in P_o^{-1}(\omega)$ *s.t.* $\forall \sigma_f \in \Sigma_f : \sigma_f \notin s_1$ *and* $\exists \sigma_f \in \Sigma_f : \sigma_f \in s_2$*, which means that two firing sequences consistent with the observation* ω *exist such that one firing sequence is fault-free and the other one contains at least one faulty transition. Thus, the reached node in the SRD, after the occurrence of* ω *is an* F*-uncertain node.* □

Remark 5 *As faults are assumed to be permanent, once the obtained diagnosis state is certainly faulty* ($\triangle(\omega, T_f) = F$)*, it is unnecessary to keep computing the diagnosis states for the continuations of the observable sequence, since all the successive diagnosis states will be certainly faulty. Thus, the subsequent of* F*-certain nodes are not constructed in the SRD.*

Remark 6 *In the SRD, the initial node can be an* F*-uncertain node contrarily to the classic diagnoser approaches where the initial diagnoser node is assumed to be normal. In fact, this is due to the way we handle the unobservable reachability in the SRD construction procedure, which is computed before the current node is left (using the Reach operator, See Section 6.6.2). This feature does not affect the correctness of the diagnosis function, since a fault transition can be fired directly from the initial marking of the LPN (as shown in Example 1).*

Without considering the markings held in the node, the SRD is used for the on-line diagnosis. In fact, for each new observable event the diagnosis state is updated accordingly. Besides, for the online diagnosis, only the types of nodes are depicted as a tag in the node when the model is diagnosable, which serves to save memory and to speed up the output diagnosis state.

6.7 Diagnosability Analysis Using The SRD

Regarding the diagnosability analysis, Sampath et al. [Sampath et al., 1995] have established a necessary and sufficient condition for diagnosability on the basis of the generator and diagnoser models. In fact, it consists in checking the existence of F-indeterminate cycles in the diagnoser.

The same condition has been reformulated in [Cabasino et al., 2009b] for analyzing diagnosability of bounded labeled Petri nets. Actually, this condition is used by most compiled diagnoser-based approaches [Liu, 2014, Ushio et al., 1998, Zad et al., 2003].

Computationally, the analysis of such a condition consists in a *double-check* procedure upon the diagnoser, as discussed in Chapter 4.

Aiming to tackle this issue, we have proposed in Section 4.3.4 a reformulation of the necessary and sufficient condition for the analysis of diagnosability on the basis of the new structure of the diagnoser and we have developed a systematic approach for the actual verification such a condition without needing any intermediate model. Hereafter, we adapt these theoretical results to the SRD. The proofs are omitted as they are similar to that provided in Chapter 4.

Proposition 10 *Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$. Then, there exists at least one fault-free cycle in LPN N_L that shares the same observation $(\sigma_1, \sigma_2, \ldots, \sigma_n)^*$.* □

This result is interesting for checking *F-indeterminate* cycles, using both the classic diagnoser-based approaches and the SRD. It is, in fact, sufficient to check that an *F-uncertain* cycle in the diagnoser corresponds to a faulty cycle in the original model (or the intermediate model), without checking the existence of the faulty-free cycle (since this is plain henceforth).

Proposition 11 *Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$ with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$. Hence, if $\forall 1 \leq i \leq n : Img(a_i.\mathcal{M}_N, T_f) = \emptyset$, then $c\ell$ is an F-indeterminate cycle.* □

This result means that if in all the diagnoser nodes of an *F-uncertain* cycle no faulty transition from the normal set of markings to the faulty one exists, therefore this cycle is an *F-indeterminate* cycle.

Remark 7 *Proposition 11 can be viewed as a sufficient condition for non-diagnosability, since an LPN is non-diagnosable if the condition in Proposition 11 is satisfied by the SRD. Then, the diagnosability analysis is stopped as soon as the condition in Proposition 11 is*

met. In this case, the SRD will be constructed partially. Indeed, as will be discussed in Section 6.8, diagnosability analysis will be performed simultaneously on the fly as the SRD is set up. Such a feature would speed up the diagnosability analysis; in particular when the LPN model is non-diagnosable.

6.7.1 Necessary and Sufficient Condition for Diagnosability

In this section, we discuss a reformulation of the necessary and sufficient condition for diagnosability on the basis of the SRD. Firstly, we introduce the notion of '*indicating sequence*' associated with a given F-uncertain cycle, which will be used to establish the necessary and sufficient condition.

Définition 42 *($c\ell$-indicating sequence)*

Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$ (the starting node a_1 can be arbitrarily chosen in the cycle), with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$. $c\ell$-indicating sequence $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$, is an infinite sequence of sets of markings, such that:

- $\mathcal{S}_1 = a_1.\mathcal{M}_F$;
- $\forall\, i > 1 :\ \mathcal{S}_i = Reach_{T_u}(Img(\mathcal{S}_{i-1}, T_{\sigma_{(i-1)_{mod_n}}}))$; □

In fact, the $c\ell$-indicating sequence tracks the subsets of faulty markings in each node of $c\ell$ without considering the faulty markings generated through the occurrence of some faulty transitions outgoing from the normal set of markings in the traversed nodes (except for $\mathcal{S}_1$ which holds all the faulty states of $a_1.\mathcal{M}_F$, i.e., $\mathcal{S}_1 = a_1.\mathcal{M}_F$).

Actually, the $c\ell$-indicating sequence is introduced with the aim of tracking the actual faulty cycles corresponding to a given *F-uncertain* cycle, if such cycles exist in the original model.

Figure 6.8 depicts an F-uncertain cycle $c\ell = a_1, a_2, \ldots, a_n$ and its corresponding $c\ell$-indicating sequence $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$.

Proposition 12 *Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$, and let $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ be the $c\ell$-indicating sequence associated with $c\ell$. Therefore, the following property holds true:*

$$\forall k \in \mathbb{N}^* \ :\ \mathcal{S}_{1+nk} \subseteq \mathcal{S}_{1+n(k-1)}$$

□

The above-mentioned proposition means that, by ignoring the faulty transitions from the normal sets of states into the faulty ones in the same node, one can ensure the (non-strict) inclusion relationship, after covering the event-trace $(\sigma_1, \sigma_2, \ldots, \sigma_n)$ k times, between $\mathcal{S}_{i+nk}$ and $\mathcal{S}_{i+n(k-1)}$, for $1 \leq i \leq n$ and $\forall k \in \mathbb{N}^*$.

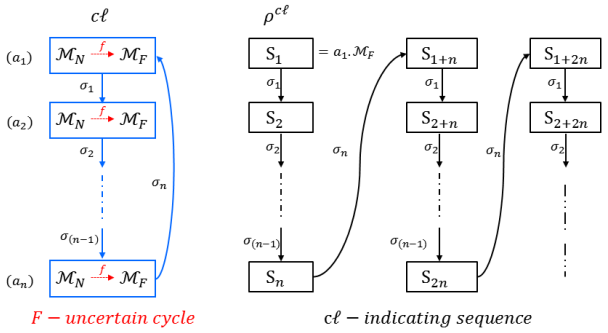


Figure 6.8 – F-uncertain cycle $c\ell$ and its $c\ell$-indicating sequence $\rho^{c\ell}$

Proposition 13 *Let $c\ell = a_1, a_2, \ldots, a_n$ be an F-uncertain cycle in $\mathcal{D}$, with $\delta_{\mathcal{D}}(a_i, \sigma_i) = a_{(i+1)mod_n}$ for $1 \leq i \leq n$, and let $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ be the $c\ell$-indicating sequence associated with $c\ell$. Therefore, the following property holds:*

$$\exists\, k \in \mathbb{N} : \forall\, i \in \mathbb{N} : \mathcal{S}_{(1+(k+i)n)} = \mathcal{S}_{1+(kn)}$$

□

Proposition 13, establishes the fact that there exists an index i from which $c\ell$-indicating sequence $\rho^{c\ell}$ shows a repetitive bloc of length n: $[\mathcal{S}_{(i+1)}, \mathcal{S}_{(i+2)}, \ldots, \mathcal{S}_{(i-1+n)}\mathcal{S}_{(i+n)}]$ with $\mathcal{S}_{(1+i+n)} = \mathcal{S}_i$ (i.e., a cycle). Phrased differently, $\rho^{c\ell}$ always takes one of these two forms:

1. *A prime sequence:* a non-cyclic elementary sequence (possibly empty) $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_i$, connected to an elementary cycle $(\mathcal{S}_{(i+1)}, \mathcal{S}_{(i+2)}, \ldots, \mathcal{S}_{(i-1+n)}, \mathcal{S}_{(i+n)})^*$, with $\mathcal{S}_{(i+1+n)} = \mathcal{S}_{i+1}$ (See Figure 6.9 (a));

2. *A finite sequence of non-empty elements followed by an infinite number of empty elements:* $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_i$ for $i \in \mathbb{N}^*$, with $\mathcal{S}_{(i+k)} = \emptyset$, $\forall k \in \mathbb{N}^*$ (See Figure 6.9 (b));

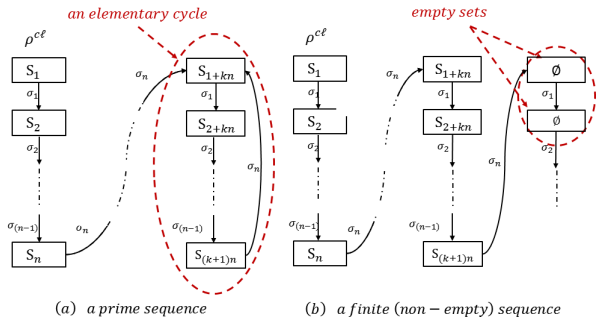


Figure 6.9 – Two possible forms of an $c\ell$-indicating sequence

In fact, the first case, i.e., a prime sequence, reveals the presence of, at least, one faulty cycle in the LPN model, that corresponds to the F-uncertain cycle $c\ell$. This is depicted in the $c\ell$-indicating sequence by the presence of the elementary cycle. However, in the second case, the F-uncertain cycle in the SRD does not correspond to an actual faulty cycle in the model. In what follows, these features, will be used to check whether an F-uncertain cycle is an F-indeterminate one or not.

Theorem 9 *For an F-uncertain cycle $c\ell = a_1, a_2, \ldots, a_n$ in $\mathcal{D}$, and $\rho^{c\ell} = \mathcal{S}_1, \mathcal{S}_2, \ldots$ its corresponding $c\ell$-indicating sequence. Then, $c\ell$ is an F-indeterminate cycle if and only if*

$$\forall i \in \mathbb{N}^* : \mathcal{S}_i \neq \emptyset$$

□

Actually, this theorem states that an F-uncertain cycle is an F-indeterminate one if the $c\ell$-indicating sequence does not reach an empty fixed-point. In other words, it takes the form of a prime sequence.

Let us take again, the SRD $\mathcal{D}$ generated from LPN N_L in Example 3. The first encountered F-uncertain cycle in $\mathcal{D}$ (since the construction is performed on the fly) is $c\ell = a_4, a_5$. The $c\ell$-indicating sequence $\rho^{c\ell}$ is depicted in Figure 6.10. According to Theorem 9, $c\ell$ is an F-indeterminate cycle since $\rho^{c\ell}$ does not reach an empty fixed-point, i.e., $\forall i \in \mathbb{N}^* : \mathcal{S}_i \neq \emptyset$. Therefore, LPN N_L, in Figure 6.1 is non-diagnosable.

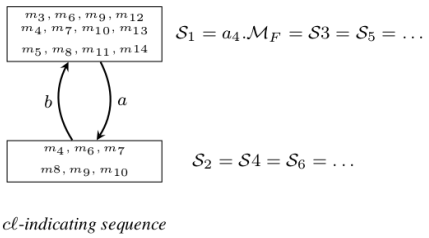


Figure 6.10 – Checking F-indeterminate cycle in Example 14 using Theorem 9

6.7.2 A Procedure for Checking Diagnosability

It is worth noticing that for the actual verification of diagnosability, *a systematic procedure*, derived directly from Theorem 9, can be performed as follows:

When an F-uncertain cycle $c\ell$ is found in SRD $\mathcal{D}$, then:

- generate the successive elements of $c\ell$-indicating sequence $\rho^{c\ell}$ (starting from $\mathcal{S}_1$), and for each element $\mathcal{S}_i$ check the following conditions:

1. if $\mathcal{S}_i = \emptyset$, then cycle $c\ell$ is not an F-indeterminate cycle and therefore the procedure is stopped;

2. else, if $\mathcal{S}_i \neq \emptyset$ and $\exists k \in \mathbb{N} : i = 1 + kn$ (with $n = |c\ell|$), then:

   (a) if $\mathcal{S}_i = \mathcal{S}_{(i-n)}$, then cycle $c\ell$ is an F-indeterminate cycle and stop the procedure;

   (b) otherwise, continue.

This procedure is repeated on each F-uncertain cycle generated (on the fly) in $\mathcal{D}$.

It is worth underlining that, on the basis of Proposition 13, one can be certain that the above procedure terminates since a fixed-point will ultimately be reached (within a finite delay).

6.8 On-the-fly Verification Algorithm

In this section, we develop an on-the-fly depth-first search (DFS) algorithm for constructing the SRD and for verifying diagnosability simultaneously. Constructing the SRD on the fly serves to avoid the systematic generation of the whole state-space of the diagnoser. That is, on one hand, one does not need to construct the part, of the diagnoser, following F-certain nodes, since such a part is unnecessary for analyzing diagnosability and performing online diagnosis (in the case where the LPN is diagnosable). On the other hand, when the LPN is non-diagnosable, we stop constructing the diagnoser as soon as an F-indeterminate cycle is found. Moreover, the proposed algorithm firstly checks the sufficient condition for non-diagnosability, proposed in Proposition 11. Thus, as soon as such a condition is met, the LPN is stated to be non-diagnosable and the verification process is stopped without checking the necessary and sufficient condition.

The following functions and data structures are used in the elaborated algorithm:

Algorithm 4 On-the-fly algorithm to construct the SRD and check diagnosability

Input: $N_L = (N, m_0, \Sigma, \varphi)$;

Output: $\mathcal{D} = \langle \Gamma, \Sigma_o, \delta_{\mathcal{D}}, \Gamma_0 \rangle$; Diagnosability verdict

Set of markings: $\mathcal{M}_n, \mathcal{M}_f, Y_{nf}, Y_f, \mathcal{M}'_n, \mathcal{M}'_f$

Set of Transitions: $\mathcal{T}_n, \mathcal{T}_f, \mathcal{T}'_n, \mathcal{T}'_f$

Boolean: $bool = True$, $tag = False$

Diagnoser node: a, a', a''

Initialization:

$\mathcal{M}_n = Reach_{T_{reg}}(m_0)$

$\mathcal{M}_f = Reach_{T_u}(Img(\mathcal{M}_n, T_f))$

$\Gamma_0.\mathcal{M}_N = \mathcal{M}_n$

$\Gamma_0.\mathcal{M}_F = \mathcal{M}_f$

$\Gamma = \{\Gamma_0\}$; $a = \Gamma_0$

$\mathcal{T}_n = Enable_{T_o}(\mathcal{M}_n)$; $\mathcal{T}_f = Enable_{T_o}(\mathcal{M}_f)$

List_Nodes = $\{a\}$; List_Labels = $\emptyset$

if *(Diagnoser_Construct (N_L, $\mathcal{D}$, $\mathcal{T}_n$, $\mathcal{T}_f$, $\mathcal{M}_n$, $\mathcal{M}_f$))* **then**

  return Non-Diagnosable

else return Diagnosable

- $IsUncertain(c\ell)$: is a function that returns a Boolean value (*true* if the encountered cycle $c\ell$ in the SRD is composed of only *F-uncertain* nodes and $false$ otherwise).

- List_Nodes, List_Nodes1, Cycle_Nodes: are three finite ordered lists of diagnoser nodes or sets of markings. These lists are used for checking the existence of cycles.

- List_Labels, Cycle_Labels: are two finite ordered lists of labels (observable events), corresponding respectively to List_Nodes, Cycle_Nodes. They are used to check the existence of cycles in the SRD.

- Add(a): an operation that adds an element a to a given ordered list.

- RemoveLast(E): an operation that removes the last added element from an ordered list E.

- Copy(i, End): is an operation that copies elements from index i to the end of a given ordered list, into a new empty ordered list.

Algorithm 5 Diagnoser_Construct() function

Input: N_L, $\mathcal{D}$, $\mathcal{T}_n$, $\mathcal{T}_f$, $\mathcal{M}_n$, $\mathcal{M}_f$

Output: Boolean value

Function *Diagnoser_Construct():*
  foreach $(T_\sigma \in \mathcal{T}_n)$ **do**
    $\mathcal{M}'_n = Reach_{T_{reg}}(Img(\mathcal{M}_n, T_\sigma))$
    $Y_{nf} = Reach_{T_u}(Img(\mathcal{M}'_n, T_f))$
    $Y_f = Reach_{T_u}(Img(\mathcal{M}_f, T_\sigma))$
    $a'.\mathcal{M}_N = \mathcal{M}'_n$
    if $(Y_{nf} \subset Y_f)$ **then**
      $\mathcal{M}'_f = Y_f$; $a'.tag = True$
      $a'.\mathcal{M}_F = \mathcal{M}'_f$
    else
      $\mathcal{M}'_f = Y_{nf} = \emptyset$
      $a'.\mathcal{M}_F = \mathcal{M}'_f$
    List_Labels.Add(σ)
    if $(\exists\, a'' \in \Gamma \mid a'' = a')$; **then**
      $a'' = \delta(a, \sigma)$
      if $(a' \in$ List_Nodes$)$ **then**
        Cycle_Nodes = List_Nodes.Copy($Index(a'), End$)
        Cycle_Labels = List_Labels.Copy($Index(a'), End$)
        if *(IsUncertain*(Cycle_Nodes)*)* **then**
          foreach *(*$a \in$ Cycle_Nodes **do**
            if *(*$a.tag = False$*)* **then** $bool = False$
          if *(*$bool = True$*)* **then return** $True$
          if *(Check_Indicating_Sequence(*G, a', Cycle_Lables*))* **then**
            return $True$
      else List_Nodes.Add(a')
    else
      $\Gamma = \Gamma \cup \{a'\}$ $a' = \delta(a, \sigma)$
      $\mathcal{T}'_n = Enable_{\Sigma_o}(\mathcal{M}'_n)$; $\mathcal{T}'_f = Enable_{\Sigma_o}(\mathcal{M}'_f)$
      Diagnoser_Construct $(\boldsymbol{G}, \boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{T}'_n}, \boldsymbol{\mathcal{T}'_f}, \boldsymbol{\mathcal{M}'_n}, \boldsymbol{\mathcal{M}'_f})$
    RemoveLast(List_Nodes)
    RemoveLast(List_Labels)
  foreach $(T_\sigma \in \mathcal{T}_f \backslash \mathcal{T}_n)$ **do**
    $\mathcal{M}'_N = \emptyset$
    $\mathcal{M}'_F = Reach_{T_u}(Img(\mathcal{M}_F, T_\sigma))$
    $a'.\mathcal{M}'_n = \mathcal{M}'_N$
    $a'.\mathcal{M}'_F = bdd\_Reduce(\mathcal{M}'_F, T_u)$
    if $(\exists\, a'' \in \Gamma \mid a'' = a')$; **then** $a'' = \delta(a, \sigma)$
    else $\Gamma = \Gamma \cup \{a'\}$, $a' = \delta(a, \sigma)$
  return $False$

The initialization step of the algorithm (cf. Algorithm 4, Lines 5-11) serves to compute the initial diagnoser node. Then, the construction of the following diagnoser nodes is performed by function DIAGNOSER_CONSTRUCT(). The construction is performed using a depth-first exploration regarding the set of enabled (observable) transitions from the current node. With the aim of using the constructed diagnoser to perform the online diagnosis while avoiding the construction of the subsequent F-certain diagnoser nodes, DIAGNOSER_CONSTRUCT() function builds only the first encountered F-certain node (if there exists) in the diagnoser path, by exploring the set of observable transitions enabled only from the set of faulty markings (i.e., $\mathcal{T}_f \backslash \mathcal{T}_n$) (Algorithm 7, Lines 32-38).

Regarding the diagnosability verification, the paths of interest are generated (and explored) from the set of observable transitions enabled from the set of normal markings of the current node (i.e., $\mathcal{T}_n$). It should be noticed that Boolean variable *tag* associated with each diagnoser node and initialized to *False* is used for checking the sufficient condition given in Proposition 11.

Algorithm 6 Check_Indicating_Sequence() function

Input: G, a', Cycle_Event, *Int* i, n
Output: Boolean value

```
1 Function CHECK_INDICATING_SEQUENCE()
2   S_1 = a'.M_F, i = 1, n = |Cycle_Event|, LIST_NODES1.ADD(S_1)
3   while (S_i ≠ ∅) do
4     S_(i+1) = Reach_{T_u}(Img(S_i, T_{σ_{(i+1)_mod_n}}))
5     if (i mod n) then
6       if ((i ≥ n)&(S_(i+1) = S_(i−n+1))) then return (True)
7       else LIST_NODES1.ADD (S_i), i++
8   return False
```

The computation of a new node a', reachable by the occurrence of an observable event σ from node a, is completed by the '*Reach*' operation upon the enabled unobservable sequences. If diagnoser node a' has already been encountered (i.e., a similar diagnoser node has already been computed) then the diagnoser is updated by only adding a new arc towards this node (cf. Algorithm 7, Lines 14, 15). In this case, function *IsUncertain*() checks if node a' belongs to an F-*uncertain* cycle (cf. Algorithm 7, line 19). If so, then the sufficient condition proposed in Proposition 11 is firstly checked by analyzing if all tags of the diagnoser nodes in this F-uncertain cycle are *True* (c.f. Algorithm 7, Lines 20-21), which means that the F-uncertain cycle is an F-indeterminate one. In this case, the function DIAGNOSER_CONSTRUCT() returns *True* and the verification process is stopped. If the sufficient condition is not satisfied, the necessary and sufficient condition

for diagnosability is checked using the proposed procedure in Section 6.7.2.

In practice, function *Check_Indicating_Sequence*() (c.f. Algorithm 6) is launched in order to compute the successive elements of $\rho^{c\ell}$ associated with a given F-uncertain cycle. If a generated non-empty element $\mathcal{S}_i$ (with $i > n$) is equal to $\mathcal{S}_{(i-n)}$, $True$ is returned, which means that the cycle is an F-indeterminate one. Thus, Algorithm 4 outputs that the LPN model is non-diagnosable and the diagnoser construction procedure is stopped. Otherwise, an empty element will ultimately be generated, and therefore $False$ is returned, which means the cycle is not F-indeterminate. The construction procedure is hence continued in a recursive manner. Once all the branches of interest are constructed and explored with no F-indeterminate cycle has been met, Algorithm 4 outputs that the LPN model is diagnosable.

6.9 Experimentation

A tool, called DPN-SOG (for **D**iagnosability analysis of **P**etri **N**ets using **S**ymbolic **O**bservation **G**raphs) [Boussif et al., 2017], implementing the proposed approach was developed in C++ programming language. DPN-SOG consists in a modified and re-implemented version of ObsGraphTool [Klai and Petrucci, 2008], which is a BDD-based tool implementing various verification approaches for workflows/PNs using symbolic observation graphs [Haddad et al., 2004].

In order to assess the effectiveness and the scalability of the SRD approach, we reuse the benchmark of Figure 4.11, which was introduced in Chapter 4 (See Section 4.6.1 for more details). The obtained results are discussed with respect to a reference approach for fault diagnosis of LPNs, called MBRG/BRD technique [Cabasino et al., 2014]. In fact, the MBRG/BRD technique is implemented as a MATLAB Toolbox called PN_DIAG tool [Pocci, 2012]. It allows generating the whole state-space of the MBRG and the BRD graphs of an LPN, which respectively are more or less equivalent to the generator and the diagnoser in [Sampath et al., 1995], but for PNs, and then performing the diagnosability analysis (for more details about MBRG/BRD technique and PN_DIAG tool, one can refer to [Cabasino et al., 2014, Cabasino et al., 2012b, Pocci, 2012]). It should be noticed that the experiments have been performed on 64-bit PC (CPU: Intel Core i5, 2.5 GHz, RAM: 6GB). We fix 5 hours as a maximum analysis duration above which we consider that the tool failed to return a result.

We evaluate the memory/time efficiency of our approach while varying the number of observable and unobservable transitions in the model. Thus, we consider the following parameters: $m = 1$, $b = 10$, and $k = 4$. Transitions t_0 and t_1 are observable while transitions $f_i \in T_f$ are faulty (for $1 \leq i \leq 4$). Regarding transitions $t_{i,10}$ (i.e., transitions in parallel with faulty transitions), two cases are considered:

Test 1: transitions $t_{i,10}$ are non-observable (for $1 \leq i \leq 4$), in this case the model is non-diagnosable;

Test 2: transitions $t_{i,10}$ are observable (for $1 \leq i \leq 4$), in this case the model is diagnosable.

Concerning the reset of transitions, we first consider them unobservable and then after each simulation, we increase the number of observable transitions in the model (by returning to observable transition in each production line of the benchmark), i.e., we increment the number of observable transitions after each simulation, from 2, 6,..., until 38 observable transitions, for Test 1 and from 6, 10,..., until 42 observable transitions for Test 2. Finally, it should be noticed that for the sake of clarity each transition symbol is considered also as its label (i.e., $\varphi(t_0) = t_0$).

6.9.1 Experimental Results

The experimental results are summarized in Table 6.3, where:

— *Obs* is the number of observable transitions in the considered model;

— $|\mathcal{D}_S|$ and $|\mathcal{D}_T|$ are, respectively, the numbers of nodes and arcs in the SRD;

— $\mathcal{D}_e$ and $\mathcal{D}_m$ are, respectively, the elapsed time for generating the SRD and analyzing diagnosability on the fly, and the memory required for building the SRD;

— $|G1_S|$, $|G1_T|$ and $G1_e$ are, respectively, the numbers of states and transitions in the $MBRG$ and the elapsed time for constructing the $MBRG$;

— $|G2_S|$, $|G2_T|$ and $G2_e$ are, respectively, the numbers of states and transitions in the BRD and the elapsed time for constructing the BRD;

— 'Diag' is the time required for giving diagnosability verdict using MBRG/ BRD approach.

One has to note that the PN_DIAG Tool takes a mathematical model of the LPN as an input, and has first to compute the whole state-space of the MBRG graph and then generate the whole state-space of BRD graph. Both graphs are then used for analyzing diagnosability by searching F-indeterminate cycles [Cabasino et al., 2014]. Our tool, DPN-SOG Tool, takes an LPN in *prod* format [Varpaaniemi et al., 1997] as an input and simultaneously has to generate the SRD state-space on the fly while analyzing diagnosability. Thus, once an F-indeterminate cycle is found, the construction of SRD and the verification process is stopped to avoid generating the whole state-space. In the

case of a diagnosable model, a partially generated state-space of the SRD is sufficient for performing the online diagnosis, as discussed earlier in the chapter.

Table 6.3 – Experimental Comparative Results for Test 1 and Test 2 Based on the LPN Benchmark

	SRD				MBRG			BRD			
Obs	$\lvert\mathcal{D}_S\rvert$	$\lvert\mathcal{D}_T\rvert$	$\mathcal{D}_e(s)$	$\mathcal{D}_m(kb)$	$\lvert G1_S\rvert$	$\lvert G1_T\rvert$	$G1_e(s)$	$\lvert G2_S\rvert$	$\lvert G2_T\rvert$	$G2_e(s)$	Diag (s)
Non-diagnosable (Test 1)											
2	4	3	$\leqslant$ 0.1	337	17	49	$\leqslant$ 0.1	4	4	$\leqslant$ 0.1	0.13
6	12	11	$\leqslant$ 0.1	699	82	233	0.5	34	68	$\leqslant$ 0.1	0.31
10	20	19	$\leqslant$ 0.1	1042	257	785	2	164	436	1	a.q.
14	28	27	$\leqslant$ 0.1	1368	626	2017	6	514	1540	10	a.q.
18	36	35	$\leqslant$ 0.1	1679	1297	4337	17	1252	4004	68	a.q.
22	44	43	$\leqslant$ 0.1	1947	2402	8249	47	2594	8644	419	a.q.
26	52	51	$\leqslant$ 0.1	2192	4097	14353	120	4804	16468	1493	-
30	60	59	$\leqslant$ 0.1	2425	6562	23345	289	8194	28676	4543	-
34	68	67	$\leqslant$ 0.1	2651	10001	36017	685	*	*	o.t.	-
38	76	75	$\leqslant$ 0.1	2845	14642	53257	1384	*	*	o.t.	-
Diagnosable (Test 2)											
6	17	34	0.2	2583	17	66	0.1	35	99	$\leqslant$ 0.1	0.48
10	82	218	0.5	8689	82	326	0.63	165	467	0.8	a.q.
14	257	770	0.6	21310	257	1026	2.28	515	1571	8	a.q.
18	626	2002	0.8	43929	626	2502	7.15	1253	4035	53.6	a.q.
22	1297	4322	1	81412	1297	9606	20.29	2595	8675	261.1	a.q.
26	2402	8234	1.1	139966	2402	9606	55.18	4805	16499	1084.5	a.q.
30	4097	14338	1.6	229487	4097	16386	155	8194	28707	3489	-
34	6562	23330	1.8	343322	6562	26246	371	13125	46691	9991	-
38	10001	36002	2.5	503601	10001	40002	721	*	*	o.t.	-
42	14642	53242	3.7	710052	14642	58566	1624	*	*	o.t.	-

*: No result obtained in 5 hours. o.t.: Out of time (simulation time $\geq$ 4 hours). a.q.: '*accident quit*'

6.9.2 Discussion

In this section, we highlight the main observations that can be derived from the obtained results. Firstly, by considering $m = 1$, $k = 4$ and $b = 10$, the LPN benchmark has 45 places, 46 transitions, while its reachability graph contains 14642 markings and 58566 transitions. Secondly, it should be noticed that the LPN benchmark is non-diagnosable for Test 1 whereas it is diagnosable for Test 2. In what follows, the obtained results are discussed according to two features:

6.9.2.1 Memory/Time Efficiency of the SRD approach

— In the case of the non-diagnosable models (i.e., Test 1), one can observe that the SRD approach efficiently analyze the diagnosability by only constructing the relevant part of the diagnoser (as regards of diagnosability analysis), which reduces the memory/time consumption. Actually, as the construction of the diagnoser and verification of diagnosability are simultaneously performed on the fly. As soon as an F-indeterminate cycle is found, the construction/verification process is stopped and the model is stated to be non-diagnosable (without necessarily building the whole state-space of the diagnoser). Consequently, the diagnosability verdicts are given in less than one second even for large values of Obs.

— In the case of diagnosable models (i.e., Test 2), the SRD approach potentially needs to construct a larger part of the diagnoser state-space. Consequently, the verification process checks all the F-uncertain cycles that exist in the diagnoser. Thanks to the systematic procedure for checking diagosability (established in Section 6.7.2), the verification time is not too much affected (it remains in the order of seconds for large number of observable events, i.e., Obs).

— The efficiency of the symbolic representation can be clearly shown in the case of diagnosable models (i.e., Test 2) since a larger part of the SRD state-space is potentially generated (contrarily to the case of non-diagnosable models where only a small part of the SRD is generated). One can observe that for the same reachability state-space (i.e., 14642 markings and 58566 transitions), the required memory to build the SRD increases proportionally with the number of observable transitions. In fact, the symbolic representation leads to an important memory saving when the model contains a large number of unobservable transitions. That is, when the SRD nodes contain a large number of markings (which corresponds to the case of a large number of unobservable transitions in the model), then the corresponding BDDs will be efficiently compacted. This is due to the fact that BDDs are particularly convenient

to represent large sets of markings. However, when The SRD nodes contain a few number of markings, it is more difficult to compact them using BDDs. Thus, this explains the considerable memory consumption when the model contains a large number of observable transitions. In fact, when the number of observable transitions increases in the model, the SRD converges to the classic diagnosers [Sampath et al., 1995, Cabasino et al., 2009b] in terms of memory required for the diagnoser construction, which decreases the efficiency of the symbolic representation.

6.9.2.2 A comparative Analysis with the MBRG/BRD approach

— It is worth recalling that for MBRG/BRD approach, the PN_DIAG tool firstly needs to generate the MBRG graph and thereafter the BRD graph to finally analyze diagnosability.

— In the case of non-diagnosable models (i.e., Test 1), our approach is largely more efficient compared to the MBRG/BRD approach. This is a logical result since our approach is based on the on-the-fly technique for generating the diagnoser and analyzing diagnosablity (as explained above).

— Our approach remains more efficient when there is a large number of unobservable transitions compared to the MBRG/BRD approach in the case of diagnosable models (Test 2). This may be explained through three points:

1. Our approach only constructs one graph since diagnosability analysis is performed directly on the diagnoser.
2. The systematic procedure for checking F-indeterminate cycles allows reducing the verification time compared to the MBRG/BRD (since it suffers from the double-check issue discussed earlier in this chapter).
3. Our approach generates only the necessary part of the SRD for analyzing diagnosability and performing online diagnosis contrarily to the MBRG/BRD approach, where the whole state-spaces of the MBRG and BRD graphs are generated to analyze diangosbaility.

— As shown in Table 6.3, when the model contains more than 30 observable transitions, PN_Diag tool spends more than 5 hours without generating the BRD and thus without deciding the diagnosability. However, our tool generates the SRD and analyzes diagnosability in few seconds.

— Regarding the diagnosability analysis using PN_DIAG tool, some *accident quits* occur during its running (i.e., an exit without any output results). This may be

caused by eventual bugs in the tool. Thus, we do not compare the elapsed time for checking diagnosability.

6.10 Conclusion

In this chapter, a diagnoser-based approach to deal with fault diagnosis of bounded and labeled Petri net models is presented. The approach consists in building a partial symbolic diagnoser called *'Symbolic Reachability Diagnoser'* with a new structure that consists in separating normal markings from faulty ones in each diagnoser node. In fact, the sets of normal and faulty markings, in each diagnoser node, are encoded using a symbolic representation, i.e. Binary Decision Diagrams. Moreover, a necessary and sufficient condition for diagnosability analysis of Petri nets is derived from the diagnoser structure, and a systematic procedure for checking such a condition without building any intermediate model is established. Finally, an on-the-fly algorithm to construct the diagnoser and check the diagnosability simultaneously is provided. Finally, some experimental results that show the effectiveness of the proposed approach are discussed.

In the second part of the manuscript, we address various DES diagnosis issues on the basis of the twin-plant structure. Indeed, as mentioned earlier in the dissertation, the twin-plant based technique shows some substantial advantages in terms of complexity compared to the diagnoser-based technique, when it comes to investigate diagnosability. The developed contributions are related to both permanent and intermittent faults while considering different aspects.

Part III

CONTRIBUTIONS REGARDING THE TWIN-PLANT BASED APPROACH

Chapter 7

Practical Verification of Diagnosability in a Model-Checking Framework

Sommaire

Summary

This chapter discusses the practical verification of permanent fault diagnosability in a model-checking framework. In fact, the diagnosability issue is reformulated as a model-checking problem and then tackled using model-checking techniques. The diagnosability condition is expressed using CTL formula while the twin-plant structure is transformed into a Kripke structure and, therefore, diagnosability is investigated as a model-checking problem. Moreover, we reformulate the K-diagnosability issue in model-checking framework and we discuss the problem of K_{min}-diagnosability (the minimal value of K ensuring diagnosability). Finally, some illustrations are provided through a benchmark.

The work presented in this chapter is the subject of publications in DCDS'15 [Boussif and Ghazel, 2015a] *and MSR'15* [Boussif and Ghazel, 2015b].

This chapter is organized as follows: after a short introduction regarding the practical verification of diagnosability, we briefly recall the main concepts regarding model-checking in Section 7.2 and we give an overview of fault diagnosis using model-checking techniques in Section 7.3. In Section 7.4, we discuss some modeling issues in view of using model-checking techniques. Section 7.5 first recalls Cimatti's definition of diagnosability [Cimatti et al., 2003], and then introduces an extension of this definition in order to comply with the classic one [Sampath et al., 1995]. The reformulation of diagnosability as a model-checking problem is discussed in Section 7.6. In Section 7.7, we introduce K-diagnosability as a CTL model-checking problem and we discuss the K_{min}-diagnosability issue. In Section 7.9, we illustrate the concepts discussed through a level crossing benchmark. Finally, Section 7.10 draws various concluding remarks.

7.1 Introduction

Diagnosability of DESs [Lin, 1994, Sampath et al., 1995, Sampath et al., 1996] has become an interesting research topic in both artificial intelligence and control theory communities. A large and a significant amount of works have been devoted the to the analysis of such a property in centralized models [Bourgne et al., 2009, Morvan and Pinchinat, 2009, Hosseini et al., 2013, Schumann and Pencolé, 2007, Zad et al., 2003, Liu, 2014, Grastien, 2009, Jiang and Huang, 2001, Yoo and Garcia, 2004, Yoo and Lafortune, 2002b], decentralized models [Philippot et al., 2013, Debouk et al., 1998, Moreira et al., 2011, Wang et al., 2007, Sayed-Mouchaweh and Carre-Menetrier, 2008, Lafortune et al., 2005, Provan, 2002, Cabasino et al., 2013a, Qiu and Kumar, 2006, Wang et al., 2004, Qiu et al., 2009, Schmidt, 2010], distributed and modular models [Ye and Dague, 2013, Contant et al., 2006, Debouk et al., 2002b, Zhou et al., 2008, Schmidt, 2013], Petri net framework [Cabasino et al., 2014, Liu, 2014, Li et al., 2015c, Cabasino et al., 2009b, Cabasino et al., 2012a, Basile et al., 2012a], etc.

Unfortunately, most of the approaches developed are oriented to the definition of a theoretical frameworks, and do not address the problems related to the practical application of the approaches developed. Moreover, such approaches propose ad-hoc algorithms for the actual verification of diagnosability which are implemented in academic tools. Hence, the practical implementation of the developed DESs diagnoser techniques is an issue that still needs exploration.

The works in [Cimatti et al., 2003, Pecheur et al., 2002], which are rather different from the works mentioned above, propose a practical framework for the formal verification of the diagnosability using model-checking techniques. In fact, authors attempted to bring forward an effective platform for the analysis of diagnosability that can be practically

applied in the development process of diagnosis systems. The main advantage of the proposed approach is that the actual verification is performed using a model-checking tool. Fortunately, a wide range of powerful and optimized model-checkers have been developed in the formal verification community and successfully used for the verification/validation of large scale industrial systems. Such tools can be used for the verification of diagnosability using the practical framework proposed in [Cimatti et al., 2003, Pecheur et al., 2002].

The work we propose in this chapter is based on this practical approach [Cimatti et al., 2003], where some improvements and extensions to deal with various fault diagnosis problems are discussed.

7.2 Model-Checking

Techniques for automatic formal verification of finite state transition concurrent systems have been developed in the last three decades to the point where major chip design companies are beginning to integrate them into their normal quality assurance process. The most widely used of these methods is called Model-Checking [Biere et al., 2003]. Model-checking is originated from the independent work of two pairs in the early eighties: Clarke and Emerson [Clarke and Emerson, 1981] and Queille and Sifakis [Queille and Sifakis, 1982]. Model-checking is an automatic formal verification technique that is widely applied to the design of complex dynamic systems (communication protocols, hardware design, software design, etc.). It allows for verifying whether the behavior of a system (modeled by a Kripke structure) satisfies a given property (expressed in a property specification language) or not using efficient algorithms based on an exhaustive exploration of the system behavior. A counter-example is generated if the system does not satisfy the property, which is an interesting feature namely for debugging [Clarke et al., 1999, Baier et al., 2008] .

The model-checking problem can be defined as follows:

Définition 43 *(Model-Checking Problem)*

Let M be a design (abstraction as a transition system) and let p be a specification (expressed in a temporal logic) to be checking. The model checking problem is to find all the states s (of the system model) such that: $M, s \models p$.

When applying model checking to a design, the following different phases can be distinguished [Baier et al., 2008]:

- **Modeling phase:**

1. model the system under consideration using the model description language of the model checker at hand;
2. formalize the property to be checked using the property specification language.

- **Running phase:** run the model checker to check the validity of the property in the system model;
- **Analysis phase:**
  1. property satisfied? → check next property (if any);
  2. property violated? →
     (a) analyze generated counterexample by simulation;
     (b) refine the model, design, or property; repeat the entire procedure.
  3. out of memory? → try to reduce the model and try again.

The prerequisite inputs to model-checking are (1) **a model of the system** (a transition system) under consideration and (2) **a formal specification** to be checked.

7.2.1 The System Modeling

Generally, the systems under verification are modeled by Kripke Structure (for state-based models) or Labeled Transition Systems (for event/transition-based model). Both models are interchangeable in many contexts [Reniers and Willemse, 2011, De Nicola and Vaandrager, 1995, De Nicola and Vaandrager, 1990]. However, the Kripke structure remains the standard representation of models in the model-checking literature.

Définition 44 *(Kripke Structure (KS))*
A Kripke Structure is a tuple $\langle S, AP, \rightarrow, L \rangle$*, where*

— S *is the set of states;*

— AP *is a set of atomic propositions;*

— $\rightarrow \subseteq S \times S$ *is a total transition relation, i.e.* $\forall s \in S, \exists t \in S : (s,t) \in \rightarrow$*;*

— $L : S \rightarrow 2^{AP}$ *is the state labeling function ;*

The notion of a Kripke structure is only a vehicle for illustrating the algorithms. It captures the semantics of the system under investigation. For a concrete design language, the process of extracting a Kripke structure from a given representation may not be that easy. In particular, the size of the system description and the size of the state-space can be very different [Biere et al., 2003].

7.2.2 The Specification Modeling

To make a rigorous verification possible, properties should be described in a precise and unambiguous manner. This is typically done using a property specification language. The specification languages most used are temporal logics. Temporal logics are basically an extension of traditional propositional logics with operators that refer to the behavior of systems over time. It allows for the specification of a broad range of relevant system properties such as functional correctness (*does the system do what it is supposed to do?*), reachability (*is it possible to end up in a deadlock state?*), safety (*something bad never happens*), liveness (*something good will eventually happen*), fairness (*does, under certain conditions, an event occur repeatedly?*), and real-time properties (*is the system acting in time?*) [Baier et al., 2008].

7.2.2.1 Linear Temporal Logic (LTL)

Pnueli [Pnueli, 1977] has introduced the linear temporal logic for the specification and verification of the designs. LTL reasons over linear traces of the Kripke structure through time. At each time instant, there is only one real future timeline that is considered. Conventionally, that timeline is defined as starting '*now*', in the current time step, and progressing infinitely in the future. LTL formulas are composed of finite set of atomic propositions, Boolean connectives $\neg, \wedge, \vee$, and the temporal connectives: 'X' that means '*neXt*', 'G: *Globally*','R: *Release*', 'F : *in the Future*','U : *Until*'

Définition 45 *(Syntax of LTL)*
LTL formulas over the set of atomic proposition (AP) are formed according to the following grammar:

$$\phi ::= true \mid a \mid \phi_1 \wedge \phi_2 \mid X\phi \mid \phi_1 \cup \phi_2$$

where $a \in AP$.

7.2.2.2 Computation Tree Logic (CTL)

CTL is a temporal logic based on propositional logic with a discrete notion of time, and only future modalities. CTL is an important branching temporal logic that is sufficiently expressive for the formulation of an important set of system properties. It was originally used by Clarke and Emerson [Clarke and Emerson, 1981] and (in a slightly different form) by Queille and Sifakis [Queille and Sifakis, 1982].

CTL has a two-stage syntax where formulas in CTL are classified into state and path formulas. The former are assertions about the atomic propositions in the states and their branching structure, while path formulas express temporal properties of paths.

Définition 46 *(Syntax of CTL)*
LTL state formulas over the set of atomic proposition (AP) are formed according to the following grammar:

$$\psi ::= true \mid a \mid \psi_1 \wedge \psi_2 \mid \neg\psi \mid \exists\phi \mid \forall\phi$$

where $a \in AP$ and ϕ is a path formula. CTL path formulas are formed according to the following grammar:

$$\phi ::= X\psi \mid \psi_1 \cup \psi_2$$

where ψ, ψ_1, ψ_2 are states formulas.

7.2.3 Progress in Model-Checking

The main technical challenge in model-checking is the state explosion problem, since the size of the global transition system can be (at least) exponential in the size of design.

Considerable progress has been made to partially overcome this problem: methods based on abstraction, counterexample-guided abstraction, symbolic representation, and compositional reasoning are used [Emerson, 2008, Clarke et al., 2001]. Indeed, the symbolic model-checking (SMC) [McMillan, 1993, Clarke et al., 1996, Biere et al., 1999b], which is originated from the CTL and Fixpoint based model-checking algorithm uses binary decision diagrams (BDDs) to handle the sets of states and transitions. Currently, with the SMC techniques, it is possible to verify designs modeled with 100 to 300 state variables and having about 10^{30} to 10^{100} or more global states. In the last decade, SAT-based bounded model-Checking (BMC) [McMillan, 2003, Biere et al., 1999a] has been put forward as an alternative approach. In fact, the SAT-based approaches [Prasad et al., 2005] can accommodate larger designs than the BDD-based approaches. However, it only explores for '*close*' errors at depth bounded by k where typically k ranges from a few tens to hundreds of steps. In general it cannot find 'deep' errors and provide *verification of correctness* [Emerson, 2008].

7.3 A Review on Fault Diagnosis Using Model-Checking

Due to their expressiveness, temporal logics have been used for for a long time in supervisory control of DESs. For instance, [Thistle and Wonham, 1986, Lin, 1991, Ling and Ionescu, 1994, Deshpande and Varaiya, 1996] uses linear-time temporal logic (LTL); [Ostroff and Wonham, 1990, Ostroff, 1989] uses real-time temporal logic (RTTL), [Antoniotti, 1995] uses computation tree logic (CTL). Regarding the fault diagnosis of DESs, the temporal logics provide a general specification for the notion of failures. Thus, in addition to expressing the fault event occurrences and the reachability of faulty states, temporal

logics can express more complex kinds of failures such as: a certain set of states should be visited infinitely often, or a certain set of states should eventually be invariant, or other *invariance*, *recurrence*, and *stability* properties.

The first work dealing with fault diagnosis of DESs using temporal logic was made by Darwiche et al. [Darwiche and Provan, 1996], where a modeling formalism was proposed. In [Cordier and Largouet, 2001], the authors have discussed how model-checking techniques can be exploited for the diagnosis task by computing trajectories explaining observations generated by the system based on automata behavioral models. In the last fifteen years, various approaches and formalism based on the formal specification and model-checking frameworks have been proposed. In what follows, we give an overview of the relevant approaches proposed.

7.3.1 Fault Diagnosis with LTL-based Specifications

Fault diagnosis of DES using LTL specifications was the subject of S. Jiang's PhD thesis [Jiang, 2002]. In this work, the LTL formulas are used for specifying failures in the system. That is, an infinite state-trace of the system is said to be faulty if it violates the given LTL formula. Fault diagnosis in this context can be viewed as a generalization of the language/automata-based fault diagnosis [Sampath et al., 1995, Zad et al., 2003], since it can capture the failure representing the violation of both liveness and safety properties, whereas the prior formal language/automaton-based framework can capture the failures representing the violation of only the safety properties (i.e., the occurrence of faulty events or the reachability of faulty states) [Jiang and Kumar, 2004].

In such a work [Jiang, 2002], diagnosability of DESs is defined in the temporal logic setting and the verification problem is reduced to that of LTL model-checking. Algorithms for performing such a verification of diagnosability and synthesis of a diagnoser are provided. The complexity of the algorithm, for the diagnosability analysis, is exponential in the length of each LTL formula, and polynomial in the number of system states and the number of specifications. The problem of the failure diagnosis of intermittent/repeated faults [Jiang et al., 2003b] has also been studied and the corresponding algorithms were given in [Jiang and Kumar, 2006].

7.3.2 Rules-based Model Using First-order LTL

The rules-based models [Chandra and Kumar, 2002] are specific assignment program models [Kumar et al., 1993], which can be used for representing DESs. In such a formalism, state-variables and rules for modifying their values are used to compactly model a DES. The representation of a system with faults in the rules-based modeling formalism is poly-

nomial in the number of signals and faults (which are assumed to be binary valued). In the rules-based modeling formalism, states correspond to values of certain variables which start from certain initial values and evolve as the events occur. The rule-based models are known to be compact. Thus, developing techniques for failure diagnosis that are able to exploit the compactness of such models is an interesting task.

In this regard, authors in [Huang et al., 2004, Huang, 2003] have developed symbolic techniques based on first-order temporal logic model-checking for verifying diagnosability. Moreover, an on-line algorithm for diagnoser synthesis is obtained by using predicates and predicate transformers. First-order linear temporal logic (FOLTL) [Emerson, 1990, Hughes and Cresswell, 1996] is obtained by taking Propositional linear temporal logic (PLTL) and endowing it with a first-order logic. That is, in addition to atomic propositions, Boolean connectives, and temporal operators, it is endowed with variables, functions, and predicates, each interpreted over appropriate domains, and existential and universal quantifiers. The diagnosability analysis in the rules-based model, is a generalization of the diagnosability analysis in the automaton setting presented in work [Jiang and Huang, 2001].

7.3.3 Fault Diagnosis via Temporal Epistemic Logic

Temporal Epistemic logic (TEL) is an extension of the Epistemic logic (which is used to describe and reason about knowledge of agents and processes) by considering the LTL with past operators. Such an extension is called KL [Halpern and Vardi, 1989] (or LTLK [Męski et al., 2012]).

The first contribution on the basis of TEL was proposed in [Cimatti et al., 2005] where a reformulation of diagnosability as epistemic properties is introduced. In this work, the DES model is represented as a deontic interpreted system [Gammie and Van Der Meyden, 2004], in which the diagnoser is a particular agent that stores in its local states the outputs and the commands. The particularity of such an approach, regarding the diagnosability analysis using (a temporal-only) model-checking, is the fact that it does not reason about pair of traces but it considers sets of (observationally) equivalent traces.

In a series of works [Bozzano et al., 2014, Gario, 2016, Bozzano et al., 2013b, Bozzano et al., 2013a], Bozzano et al. have proposed a formal approach to the design of FDI (Fault Detection and Identification) components for DESs and a logical language for the specification of FDI requirements. More precisely, they have reformulated the diagnosability property and the maximality of the diagnoser, that is, the ability of the diagnoser to raise an alarm, as soon as and whenever, possible, as epistemic properties. Such properties are then checked via epistemic model-checking [Gammie and Van Der Meyden, 2004] using

the semantics of the alarm conditions [Bozzano et al., 2014]. In addition, the automated synthesis of a diagnoser according to several TEL specifications is discussed.

7.3.4 Fault Diagnosis via Satisfiability

The Boolean Satisfiability Problem (SAT) [Cook, 1971] is a dual technique of model-checking, which is considered as an efficient approach to solving path finding problems such as AI planning. It consists in reformulating a problem as a conjunction of Boolean formula and then determining an interpretation (model) that satisfies this formula.

In [Grastien et al., 2007], authors have proposed a SAT framework for modeling and solving fault diagnosis problems. In this work the diagnosis problem is translated into the propositional satisfiability problem (SAT) and then solved by the state-of-the-art SAT algorithms. Some experiments have been conducted to demonstrate the applicability of the proposed framework [Grastien and Anbulagan, 2013, Grastien and Anbulagan, 2010].

The incremental diagnosis problem consists in computing the diagnosis of only a temporal window, and then updating the diagnosis by extending the considered temporal window. Such a problem has been tackled by satisfiability techniques in [Grastien et al., 2008] and solverd using MiniSAT solvers [Eén and Sörensson, 2003].

Regarding the diagnosability analysis, the authors in [Rintanen et al., 2007b] consider the fact the diagnosability can be viewed as finding paths problem, similarly to AI planning or LTL model-checking, which can be reduced to the satisfiability problem of the classical propositional logic [Kautz and Selman, 1996, Biere et al., 1999a]. The proposed approach is based on a reformulation of the twin-plant technique [Jiang and Huang, 2001], by constructing a formula for which the satifiable valuations (of state variables) corresponds to pairs of states that depict the concept of ambiguous traces [Jiang and Huang, 2001, Yoo and Lafortune, 2002b]. Then, the formula is satisfiable if and only if it is not possible to detect the occurrence of a fault event (for a given event sequence length).

7.3.5 Fault Diagnosis as a Practical Model-Checking Problem

Cimatti et al. [Cimatti et al., 2003] have reformulated the twin-plant technique [Jiang and Huang, 2001] as a symbolic model-checking problem, which can be tackled directely using state-of-the-art symbolic model-checkers. In particular, this definition is based on the notion of critical pairs, which refer to ambiguous behavior in the system. Checking diagnosability is reduced to a reachability analysis problem in the coupled twin plant. In this work, diagnosability is expressed as a CTL/LTL formula and then tackled using a model-checking engine to check whether the model satisfy such a formula. A counter-example is given when the formula is violated. The approach has been tested in a real

case-study [Bajwa et al., 2003] and shown promising results. The particularity of such an approach is that it can be practically implemented, applied and reused for various kinds of real-life systems. Various approaches have been proposed later with the same spirit: [Bourgne et al., 2009] (for input/output symbolic transition systems), [Cassez and Tripakis, 2008] (reducing the diagnosability problem to the Buchi emptiness problem), and [Philippot et al., 2013] (for a decentralized framework). Recently, [Peres and Ghazel, 2014] proposed a generic operative formulation for diagnosability using μ-calculus logic. The main contribution of this approach is that it deals with the diagnosability issue in a single homogeneous logical framework while using various μ-calculus formula.

The work we propose in this chapter is based on this practical approach [Cimatti et al., 2003], where some improvements and extensions to deal with various fault diagnosis problems are discussed.

7.4 State-Based Modeling of DESs

The formal verification of diagnosability via model-checking is more suitable for the *state-based diagnosis* [Lin, 1994, Zad et al., 2003], i.e., where diagnosability consists in detecting if the model reaches bad (faulty) states. Contrarily to Sampath's context [Sampath et al., 1995], in which diagnosability consists of the detection of the fault event occurrences, i.e., an *event-based diagnosis*. It is possible to transform an event-based FSA into a state-based one. Such a transformation can be performed by means of a label automaton, i.e., a supervision pattern [Jéron et al., 2006] $\Omega = \langle \{N, F\}, \Sigma, \delta_\Omega, N \rangle$, which depicts the dynamic behavior of the system regarding the permanent faults, as illustrated in Figure 7.1. Therefore, the state-based FSA is obtained by the parallel composition of the event-based model G and the label automaton Ω. One can note that automaton Ω plays the role of the labeling function, which is usually used in fault diagnosis [Sampath et al., 1995].

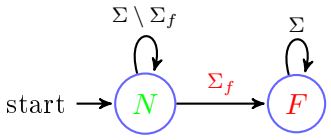


Figure 7.1 – The label automaton Ω

In this chapter, we consider that we have the state-based model $\mathcal{Q}$ of the system to be diagnosed, obtained by a parallel composition of the event-based model G (as defined in Section 3.2) with label automaton Ω. Therefore, $\mathcal{Q} = G \parallel \Omega = \langle X, \Sigma, \delta, x_0 \rangle$, with $X = X_n \biguplus X_f$ such that X_n denotes the set of normal states and X_f the faulty ones. The

notations introduced inSection 3.2 for event-based FSA G remains valid for state-based FSA $\mathcal{Q}$.

7.5 Diagnosability Analysis

In this section, we review the main definitions and results on the diagnosability analysis using model-checking introduced by Cimatti et al. in various works [Cimatti et al., 2003, Bertoli et al., 2007]. Then, we introduce an extended definition of diagnosability and we discuss its formulation as a CTL model-checking problem.

7.5.1 Cimatti's Diagnosability Definition

In [Cimatti et al., 2003] diagnosis is introduced as a function which associates each observable event sequence with a set of conditions; each condition represents a set of of states which satisfies the same properties. Using such a condition offers a quite generic framework to deal with various fault specification forms (event/state faults, intermittent faults, or complex ones). Hereafter, we give the definition of diagnosis condition.

Définition 47 *(Diagnosis condition [Cimatti et al., 2003])*
A diagnosis condition for an FSA $\mathcal{Q}$ is a pair of non-empty sets of states c_1 and $c_2 \subseteq X$, with $c_1 \cap c_2 = \emptyset$, denoted by $c_1 \perp c_2$.

One can note that the diagnosis condition concept offers a generic frame to deal with various issues. For instance, one can express,

— **Fault detection**, i.e. deciding whether any fault has occurred: $fault_i \perp \neg fault_i$.

— **Fault separation**, i.e. distinguishing between different faults (or fault classes): $fault_i \perp fault_j$.

For the sake of simplicity and without loss of generality, we will assume the existence of one single class of faults. Therefore, only *the fault detection* issue will be dealt with. Then, if we consider several fault classes, it suffices to prove that the system is diagnosable for each kind of fault taken separately. According to [Cimatti et al., 2003], we can verify the diagnosability of a diagnosis condition by checking that the plant does not have a pair of finite executions with identical observation, where the final state of one execution belongs to set c_1, whereas the final state of the other execution belongs to set c_2 . We call such a pair of executions a critical pair for the diagnosis condition $c_1 \perp c_2$.

Définition 48 *(Critical pair)*

A critical pair in FSA $\mathcal{Q}$ relative to a diagnosis condition $c_1 \perp c_2$, is a pair of feasible executions s_1 and s_2, both of length t, with identical observation $P(s_1) = P(s_2)$, such that $\exists t \in \mathbb{N} : (c_1(x^t_{s_1}) \wedge c_2(x^t_{s_2}))$ holds. □

Here $c_i(x)$ denotes that state $x \in c_i$ and state $x^t_{s_i}$ denotes the reached state upon the t^{th} observable event in execution s_i, as illustrated in Fig. 7.2. It is worthwhile noticing that, if we consider that c_1 refers to normal states and c_2 to faulty states, the above definition is equivalent to the ambiguity notion in the sense of [Sampath et al., 1995].

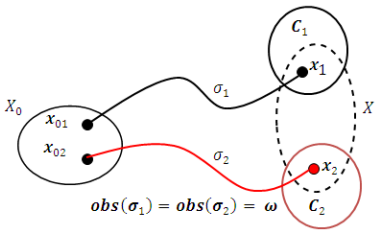


Figure 7.2 – Critical pair

Theorem 1 *(Necessary and Sufficient Condition of Diagnosability [Cimatti et al., 2003])*

Considering c_1 a set of normal states and c_2 a set of faulty states, a diagnosis condition $c_1 \perp c_2$ is said to be diagnosable in FSA $\mathcal{Q}$ if and only if $\mathcal{Q}$ has no critical pair.

According to this theorem, diagnosability consists in verifying if two executions exist, both of length t and with identical observation in which the first execution leads to a state in set c_1 (normal states) and the second one leads to a state in set c_2 (faulty states). The existence of such a pair of executions implies that the system is not diagnosable as it corresponds to a critical pair.

7.5.2 Extension of Cimatti's Diagnosability Definition

Diagnosability as defined here before is very important for safety-critical systems in which, knowing the accurate estimation state of system instantaneously after each observation and identifying whether it behaves in normal or faulty behavior, is considered as crucial task.

One can note that Cimatti's definition of diagnosability can be very useful when dealing with safety-critical systems, since it considers that a system is not diagnosable as soon as some ambiguity on the state estimation may occur, i.e. one does not consider the possible

continuations on the system behavior to decide about diagnosability. Nevertheless, in the original context of [Sampath et al., 1995], the diagnosability property is fulfilled if ambiguity disappears within a finite delay. Therefore, taking into account the classic definition of diagnosability [Sampath et al., 1995], such a definition can be considered as simplified version of diagnosability with a strong condition, since it does not take into account the evolution of the system state after the last observation t, i.e. it does not take into account the continuations of executions. More precisely and from a quantitative viewpoint, such a definition is equivalent to $Zero-$diagnosablity, which means '*the immediate diagnosability with no delay*'.

With the aim of approaching the classic definition of diagnosability [Sampath et al., 1995], which takes into account the evolution of the state estimation after the first occurrence of the fault, i. e., taking the infinite continuation of executions into consideration, before checking whether a system is diagnosable on not, this definition of diagnosability [Cimatti et al., 2003] has to be extended by introducing a novel notion called *infinite critical pairs*. Using this extended definition, it becomes possible to take into consideration, not only the actual state estimation, but also its evolution after the first occurrence of the fault, and so to give a more accurate decision about diagnosability.

Définition 49 *Given an FSA $\mathcal{Q}$, c_1 and c_2 are two sets of states and $c_1 \perp c_2$ is a diagnosis condition. Two infinite feasible executions s_1 and s_2 are said to form an infinite critical pair for diagnosis condition $c_1 \perp c_2$, if and only if,*

$$s_1, s_2 \in \Sigma^* \text{ such that } P(s_1) = P(s_2) \wedge \exists\, t \in \mathbb{N} \mid \forall i \geq t : c_1(x^i_{s_1}) \wedge c_2(x^i_{s_2})$$

The above definition means that given two infinite feasible executions s_1 and s_2 with identical observation, s_1 and s_2 form an infinite critical pair if, from a given step t, s_1 leads to, and stays in, states in condition c_1 infinitely, and s_2 leads to, and stays in, states in condition c_2 infinitely, as illustrated in Fig. 7.3.

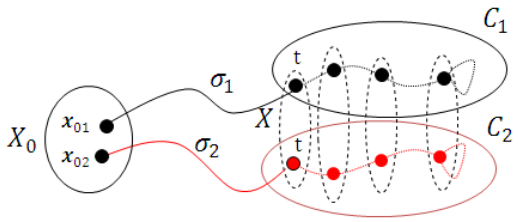


Figure 7.3 – Infinite critical pair

Recall here that, without loss of generality, we consider that one condition represents the faulty states and the other the normal states. Then, for any execution that leads to,

and stays in, states in this condition we can infer that a permanent fault has occurred. Furthermore, it is clear that the definition above corresponds exactly to the definition of indistinguishable infinite event-sequence in the original definition of diagnosability [Sampath et al., 1995].

According to this definition, we can state that the absence of infinite critical pairs for a diagnosis condition $c_1 \perp c_2$ over Σ^* is a necessary and sufficient condition for $c_1 \perp c_2$ to be diagnosable.

Définition 50 *A diagnosis condition $c_1 \perp c_2$ is said to be diagnosable over Σ in $\mathcal{Q}$, if and only if $\mathcal{Q}$ has no infinite critical pairs for $c_1 \perp c_2$ in Σ^*.*

7.6 Diagnosability as a Model-Checking Problem

In order to use model-checking for verifying diagnosability of permanent faults, [Cimatti et al., 2003] proposed a method to formulate the diagnosability issue as a model-checking problem using a temporal logic formulas. Using such a formalism, diagnosability investigation on FSA $\mathcal{Q}$ is reduced to the problem of searching critical pairs, which can be done by means of twin-plant construction [Jiang and Huang, 2001]. Actually, the search for the critical pairs, i.e., the problem of diagnosability, can be reformulated as a reachability problem on the twin-plant. Indeed, the twin-plant structure is reformulated as a Kripke structure and the necessary and sufficient condition for diagnosability, which can be viewed as a safety property, is expressed using a temporal logic formula. In what follows, we discuss such a reformulation.

7.6.1 The Twin-Plant as a Kripke Structure

As presented in Chapter 3, Section 3.5.2. The twin-plant $\mathcal{G}$ simply consists of two synchronized copies of the system model, i.e. the transitions are synchronized on the (observable) transitions so that any path in the twin-plant corresponds to a pair of executions in the system model $\mathcal{Q}$ that share the same observation.

In order to formulate a twin-plant as a Kripke structure, one can simply encode states (of the two copies of the system model) and the observed events of the twin-plant in the state-space of the Kripke structure, i.e., a state in the Kripke structure is defined as a vector (x_1, x_2, σ, ϕ), where x_1, x_2 are the states of the system copies and σ is a feasible (observable) event from both x_1 and x_2, and ϕ is an atomic proposition associated with each state, and which takes e takes one proposition from $\{N, F\} \times \{N, F\}$.

7.6.2 Diagnosability Condition as a Temporal Logic Formula

According to [Jiang and Kumar, 2004], the diagnosability property can be considered as a safety specification, which can be expressed using temporal logics. Therefore, the diagnosisability problem, in the sense of [Cimatti et al., 2003], is reduced to a reachability problem that can be checked by looking for the existence of critical pairs in a twin plant.

The atomic proposition ϕ that expresses the diagnosis condition in a state $p = (x_1, x_2, \sigma, \phi)$ of the Kripke structure is: $\phi : x_1 \in c_1 \wedge x_2 \in c_2$, shortly, we write $\phi_1 : c_1 \wedge c_2$. Therefore, the LTL formula which characterizes the reachability of critical pairs for a diagnosis condition $c_1 \perp c_2$ is:

$$\textbf{LTL-SPEC:} \qquad \boldsymbol{F\ (\ c_1\ \wedge\ c_2\ )}$$

Such a specification can be read as follows: '*there exists a path from the current state in the twin-plant, which contains at least one state that satisfies the diagnosis condition* $c_1 \perp c_2$'.

The corresponding CTL specification is:

$$\text{CTL-SPEC:} \qquad \boldsymbol{EF\ (\ c_1\ \wedge\ c_2\ )}$$

The model-checking problem expressing the diagnosability of the diagnosis condition in the sens of [Cimatti et al., 2003], i.e., the *zero*−diagnosability, is:

$$\boldsymbol{K_{\mathcal{G}}, S_{\mathcal{G}}\ \models\ \neg\, EF\ (\ c_1\ \wedge\ c_2\ )}$$

with $K_{P\times P}$ the Kripke structure corresponding to the twin-plant of P $\mathcal{G}$ and S_P the set of initial states.

The diagnosability problem, in the sense of [Sampath et al., 1995], i.e., the verification of the infinite critical pairs, can be expressed using the following model-checking problem:

$$\boldsymbol{K_{\mathcal{G}}, S_{\mathcal{G}}\ \models\ \neg\, EF\ (EG\ (\ c_1\ \wedge\ c_2\ )\ )}$$

which means that no paths exist in twin-plant $\mathcal{G}$ where from a state (in the path) the diagnosis condition holds infinitely.

7.7 K/K_{min}−Diagnosability as a Model-Checking Problem

As discussed in Chapter 3 (Section 3.4.1), the classic definition of diagnosability requires the existence of a finite delay after the occurrence of any fault, which allows for stating with certainty that the fault has occurred [Sampath et al., 1995]. Therefore, diagnosability only means the existence of an upper bound without specifying its value. A finer version

of diagnosability is called K-diagnosabiliy and requires the quantitative determination of the finite delay (as an integer K). Thus, K-diagnosability means that one can determine with certainty the occurrence of a fault in the system after K observations. Hereafter, we recall the original definition of K-diagnosability introduced by [Dallal and Lafortune, 2011] in the context of automata formalism, more precisely in event-based diagnosis.

Définition 51 *[Dallal and Lafortune, 2011] A given FSA G, it is said to be K-diagnosable if no pair of event-executions S_y, $S_n \in L(G)$ exists such that:*

1. *S_y has an occurrence of a fault event $f \in \Sigma_f$ and S_n does not.*
2. *S_y has at least $K+1$ events after the fault event f.*
3. *$P(S_y) = P(S_n)$.* □

This means that, for any two event-sequence in the system with the same observation, one faulty and the other normal, the system is said to be K-diagnosable if and only if the two executions do not have K (or more) successive identical observation after the occurrence of the fault.

It is important to state that K-diagnosability is stronger than the classic diagnosability. Thus, K-diagnosability of a fault always implies its diagnosability, while the converse is not necessarily true [Basile et al., 2012b]. Furthermore, if a fault is K-diagnosable, it is also $(K+1)$-diagnosable, but it is not necessarily $(K-1)$-diagnosable.

Generally, there are often two main problems to deal with in K-diagnosability. The first is to analyze K-diagnosability for a given value of K, i.e. if any fault is diagnosable within at most K steps (observable events) after its occurrence. The second is to find the minimal value of K (K_{min}) for a diagnosable system.

7.7.1 Reformulation of K-Diagnosability Definition

As mentioned in Section 7.5.2, we define the infinite critical pair as two infinite feasible executions with the same observation, where the first one leads to states in c_1 (normal states) and the second leads to states in c_2 (faulty states). We refer to diagnosability as the absence of such an infinite critical pair. Roughly speaking, the absence of infinite critical pairs means that, after a finite delay either both faulty and normal execution lead to states in the same set of states or they do not generate the same observation any more. Thus, determining this delay means looking for the number K of observable events from which the diagnosis condition becomes diagnosable.

It is worth noticing that the finite delay for diagnosability can be computed either with the number of observable events (in event-based diagnosis) or with the number of reachable

states (in state-based diagnosis). There is an equivalence between these two forms, i.e. if the system is diagnosable in K observable events this means that it is diagnosable in $(K+1)$ reachable states. As we work in a state-based context, in the remainder of this chapter,K-diagnosability is examined with regard to reachable states.

Now, we are ready to state the following definition of K-diagnosability in the context of [Cimatti et al., 2003].

Définition 52 *Given an FSA $\mathcal{Q}$, c_1 and c_2 are two sets of states and $c_1 \perp c_2$ is a diagnosis condition. $c_1 \perp c_2$ is said to be K-diagnosable if no pair of execution s_1 and s_2 exists such that:*

1. $\exists\, t \in \mathbb{N} \mid c_1(x_{s_1}^{t+i}) \wedge c_2(x_{s_2}^{t+i})$, *for* $0 \leq i \leq K$
2. $P(s_1) = P(s_2)$. □

In other terms, a diagnosis condition is said to be K-diagnosable, if no pair of feasible executions with identical observation exists, where the system enters and stays in an ambiguous status for more than K reachable states.

7.7.2 $K-$Diagnosability Reformulation

Our aim now is to reformulate K-diagnosability as a model-checking problem. First, we deal with the problem of checking K-diagnosability for a given value of K.

K-diagnosability is solved by browsing each infinite feasible execution in the twin-plant and checking if an execution exists where the model state reaches and stays in an ambiguous state (x_1, x_2) ($x_1 \in c_1$ and $x_2 \in c_2$) for K successive states.

To achieve this task, we define a *delay* function that associates each state in an infinite feasible execution with the number of the precedent successive ambiguous states. We define the *delay* function as follows,

Algorithm 7 Delay function

$delay : X \times X \rightarrow N$;
$delay(x_1^0, x_2^0) = 0$;
while $(i \geq 0 :)$ **do**
  if $c_1(x_1^{i+1}) \wedge c_2(x_2^{i+1})$ **then**
    $delay(x_1^{i+1}, x_2^{i+1}) = delay(x_1^i, x_2^i) + 1$;
  else
    $delay(x_1^{i+1}, x_2^{i+1}) = 0$;

To check K-diagnosability by model-checking, it is enough to check that the value returned by *delay* function never reaches the value K in all the reachable states. Therefore, the model-checking problem expressing the K-diagnosability property is:

$$\boldsymbol{K_{\mathcal{G}}, S_{\mathcal{G}} \models AG\ delay() < K}$$

If the Kripke structure corresponding to the twin-plant satisfies such specification, it means that the system model $\mathcal{Q}$ is K-diagnosable and thus the system is diagnosable. If it does not satisfy the specification, then the system is not K-diagnosable.

7.7.3 K_{min}−Diagnosability Reformulation

The second main problem related to K-diagnosability is to find the minimum value of K (K_{min}) for which the system is K-diagnosable, i.e., the case when the system is K-diagnosable but not $(K-1)$-diagnosable.

It is possible to adapt an incremental approach by means of successive logical formula in order to find the minimum value K_{min} ensuring diagnosability. The idea is to introduce increment of K in the CTL specification as illustrated below:

$$k = 0 : \quad \neg\, EF\ \phi$$

$$k = 1 : \quad \neg\, EF\ (\ \phi\ \wedge\ EX\ \phi)$$

$$k = 2 : \quad \neg\, EF\ (\ \phi\ \wedge\ EX\ (\ \phi\ \wedge\ EX\ \phi)\ )$$

$$\ldots$$

$$k = K : \quad \neg\, EF\ (\ \phi\ \wedge\ EX\ (\ \underbrace{\phi\ \wedge \ldots \wedge\ EX\ \phi}_{\text{K-1 times}})\ldots)$$

with $\phi = c_1\ \wedge\ c_2$.

For $K = 0$, we find the same specification for expressing $zero-$diagnosability. For $K = 1$, we only extend verification of diagnosis condition to the states reachable upon one observation using CTL formula $EX\ \phi$, which means that ϕ holds in a successor of the current state. In other words, such a formula means that the system remains in ambiguous status after the first observation from the original ambiguous state. We repeat the same procedure for the following values of K.

We can note that this method suffers from the number of model-checking runs, since we repeat the procedure for each specification, which is of high time complexity. Another drawback of this approach is its effectiveness, since the length of a specification is proportional to the value of K. Thus, for great integers it becomes difficult to express

K-diagnosability with CTL specification. Furthermore, as is well known, the complexity of CTL model-Checking is ExpTime−*Complete* [Kupferman and Vard, 2001] relatively to the property size. Thus, this method remains impractical for large systems. To overcome this limitation, an alternative approach based on the central feature of model checking which is the generation and analysis of counter-example, can be explored. The main idea is to establish a CTL specification in such a way that, if this specification is violated, then the model-checker generates a counter-example which contains a sequence of successive ambiguous states. The following formula can be used:

$$\textbf{CTL-SPEC:} \quad \neg \boldsymbol{EF} \left(\phi \wedge \boldsymbol{E} \left[\phi \, \boldsymbol{U} \, \neg \phi \right] \right)$$

with ϕ representing the ambiguous states ($c_1 \wedge c_2$).

This means that no finite sequence containing ambiguous states exists in the twin-plant structure. By analyzing the generated counter-example, we can find the length of this sequence. If we generate all counter-examples, then finding K_{min} is nothing else than finding the maximal length of the sequence of successive ambiguous states among counter-examples. Thanks to the existing algorithms that allow for generating multiple or all counter-examples in one model-checking run [Basu et al., 2003], one can avoid the multiple model-checking runs and thus avoid high time complexity.

7.7.3.1 K_{min}-Diagnosability Using RT-CTL Model-Checking

By analyzing the concept of the K_{min}-Diagnosability, one can observe that the value k_{min} corresponds to the maximal number of successive ambiguous states which an observable event-sequence may contain. Such a concept can be viewed as the problem of generating of the lengthiest counter-example, known in the model-checking framework. Actually, such a problem cannot be expressed using only one CTL/LTL specification. However, it can be easily expressed using an RT-CTL specification (Real-Time CTL) [Alur et al., 1990, Larsen et al., 1995, Henzinger et al., 1994, Alur et al., 1993, Cimatti et al., 2000, Bellini et al., 2000]. Such a temporal logic allows expressing the properties with quantitative information regarding the number of events/states or discrete time, for instance, determining the min/max delay between two CTL properties which hold along a path. The NuSMV model-checker [Cimatti et al., 2000] is one among model-checking tools which handle such problems.

The RT-CTL specification expressing the K_{min}-Diagnosability is : MAX$[\phi, \neg\phi]$, with 'MAX' is a temporal operation indicating the maximal delay, exists in the Kripke structure, between properties ϕ and $\neg\phi$ (ϕ expresses the diagnosis condition $c_1 \wedge c_2$). Finally, the model-checking problem expressing the K_{min}-Diagnosability is :

$$\boldsymbol{K_{\mathcal{G}}, S_{\mathcal{G}}} \models \boldsymbol{MAX}[\phi, \neg\phi]$$

The output decision through such a model-checking problem is either a finite integer, which represents the K_{min} value, or infinity value ($+\infty$), which means that the model is non-diagnosable.

7.8 Diagnosability Verification Using NuSMV Model-Checker

For the actual verification of the various reformulation of diagnosability properties that we have established, we use the symbolic model-checker NuSMV [Cimatti et al., 2000] (version 2.5.4), originated from the reengineering, reimplementation and extension of the CMU SMV tool [Srinivasan and Gluch, 1998, Plath and Ryan, 2000], the original BDD-based model-checker developed at CMU [McMillan, 1993]. In fact, NuSMV is one of the most popular model-checkers for temporal logics (LTL, CTL, RT-CTL, PSL, etc.). It is equipped with a dedicated modeling language (SMV language) which allows for the representation of synchronous/asynchronous finite state/transition systems as Kripke structures and the analysis of specification expressed in various temporal logics. Its main advantage is the integration of model checking techniques based on propositional satisfiability analysis (SAT), which is currently enjoying a substantial success in several industrial fields.

Aiming at automatizing the generation of formal models (as a NuSMV modules) for the twin-plant and the integration of the various specifications expressing the diagnosability properties, we developed a platform, in C# programming language, that carried out these tasks. In fact, the platform takes event-based automata (*.fsm files), and the diagnosability properties to be checked, as input and generates a NuSMV model (*.smv) which contains NuSMV modules describing the twin-plant and the specification to be checked in the corresponding temporal logic (LTL, CTL, or RT-CTL). An illustrative example, which details the procedure for checking diagnosability of permanent faults using NuSMV, is given in Appendix A.

7.9 Experimentation

In order show the applicability of the proposed technique, we perform some experimentation through a benchmark, taken from [Ghazel and Liu, 2016], which abstracts a level crossing system and depicts the concept of permanent faults.

7.9.1 Presentation of the Level Crossing Benchmark

A level crossing (LC) is an intersection where a railway line intersects with a road or path at the same level. It can be seen as a composition of three subsystems: the railway traffic, the LC controller and the barriers. Such subsystems are modeled by Labeled Petri Nets (LPN). The global LC system is established using some shared places and transitions between these sub-models (See Figure 7.4). The LC model has two classes of faults represented as unobservable transitions. The first one (Σ_{F1}) pertains to train-sensors along the track and may cause the arrival of the train into the LC intersection zone before the barriers are ensured to be lowered. This fault class is represented by the occurrence of unobservable (faulty) transition t_6 in the benchmark. The other class (Σ_{F2}) indicates a barrier failure and is manifested by an early raising of the barriers. It is represented by the occurrence of unobservable (faulty) transitions $t_{i,5}$ (for $1 \leq i \leq n$) in the benchmark.

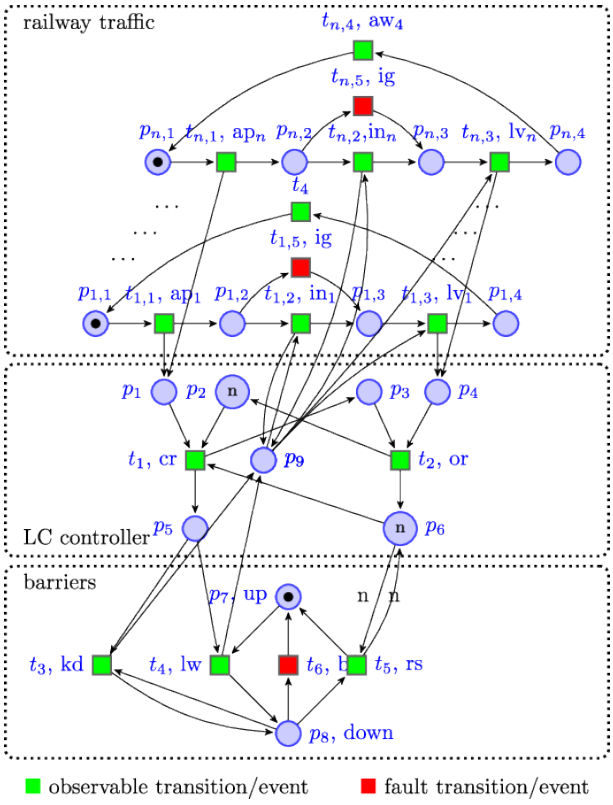


Figure 7.4 – The Level Crossing Benchmark

An interesting feature of this benchmark is that it can be extended to n railway tracks to obtain larger models. For more details about the modeling and the development of the

benchmark, the reader can refer to [Liu et al., 2014a, Liu et al., 2016, Ghazel and Liu, 2016, Ghazel and El-Koursi, 2014].

As the LC system is modeled by an LPN, we first generate the reachability graph of the LPN considered with the help of the TINA tool [Berthomieu et al., 2007]. We note that the reachability graph is represented by an automaton in an event-based formalism, i.e. faults are treated as unobservable events rather than faulty states. Thus, we have to make a transformation from the event-based representation to the state-based one (as discussed in Section 7.4).

Before proceeding with tests, we construct the twin-plant as a Kripke structure. It is described as a synchronous composition of two copies of system modules instead of enumerating the whole model. In order to reduce the state-space, we take into account the symmetric property of the twin-plant regarding the ambiguous states. Thus, we consider one copy of the system as to be faulty-free by deleting the faulty states.

Then, the verification task is conducted as follows:

1. Checking diagnosability of Σ_{F1} fault class (resp. Σ_{F2}): we define the diagnosis condition $c_1 \perp c_2$, with c_1 the set of normal states and c_2 the set of faulty states w.r.t. fault class. This task is performed using our extended formulation of diagnosability.

2. Determining the K_{min}-diagnosability: if the fault class Σ_{F1} (resp. Σ_{F2}) is diagnosable, we aim to find the minimal delay ensuring its diagnosability. This task follows the diagnosability verification immediately if the system is diagnosable.

7.9.2 Results and Discussion

The experimental results are summarized in Table 1, where the columns from left to right correspond to: fault class, number of railway tracks, number of states in the reachability graph of the LPN model, number of states of the state-based FSA obtained after the transformation, number of reachable states generated by the Model-Checker, diagnosability verdict, time elapsed by the model-checker, K_{min} ensuring diagnosability if the system is diagnosable. All experiments were conducted on a 64-bit PC, Ubuntu 14.04, an Intel Core i5, 2.5 GHz Processor with 4 cores and 4 GB RAM.

The analysis of diagnosis conditions shows that the 1^{st} class of faults is diagnosable only for the 1-track LC model where the 2^{nd} fault class is diagnosable for any number of tracks in the LC model. These results correspond to those obtained in [Liu, 2014]. Once the diagnosability is checked for the fault class Σ_{F2}, we immediately proceed to the determination of the value K_{min} ensuring the diagnosability.

We have used BDD-based and SAT-based model-checking techniques. The first failed, while the second achieved the task. The difference between the results is caused by the fact that the BDD-based technique investigates only the infinite paths, whereas the SAT-based

Table 7.1 – Experimental results for LC models

Σ_{Fi}	n	MG	X	RS	$Diag$	$Time$	K_{min}
Σ_{F1}	1	24	38	121	YES	00.01 s	4
	2	216	416	3013	NO	00.50 s	–
	3	1632	3240	40603	NO	50.20 s	–
Σ_{F2}	1	24	38	115	YES	00.01 s	6
	2	216	352	2573	YES	00.28 s	13
	3	1632	2604	38310	YES	21.51 s	20

technique is also able to deal with finite paths. Thus, it can check the existence of finite ambiguous state execution. The analysis of the counter-examples generated confirms the correctness of our extended diagnosability formulation.

A few seconds are sufficient for NuSMV to perform the analysis which is quite promising, given the size of the models dealt with, especially for $n = 3$. It is worth noticing that no reduction or optimization technique performed on the model. We can see that the runtime when the fault is not diagnosable is greater that the runtime when it is diagnosable. This difference is mainly due to the delay necessary for generating counter-examples. Compared to the results obtained through the UMDES tool [Liu, 2014], the model-checking based technique provides better results in terms of time cost. Moreover, (1) our tool decides about non-diagnosability of the first fault class, however UMDES fails and (2) it investigates also K_{min}-diagnosability which is a finer result than only a verdict about diagnosability.

7.10 Conclusion

In this work, we discuss the reformulation of various diagnosability issues in a model-checking framework. We have first extended Cimatti's diagnosability definition in the aim of complying with the classic definition [Sampath et al., 1995]. Then, we have formulated the K-diagnosability as a model-checking issue using a CTL specification and the K_{min}-diagnosability as a RT-CTL one. These reformulation have been illustrated through a level crossing benchmark. The promising results obtained from the conducted experimentation in our work (and also from other similar works), urge us to extend such a framework in order to deal with more complex types of failures such as faults specified by supervision patterns and other notions for diagnosability of repeated/intermittent faults. Some of these extensions are the subject of the next chapter.

Chapter 8

Twin-Plant Based Approach for Intermittent Faults Diagnosis

Sommaire

Summary

This chapter focuses on the diagnosability analysis of intermittent faults in discrete-event systems. Various notions of diagnosability introduced in Chapter 5 are carried out, in this chapter, using a twin-plant based approach. Firstly, the necessary and sufficient conditions for the weak diagnosability properties are established and proved. Regarding the strong diagnosability properties, necessary conditions are first deduced (from the necessary and sufficient conditions for the weak notions) and, then, the necessary and sufficient conditions are provided and proved. On-the-fly and incremental algorithms are then established for the actual checking of the elaborated conditions.

The work presented in this chapter is the subject of publications in ICPHM'16 [Boussif and Ghazel, 2016b], *WoDES* [Boussif et al., 2016c], *VeCOS'16* [Boussif and Ghazel, 2016d] *and a submitted journal paper on IEEE– TASE* [Boussif and Ghazel, 2016a].

This chapter is structured as follows: Section 8.2 discusses the twin-plant construction and provides the necessary and sufficient conditions for *weak* diagnosability notions introduced in Section 5.5.2. Dedicated algorithm for checking such conditions are also provided. Necessary and sufficient conditions for the *strong* notions of diagnosability are developed in Section 8.3, where the associated checking algorithms are also discussed. Reformulation of (some of) these diagnosability properties as LTL model-checking problems is discussed in Section 8.4). A Benchmark is used to illustrate the applicability of such a reformulation and to assess its efficiency is provided in Section 8.5. Finally, Section 8.7 draws some concluding remarks.

8.1 Introduction

Most of the approaches in the literature pertaining to intermittent fault diagnosis deal with the diagnosability of intermittent faults on the basis of the structural analysis of the so-called "diagnoser" (as widely discussed in Section 5.2 (of Chapter 5)). However, developing such a model suffers from the combinatorial explosion problem and shows a high complexity level (exponential) regarding the state-space of the system model. Instead, we use the twin-plant structure [Jiang and Huang, 2001, Yoo and Lafortune, 2002b, Cimatti et al., 2003] to perform diagnosability analysis in the current work. In fact, the twin-plant shows a lower complexity (polynomial), which can help to tackle the combinatorial explosion problem (as discussed in Chapter 4).

8.2 Analysis of the Weak Diagnosability

This section is dedicated to the analysis of WF and WR-diagnosability by developing necessary and sufficient conditions for each notion before elaborating algorithms to check these conditions. The procedure to establish such conditions is based on the twin-plant structure [Jiang and Huang, 2001]. In what follows, we discuss the twin-plant construction.

8.2.1 Twin-Plant Construction

The twin-plant, firstly introduced in [Jiang and Huang, 2001], simply consists of two synchronized copies of generator G' of the system model G, while the synchronization is performed. Thus, any event-trace in the twin-plant corresponds to a pair of event-traces in the system model that share the same observation. More precisely, a path in the twin-plant corresponds to two *indistinguishable* traces in the system model.

To keep tracking the system status labels, we use constructed generator G'_ℓ, instead of constructed generator G'. Then, in order to generate a reduced state-space of the twin-plant (by generating only the behavior of interest for fault diagnosis), a synchronous composition $G'_\ell||G'_{\ell F}$ is performed, which is different from that in [Jiang and Huang, 2001]. In fact, $G'_{\ell F}$ depicts only the co-accessible part of generator G'_ℓ from faulty states, i.e. it only contains the generated faulty event-traces. Thus, $G'_{\ell F} = \langle X_{oF}, \Sigma_o, \delta_{oF}, x_0\rangle$, where X_{oF} is the set of states in G'_ℓ that are reachable by event-traces that, in turn, contain at least one fault event. For more details about generating $G'_{\ell F}$, the reader can refer to [Moreira et al., 2011].

Définition 53 *(The reduced twin-plant)*

A reduced twin-plant of model G is an FSA $\mathcal{P} = \langle \mathcal{Q}, \Sigma_o, \Gamma, q_0\rangle$, where:

- $\mathcal{Q} \subseteq \{(x, x') \mid x \in X_o, x' \in X_{oF}\}$ *is the set of states.*
- Σ_o *the set of observable events.*
- $\Gamma : \mathcal{Q} \times \Sigma_o \to 2^{\mathcal{Q}}$ *is the partial transition relation. $q' \in \Gamma(q, \sigma)$, with $q = (x_1, x_2)$, and $q' = (x'_1, x'_2)$ if and only if $x'_1 \in \delta_o(x_1, \sigma)$, $x'_2 \in \delta_{oF}(x_2, \sigma)$.*
- $q_0 = (x_0 \times x_0) \in \mathcal{Q}$ *is the initial state.*

□

It is worthwhile recalling that constructing the twin-plant can be performed in $(\mathcal{O}(|X|^4 \times |\Sigma_o|))$ [Jiang and Huang, 2001].

As the reduced twin-plant is established directly based on the constructed generator G'_ℓ, then label tracking is preserved and therefore, the *fault-assignment* function is extended as follows:

$$\Psi : \mathcal{Q} = (X_o, X_{oF}) \to (\{N, F, R\} \times \{N, F, R\})$$

Hence, different types of states can be distinguished between in the reduced twin-plant. Hereafter, only state types which will be used in the sequel for developing necessary and sufficient conditions for weak diagnosability are defined.

Définition 54 *(Types of states in the twin-plant)*

We define the following state types,

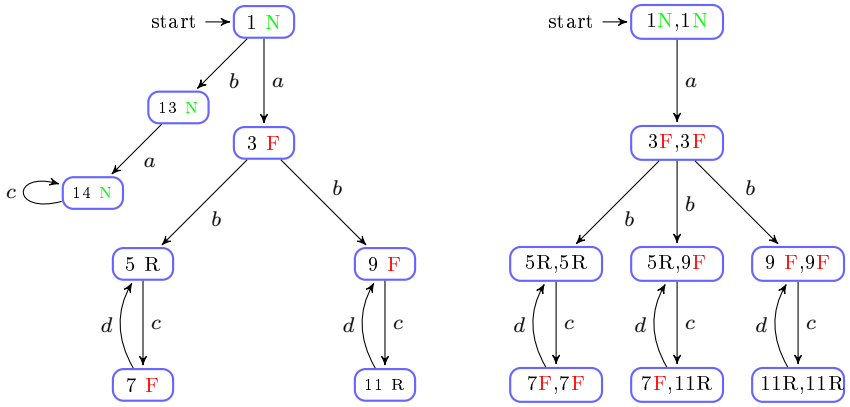
- ***N-state*** *(resp.* ***F-state***, ***R-state****): is a state $q = (x, x') \in \mathcal{Q}$, such that $\Psi(q) = (N, N)$ (resp. $\Psi(q) = (F, F)$, $\Psi(q) = (R, R)$).*
- ***NF-state****: is a state $q = (x, x') \in \mathcal{Q}$, such that $\Psi(q) = (N, F)$.* ***FN-state*** *is defined similarly.*

- ***NR-state**: is a state $q = (x, x') \in \mathcal{Q}$, such that $\Psi(q) = (N, R)$. **RN-state** is defined similarly.*
- ***N1-state**: is a state $q = (x, x') \in \mathcal{Q}$, such that $\Psi(q) = (N, \triangle)$. with $\triangle \in \{N, F, R\}$.*
- ***non-N-state** (resp. **non-F-state**, **non-R-state**): is a state which is not an **N-state** (resp. **F-state**, **R-state**).*

□

One can underline the fact that the twin-plant has an interesting feature, which is the symmetric property. It means that each path in the twin-plant has its symmetric path (e.g., a path containing FN-states has its symmetric path which contains the symmetric NF-states, and vice versa). In the following section, we take into account this property for developing the necessary and sufficient conditions.

Example 15 *Figure 8.1 (a) depicts generator G'_ℓ corresponding to automaton G (cf. Figure 5.3 in Chapter 5), and Figure 8.1 (b), depicts the reduced twin-plant $\mathcal{P}$ (taking into account the symmetric property) of automaton G.*



(a) Generator G'_ℓ (b) Reduced twin-plant $\mathcal{P}$

Figure 8.1 – Example 15

8.2.2 Necessary and Sufficient Conditions

The twin-plant structure has been used to establish a necessary and sufficient condition for the diagnosability of permanent faults [Jiang and Huang, 2001, Boussif and Ghazel, 2015a]. Such a condition stipulates that there is no *F-uncertain cycles* in the constructed

twin-plant, i.e., cycles that are composed only of NF (or FN)-states. Here, we also use the twin-plant structure to establish necessary and sufficient conditions for the weak diagnosability of intermittent faults. To do so, some further concepts will be introduced in what follows.

Définition 55 *(F-confused cycle)*

An F-confused cycle is a cycle $c\ell = (q_1, \sigma_1, q_2, \ldots, q_n, \sigma_n, q_{n+1} = q_1)$ *in the twin-plant, s.t.* $\forall\, 1 \leq i \leq n$, q_i *is an* $N1$*-state, and* $\exists\, j : 1 \leq j \leq n$, *s.t.* q_j *is an* NF*-state.*

□

An *F-confused cycle* in the twin-plant corresponds to two cyclic traces in the system model (automaton G) which generate the same observed trace, such that the first one has no fault event (a fault-free cyclic trace) and the second one contains at least one fault event (which explains the existence of an NF-state in the twin-plant).

Définition 56 *(R-confused cycle)*

An R-confused cycle is a cycle $c\ell = (q_1, \sigma_1, q_2, \ldots, q_n, \sigma_n, q_{n+1} = q_1)$, *in the twin-plant, s.t.* $\forall\, 1 \leq i \leq n$, q_i *is an* $N1$*-state, and* $\exists\, 1 \leq j \leq n$, *s.t.* q_j *is an* NR*-state.*

□

After having set up these preliminary notions, the necessary and sufficient conditions for the weak diagnosability of intermittent faults can be stated.

Theorem 10 *(Necessary & sufficient conditions for* WF *and* WR*-diagnosability)*
A system model G, *w.r.t projection* P, *class of fault events* Σ_f *and its corresponding class of reset events* Σ_r, *is:*

1. WF*-diagnosable, if and only if no F-confused cycle exists in its corresponding twin-plant.*
2. WR*-diagnosable, if and only if no R-confused cycle exists in its corresponding twin-plant.*

□

Hereafter, we provide a proof of this Theorem regarding $WF-$diagnosability.

Proof. *(⇒) Assume that* $L(G)$ *is* WF*-diagnosable and there exists an F-confused cycle in its corresponding twin-plant:* $q_1, \sigma_1, q_2, \ldots, q_n, \sigma_n, q_1$, $n \geq 1$. *Such a cycle corresponds to two cycles in* G'_ℓ: $c\ell = x_1^1, \sigma_1 x_2^1, \ldots, x_n^1, \sigma_n, x_1^1$ *and* $c\ell' = x_1^2, \sigma_1, x_2^2, \ldots, x_n^2, \sigma_n, x_1^2$. *Let* $t = v_1, \sigma_1, v_2, \sigma_2, \ldots, v_n, \sigma_n$ *and* $t' = v'_1, \sigma_1, v'_2, \sigma_2, \ldots, v'_n, \sigma_n$ *be the event-traces that correspond*

to cyclic executions $c\ell$ and $c\ell'$ in G s.t. $\forall\, i \leq n, v_i, v'_i \in \Sigma_u^*$. (i.e., $P(t) = P(t') = \sigma_1, \sigma_2, \ldots, \sigma_n$).

By construction of the twin-plant, $\exists\, s_0, s'_0 \in L(G)$, s.t.

$$[P(s_0) = P(s'_0)] \wedge [\delta(x_0, s_0) = x_1^1] \wedge [\delta(x_0, s'_0) = x_1^2] \wedge [\Sigma_f \notin s_0].$$

(the last condition $\Sigma_f \notin s_0$ is due to the fact that x_i^1 is an N-state $\forall 1 \leq i \leq n$). *Also, according to Definition 55, $\Sigma_f \in t'$ and $\Sigma_f \notin t$. Thus, one can consider $t' = t'_1.t'_2$ such that $t_1^{\prime|t'_1|} \in \Sigma_f$ (i.e., t'_1 ends with a fault event). Now, let us consider $s = s'_0.t'_1$, then $s \in \psi(\Sigma_f)$. Thus, for any $n \in \mathbb{N}$ let us take $t''_n = t'_2(t')^n \in L/s$, then $(|t''_n| \geq n)$ and $(\exists\, \omega_n = s_0.(t)^{n+1})$ such that $[\omega_n \in P^{-1}P(st''_n)] \wedge [\Sigma_f \notin \omega_n]$.*

Therefore, WF-diagnosability definition is violated.

($\Leftarrow$) Assume that automaton G is non-WF-diagnosable. Then,
$(\forall n \in \mathbb{N})(\exists\, s \in \psi(\Sigma_f))\ (\exists\, t \in L(G)/s)$ *such that:*

$$[|t| \geq n] \wedge [(\exists\, \omega \in P^{-1}P(s.t)) \wedge [\Sigma_f \notin \omega]]$$

Let us suppose that twin-plant $\mathcal{P}$ is F-confused-cycle-free and pick any $n \geq |X|^2$, $\omega \in L(G)$ such that $P(\omega) = P(s.t) = \sigma_1, \sigma_2, \ldots, \sigma_k$, with $k \in \mathbb{N}$. By constructing twin-plant $\mathcal{P}$ of G, we have a path $\pi = q_0, \sigma_1, q_1, \ldots, \sigma_k, q_{k+1}$, $k \leq |s.t|$ that corresponds to executions ω and $s.t$.

As $|X|^2 \leq n \leq |t|$, it is clear that executions corresponding to $s.t$ and ω contain cycles [Jiang and Huang, 2001]. Thus, $\exists\, i :\ 0 \leq i \leq k'$, with $k' \leq k$ such that $c\ell \in \pi$, with $c\ell = q_i, \sigma_{i+1}, q_i, \ldots, q_{k'-1}, \sigma_{k'}, q_i$ (i.e., a cycle $c\ell$ exists in π).

Since $\Sigma_f \notin \omega$, then $\forall q \in c\ell$, q is an $N1$-state. Moreover, $\Sigma_f \in s$ (since $s \in \psi(\Sigma_f)$). According to assumption (A3), the fault event occurs and reset regularly. Then, $\exists\, i \leq k'$ s.t. q_i is an NF-state. Thus, $c\ell$ is an F-confused cycle, according to Definition 53, which contradicts our assumption. ■

The proof of the necessary and sufficient condition regarding WR-diagnosability is omitted as it is similar to that of WF-diagnosability.

Corollary 4 *Let an automaton G satisfy assumptions (A1−A3). Then, G is WF-diagnosable if and only if G is WR-diagnosable.*

This corollary has been introduced in Chapter 5 (Corollary 3). Hereafter, we give its formal proof.

Proof. Assume that G is not WF-diagnosable. Then, from Definition 18, there exists, in $\mathcal{P}$ (its corresponding twin-plant), a set of states $q_1, q_2, \ldots, q_n$ that form an F-confused cycle. This cycle corresponds to two cycles in G: $x_1^1, x_2^1, \ldots, x_n^1 = x_1^1$ and $x_1^2, x_2^2, \ldots, x_n^2 = x_1^2$ such that $\forall i \leq n$, $\Psi(x_i^1) = N$ and $\exists\, j \leq n$ s.t. $\Psi(x_j^2) = F$ (cf.

Definition 5.). By assumption (A3), each fault is recovered within a finite delay. Consequently, from Definition 18, $\exists\, i' \leq n$ s.t. $\Psi(x^2_{i'}) = R$. Therefore, states $q_1, q_2, \ldots, q_n$ also form an R-confused cycle in $\mathcal{P}$. Hence, automaton G is not WR-diagnosable.

By similar reasoning, we prove that, if automaton G is not WR-diagnosable, then it is not WF-diagnosable. Indeed, it suffices to note that, by assumption (A3), each reset event is followed by a new occurrence of a fault event in a finite delay. ■

Example 16 *From reduced twin-plant $\mathcal{P}$ (cf. Figure 8.1) corresponding to automaton G (cf. Example 7), one can infer that G is WF-diagnosable (and then WR-diagnosable), since no F-confused cycle exists in its corresponding reduced twin-plant $\mathcal{P}$.*

8.2.3 Verification Algorithm for Weak Diagnosability

The proposed algorithm for checking the weak diagnosability consists in constructing, on the fly, the reduced twin-plant $\mathcal{P}$ corresponding to model G, while checking in parallel the necessary and sufficient condition for WF-diagnosability according to Theorem 10. It should be noticed that only the verification algorithm for WF-diagnosability is discussed, since it also applies for WR-diagnosability in the same manner.

Algorithm 8 is based on a depth-first search procedure. It builds twin-plant $\mathcal{P}$ and simultaneously checks the necessary and sufficient condition for WF-diagnosability. As soon as a cycle is found in $\mathcal{P}$, function CHECK_F-CONFUSED_CYCLE() (cf. Algorithm 8, Line 29) checks whether this cycle is an F-confused cycle or not. If it is an F-confused cycle, then G is not WF-diagnosable and the verification process is stopped. Otherwise, the construction and verification process continues. If the whole state-space of $\mathcal{P}$ is generated, through Algorithm 8, without looking for any F-confused cycle, then model G is stated to be WF-diagnosable (cf. Algorithm 8, Line 33).

Below, functions and data, used in the algorithm, are briefly explained:

- STACK.PUSH, STACK.TOP, STACK.POP: are the usual methods to handle a stack.
- ENABLE(q): returns the set of output (observable) events σ_i from state x. Formally, $Enable(x) = \{\sigma \in \Sigma_o \mid \delta_{G'}(x, \sigma) \neq \emptyset\}$.
- LIST_CURRENT: The list of twin-plant states that need to be handled.
- ADD(), REMOVELAST(): two functions used to handle lists of twin-plant states.
- CHECK_F-CONFUSED_CYCLE(): the function that checks the existence of an F-confused cycle in LIST_CURRENT. Such a function returns a Boolean value: TRUE if a cycle is found; otherwise, it returns FALSE.

Algorithm 8 Generating the reduced twin-plant and checking the existence of F-confused cycle

Input: $G'_\ell = (X_o, \Sigma_o, \delta_o, x_0)$, $G'_{\ell F} = (X_{oF}, \Sigma_o, \delta_{oF}, x_0)$

Output: $\mathcal{P} = G'_\ell || G'_{\ell F} = (\mathcal{Q}, \Sigma_o, \Gamma, q_0)$

9 $\Gamma \leftarrow \emptyset$, STACK $\leftarrow \emptyset$ // Initialization step

10 $\mathcal{Q} \leftarrow \{q_0\} = \{(x_0, x_0)\}$ // Creating the initial state of twin-plant

11 LIST_CURRENT $= \{(x_0, x_0)\}$

12 STACK.PUSH($\langle (x_0, x_0)$, ENABLE$(x_0) \times$ ENABLE$(x_0) \rangle$)

// Formally, ENABLE(x)= $\{\sigma \in \Sigma$ s.t. $\delta(x, \sigma) \in X\}$.

13 **while** STACK $\neq \emptyset$ **do**

14 $\langle (x, x'),$ ENABLE_SET$\rangle$ = STACK.TOP // Take the top element in the stack

15 **if** ENABLE_SET$= \emptyset$ **then**

16 $\langle (x, x'),$ ENABLE_SET$\rangle$ = STACK.POP // Remove the element from stack

17 **else**

18 (σ, σ') = REMOVELAST(ENABLE_SET) // Pick up an element

19 **if** $(\sigma = \sigma')$ **then**

// Only the shared observable event are handled

20 $x_1 \leftarrow \delta(x, \sigma)$ // Get back the target state

21 $x'_1 \leftarrow \delta(x', \sigma')$

22 **if** $(x_1, x'_1) \notin \mathcal{Q}$) **then**

23 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x_1, x'_1)\}$ // Update the set of states

24 $\Gamma \leftarrow \Gamma \cup \{(x, x') \xrightarrow{\sigma} (x_1, x'_1)\}$ // Update the transition function

25 LIST_CURRENT.ADD$((x_1, x'_1))$

26 STACK.PUSH($\langle (x_1, x'_1)$,ENABLE$(x_1) \times$ENABLE$(x'_1) \rangle$) // Pile up the current element

27 **else**

28 $\Gamma \leftarrow \Gamma \cup \{(x, x') \xrightarrow{\sigma} (x_1, x'_1)\}$

29 **if** $((x_1, x'_1) \in$ LIST_CURRENT) **then**

30 **if** CHECK_F-CONFUSED_CYCLE*()* **then**

31 **return** "non-WF-diagnosable"

// Check for F-Confused-cycle

32 REMOVELAST(LIST_CURRENT)

33 **return** "WF-diagnosable"

8.3 Analysis of the Strong Diagnosability

In this section, we firstly provide the necessary conditions for SF-(and SR-)diagnosability on the basis of the necessary and sufficient conditions for the weak diagnosability developed above. Then, we establish the necessary and sufficient conditions for SF-(and SR-)diagnosability on the basis of the reduced twin-plant introduced in Section 8.2.1.

It is worth underlining that the necessary conditions are firstly developed since such conditions can be used to speed up the analysis. In fact, the model is stated to be non-SF-diagnosable as soon as these conditions are violated (i.e., there is no need to continue the construction of the twin-plant and check the necessary and sufficient condition).

8.3.1 Necessary Conditions

Firstly, according to Proposition 5 (introduced in Chapter 5), it is easy to infer that the necessary and sufficient condition for WF-diagnosability (resp. WR-diagnosability) represents a necessary condition for SF-diagnosability (resp. SR-diagnosability). That is, the presence of an F-confused cycle in the twin-plant implies the non-WF-diagnosability and, therefore, the non-SF-diagnosability, since that non-WF-diagnosability $\Rightarrow$ non-SF-diagnosability (cf. Proposition 5). However, it is only a necessary condition, which means that the absence of F-confused cycle does not imply the SF-diagnosability, as witnessed in Example 9. In what follows, the notions of F-confused and R-confused cycles are used to develop stronger necessary conditions for SF-(and SR-)diagnosability. We firstly introduce two particular cycles.

Définition 57 *(non-F-cycle, non-R-cycle)*

- *A non-F-cycle, is a cycle* $c\ell = (q_1, \sigma_1, q_2, \ldots, q_n, \sigma_n, q_{n+1} = q_1)$ *in twin-plant* $\mathcal{P}$*, such that* $\forall\, 1 \leq i \leq n$, q_i *is a non-F-state and at least one state is a non-N-state.*
- *A non-R-cycle, is a cycle* $c\ell = (q_1, \sigma_1, q_2, \ldots, q_n, \sigma_n, q_{n+1} = q_1)$ *in twin-plant* $\mathcal{P}$*, such that* $\forall\, 1 \leq i \leq n$, q_i *is a non-R-state and at least one state is a non-N-state.*

Remark 5 *It should be noticed that an F-confused cycle is also a non-F-cycle, since an N1-state is also a non-F-state and an NF-state is a non-N-state as well. In an analogous way, an R-confused cycle is also a non-R-cycle.*

Now, we state a necessary condition for SF (and SR)-diagnosability.

Proposition 14 *(Necessary conditions for the strong diagnosability)*

A DES model G, w.r.t. projection P, class of fault events Σ_f and class of reset events Σ_r, is:

- *non-SF-diagnosable if a non-F-cycle exists in its corresponding twin-plant.*
- *non-SR-diagnosable if a non-R-cycle exists in its corresponding twin-plant.*

Proof.

The proof of this proposition is straightforward. Indeed, the existence of a non-F-cycle in the twin-plant means that two event-traces which share the same observation exist in the generator G'_ℓ and lead to two cycles: $c\ell = x_1, \sigma_1 x_2, \ldots, x_n, \sigma_n, x_1$ and $c\ell' = x'_1, \sigma_1, x'_2, \ldots, x'_n, \sigma_n, x'_1$ such that $\forall 1 \leq i \leq n$: $\Psi(x_i)$ and $\Psi(x'_i)$ will never be F at the same time. Therefore, it follows that it is not possible to identify the status of the system along this infinite observable event-trace. Thus, according to Definition 20, the model is non-SF-diagnosable. An analogous reasoning can be made for SR-diagnosability. □

Let us now establish the necessary and sufficient conditions for SF and SR-diagnosability on the basis of the reduced twin-plant.

8.3.2 Necessary and Sufficient Conditions

In order to formalize the necessary and sufficient conditions for SF (and SR)-diagnosability, the following definitions are introduced.

Définition 58 *(Types of state-traces in the twin-plant)*

- *A **path** $\wp$ in twin-plant $\mathcal{P}$ is a state-trace $(q_1, q_2, \ldots, q_n)$ such that $\forall q_i, q_{i+1} \in \wp, \exists \sigma_i \in \Sigma_o$ satisfying $q_{i+1} \in \Gamma(q_i, \sigma_i)$, for $(1 \leq i < n)$.*
- *$\wp$ is a **closed path**, if it is a path whose ending state is also the starting one (i.e., $q_n = q_1$).*
- *$\wp$ is a **generated path** if q_1 is the initial state of twin-plant $\mathcal{P}$ (i.e., $q_1 = q_0$).*
- *$\wp$ is an **elementary path** if no state $q_i \in \wp$ is visited twice (i.e., $\forall i, j \in \{1, \cdots, n\}$ and $i \neq j$, we have $q_i \neq q_j$).*
- *$\wp$ is an **elementary cycle** if it is a closed path whose states are different from each other, except for the first and the last ones.*
- *$\wp = \wp' c\ell_\wp$ is a **prime path** if $\wp'$ is an elementary path and $c\ell_\wp$ is an elementary cycle.*

- $\wp = \wp' c\ell_\wp$ *is a* ***generated prime path*** *if* $\wp'$ *is a generated elementary path and* $c\ell_\wp$ *is an elementary cycle.*

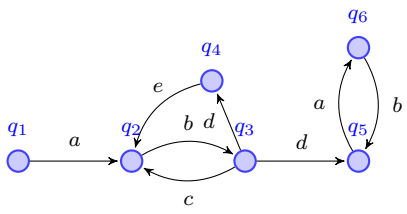
For more details about the above definitions, the reader can refer to [Zhou and Kumar, 2009].

Remark 8 *It is worth noticing that:*

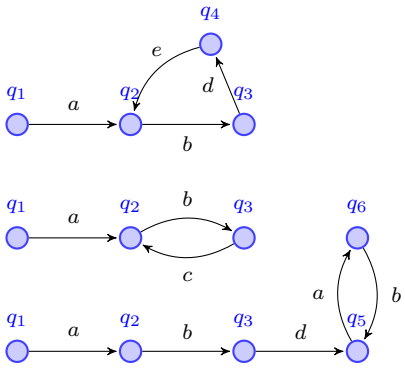
1. *An infinite path is composed of a (possibly empty) elementary path and a set of elementary cycles.*
2. *A closed path is composed of one or more elementary cycles.*

Définition 59 *(Set of prime-paths (SPP))*
For any infinite path $\wp$*, we define* $\mathbb{P}_\wp$ *as the set of prime-paths associated with* $\wp$. □

Example 17 *Let us consider the infinite path* $\wp = (q_1, (q_2, q_3, q_4)^3, (q_2, q_3)^1, (q_5, q_6)^*)$ *in* G*, depicted in Figure 8.2 (a). The set of prime-paths associated with* $\wp$ *is* $\mathbb{P}_\wp = \{(q_1, (q_2, q_3, q_4)^*), (q_1, (q_2, q_3)^*), (q_1, q_2, q_3, (q_5, q_6)^*)\}$ *is depicted in Figure 8.2 (b).*



(a) An FSA model G



(b) Sub-models of the FSA G

Figure 8.2 – Illustration of a set of prime paths (SPP).

Définition 60 *(The associated set of generated prime-paths (ASGPP))*
For a given observable event-trace $s = (\sigma_1, \sigma_2, \dots) \in \Sigma_o^*$ *associated with generated prime path* $\wp \in Tr(\mathcal{P})$*:*

$$\Pi_s(\mathcal{P}) = \{\wp = (q_0, q_1, \dots) \in Tr(\mathcal{P}) \text{ is prime-path} \,|\, \forall\, i \geq 0,\ q_{i+1} \in \Gamma(q_i, \sigma_{i+1})\}$$

is the set of all generated prime-paths associated with s*.*

One recalls that $Tr(\mathcal{P})$ is the set of all state-traces of the (reduced) twin-plant. Each prime-path $\wp$ in $\Pi_s(\mathcal{P})$ has the following form $\wp = \wp' c\ell^*$, where $\wp'$ is a generated elementary path, and $c\ell$ is an elementary cycle.

Définition 61 *(F-Interception condition)*
Let $\wp$ be a generated prime-path and $s \in \Sigma_o^*$ its associated observable event-trace, then the F-Interception condition relatively to $\wp$ is defined as follows:
$\exists\, k \in \mathbb{N}$ *s.t.* $\forall \wp_i = \wp'_i c\ell^*_{\wp_i} = (q_0, q_1, \dots) \in \Pi_s(\mathcal{P})$*:*

1. *q_k is an F-state.*
2. *$q_k \in c\ell^*_{\wp_i}$.*

The *F-Interception* condition ensures that, after a finite delay, all the generated prime-paths that share the same observation as $\wp$ (i.e., event-trace s) reach F-states, in their corresponding elementary cycles at the same time (i.e., after $k-1$ observations).

Example 18 *Let us consider twin-plant $\mathcal{P}$ (the corresponding system model is not given), depicted partially in Figure 8.3. Each state q_i in twin-plant $\mathcal{P}$ is represented by its corresponding labels, i.e., $q_1 = (x_1, x'_1)$ is represented by the labels of x_1 and x'_1 (here, $\Psi(x_1) = \Psi(x'_1) = N$). Let us consider event-trace $s = ab\,(cd)^*$ associated with the prime-path: $q_1, q_2, (q_3, q_4, q_5, q_6)^*$. Then, $\Pi_s(\mathcal{P}) = \{(q_1, q_2, (q_3, q_4, q_5, q_6)^*),\ (q_1, q_7, q_8, (q_9, q_{10})^*),\ (q_1, q_7, q_{11}, q_{12}, (q_{13}, q_{14})^*)\}$. One can deduce that for $k = 6$ (i.e. after 5 observations), the k^{th} states of all the generated prime-paths in $\Pi_s(\mathcal{P})$ are F-states (i.e., states q_6, q_9, and q_{14} respectively to the order of prime-paths in $\Pi_s(\mathcal{P})$). Thus, the F-Interception condition is satisfied by $\Pi_s(\mathcal{P})$.*

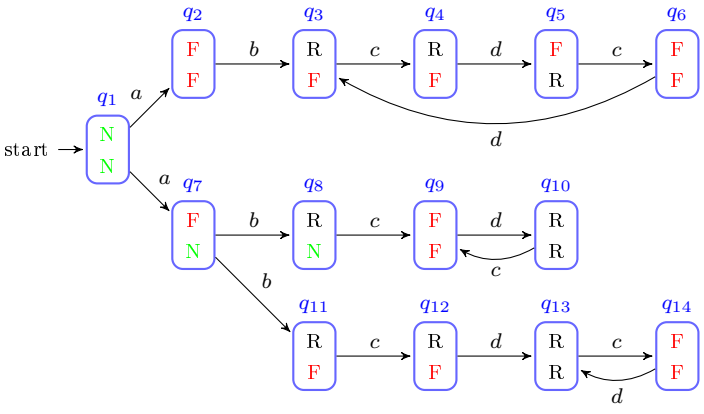


Figure 8.3 – A part of a twin-plant (Example 18)

In what follows, the *F- Interception* condition is generalized to any generated infinite path, by considering the different elementary cycles in it.

Corollary 5 *The F-Interception condition is satisfied for any generated infinite path $\wp$ in twin plant $\mathcal{P}$ if and only if the F-Interception condition is satisfied for any generated prime-path in $\mathcal{P}$.*

Proof. The proof of this corollary is straightforward. Indeed, for necessary condition ($\Rightarrow$), let us assume that all the generated infinite paths $\wp$ in $\mathcal{P}$ satisfy the *F-Interception* condition. As the generated prime-paths are also generated infinite paths, it follows directly that all the generated prime-paths in $\mathcal{P}$ satisfy the *F-Interception* condition.

For the sufficient condition ($\Leftarrow$), let us take a generated infinite path $\wp$ that contains n elementary cycles $c\ell_i$. It follows that $\mathbb{P}_\wp$ also contains n generated prime-path $\wp_i \in \mathbb{P}_\wp$ for $1 \leq i \leq n$. Let us assume that all the generated prime-paths in $\mathcal{P}$ satisfy the *F-Interception* condition. Then all the prime-paths in the ASGPP Π_{s_i} satisfy the *F-Interception* condition (s_i is the event-trace associated to $\wp_i$). Then, it follows that $\wp$ also satisfies the *F-Interception* condition. Thus, this is true for all the generated infinite paths in $\mathcal{P}$. □

It is worth noticing that the above-mentioned corollary will be used to develop the necessary and sufficient conditions for strong diagnosability by reasoning directly on the prime-paths, in the twin-plant.

Now, the necessary and sufficient condition for SF-diagnosability can be stated.

Theorem 11 *(SF-diagnosability)*

A DES model G is SF-diagnosable w.r.t projection P, class of fault events Σ_f and its corresponding class of reset events Σ_r, if and only if the F-Intersection condition is satisfied by any event-trace $s \in L(\mathcal{P})$ associated with a prime-path $\wp \in Tr(\mathcal{P})$.

Proof.

($\Rightarrow$) Assume that automaton G is SF-diagnosable and let $s \in \psi(\Sigma_f)$ be an event trace which ends with a faulty event from Σ_f. According to Definition 20, $\exists\, n \in \mathbb{N}$ s.t. $\forall t \in L(G)/s, |t| \geq n \Rightarrow (\exists t' \leq t)$, satisfying $\forall \omega \in P_L^{-1}(P(s.t') : \ell(\omega) = F$. Let us consider $s' \in \Sigma^*$ such that $s.t \leq s'$. Then, by construction of the reduced twin-plant (Definition 53), we have $\Pi_{s'}(\mathcal{P}) \neq \emptyset$.

Consider $k = |P(s.t')|$. Then, from Definition 20, we have $\forall \wp \in \Pi_{s'}(\mathcal{P}) : q_k \in \wp$ is an F-state. This applies for any $s_i \in \psi(\Sigma_f)$. Thus, it is true for any $\Pi_{s'_i}(\mathcal{P})$, with $s_i.t \leq s'_i$.

Finally, to show that $q_k \in c\ell_\wp$, by assumption (A3), the fault repeats indefinitely, it then suffices to pick t sufficiently long such that $|P(s.t')| \geq |X_o|^2$ (See [Jiang and Huang, 2001]). Thus, the *F-Interception* condition is satisfied.

($\Leftarrow$) Assume that the *F-Interception* condition is satisfied by any event-trace $s \in L(\mathcal{P})$ associated with a prime-path $\wp \in Tr(\mathcal{P})$. Then, for any infinite observable event-trace $s \in P(L(G))$ such that $\Pi_s(\mathcal{P}) \neq \emptyset$, the *F-Interception* condition states that $\forall \wp = \wp' c\ell_\wp \in \Pi_s(\mathcal{P})$, $\exists k \in \mathbb{N}$ such that $q_k \in \wp$ is an F-state. This means that $\exists\, n \in \mathbb{N}$ $(n \geq k)$ such that $\forall \omega \in \Sigma^*$, $P(\omega) = s$, then $\ell(w_n) = F$. This ensures that some occurrences of the intermittent fault in ω can be diagnosed according to Definition 20. Moreover, the *F-Interception* condition states that $q_k \in c\ell_\wp$. Then, according to assumption (A3), $\forall \omega = \omega' c\ell_\omega \in \Sigma^*$ such that $P(\omega) = s, \exists n' \in \mathbb{N} : \ell(\omega'(c\ell_\omega)^{n'}) = F$, which ensures that each occurrence of an intermittent fault in ω is diagnosable. As the *F-Interception* condition holds for any $s_i \in P(L(G))$, Then, $L(G)$ is SF-diagnosable. ■

In order to state the necessary and sufficient condition for SR-diagnosability, we introduce the *R-interception* condition, in the same way as Definition 61.

Définition 62 *(R-Interception condition)*
Let $\wp$ be a generated prime-path, and $s \in \Sigma_o^*$ its associated observable event-trace, then the R-Interception condition relatively to $\wp$ is as follows:

$\exists\, k \in \mathbb{N}$ *s.t.* $\forall \wp_i = \wp_i' c\ell_{\wp_i}^* = (q_0, q_1, \dots) \in \Pi_s(\mathcal{P})$*:*

1. *q_k is an R-state.*

2. *$q_k \in c\ell_{\wp_i}^*$.* □

Example 19 *Let us consider again the twin-plant of Example 18. For event-trace $s = (a, b, (c, d)^*)$, we have $\Pi_s(\mathcal{P}) = \{(q_1, q_2, (q_3, q_4, q_5, q_6)^*), (q_1, q_7, q_8, (q_9, q_{10})^*), (q_1, q_7, q_{11}, q_{12}, (q_{13}, q_{14})^*)\}$. One can deduce that no $k \in \mathbb{N}$ exists such that all the prime-paths in Π_s are in R-states at this step. Besides, for the primary cycle of the first generated prime-path in Π_s, i.e. $q_1, q_2, (q_3, q_4, q_5, q_6)^*$, no R-state exists. As a consequence, the $\Pi_s(\mathcal{P})$ does not satisfy the R-Interception condition.*

Theorem 12 *(SR-diagnosability)*
A DES model G is SR-diagnosable w.r.t a projection function P, a class of fault events Σ_f and its corresponding class of reset events Σ_r, if and only if the R-Intersection condition is satisfied by any event-trace $s \in L(\mathcal{P})$ associated to a prime-path $\wp \in Tr(\mathcal{P})$. □

Proof. The proof of this theorem is omitted, since it can be elaborated in an analogous way as for Theorem 11. ■

It is worth noticing that SF-diagnosability does not imply SR-diagnosability and vice versa (as witnessed through Example 18 and 19), unlike in the case of weak diagnosability.

8.3.3 Verification Algorithm

In order to check SF and SR-diagnosability based on the elaborated conditions in the previous section, we have developed an incremental depth-first search algorithm.

The algorithm consists in constructing, on the fly, twin-plant $\mathcal{P}$ corresponding to model G, while checking in parallel the necessary condition for SF-diagnosability according to Proposition 14. It is interesting to note that checking the necessary condition for the SF-diagnosability is performed by means of the same algorithm used for the WF-diagnosability (cf. Algorithm 8). The only difference is that, instead of checking the existence of an F-confused cycle, we check the existence of non-F-cycle.

If the necessary condition is satisfied (i.e., the absence of non-F-cycle), then the necessary and sufficient condition (cf. Theorem 11) is checked in an incremental way. Thus, for any event-trace s that is associated with a prime-path in $Tr(\mathcal{P})$, set $\Pi_s(\mathcal{P})$ is generated and the *F-Interception* condition is checked incrementally in $\Pi_s(\mathcal{P})$. This task is repeated until all the prime-paths in $\mathcal{P}$ have been investigated. As soon as the *F-Interception* condition is violated, G is stated to be non-SF-diagnosable. On the contrary, if no violation of the *F-Interception* condition is detected, G is stated to be SF-diagnosable. A similar algorithm is also used for checking SR-diagnosability. Hereafter, the various tasks and functions that constitute the algorithm are detailed.

Algorithm 9 is a depth-first search algorithm. It builds twin-plant $\mathcal{P}$ and verifies the necessary condition for SF-diagnosability simultaneously. As soon as a cycle is found in $\mathcal{P}$, function Check-Non-F-Cycle() checks whether such a cycle is a non-F-Cycle or not. If it is a non-F-cycle, then G is non-SF-diagnosable and the verification process is stopped. If the whole state-space of $\mathcal{P}$ has been generated, through Algorithm 9, without finding any non-F-cycle, then condition $\mathcal{C}1$ will be checked through Algorithm 10.

Algorithm 10 is a recursive algorithm that generates an arbitrary event-trace List_Event (associated with prime state-trace List_State) using function GenEventTrace(). Once an event-trace has been generated, function Check_Condition (cf. Algorithm 11) is called to perform two tasks: generating (in an incremental manner) set $\Pi_{\text{List\_Event}}$ corresponding to the generated event-trace List_Event and checking simultaneously the F-Interception condition (Definition 61). This task is repeated for any generated event-trace List_Event.

Algorithm 11, a depth-first search procedure, allows us to generate set $\Pi_{\text{List\_Event}}$ corresponding to the input event-trace List_Event (generated from Algorithm 10) and check the F-Interception condition in an incremental way.

Algorithm 9 Generating the reduced twin-plant and checking the existence of non-F-cycle

Input: $G'_\ell = (X_o, \Sigma_o, \delta_o, x_0)$, $G'_{\ell F} = (X_{oF}, \Sigma_o, \delta_{oF}, x_0)$

Output: $\mathcal{P} = G'_\ell || G'_{\ell F} = (\mathcal{Q}, \Sigma_o, \Gamma, q_0)$

1 $\Gamma \leftarrow \emptyset$, STACK $\leftarrow \emptyset$ // Initialization step

2 $\mathcal{Q} \leftarrow \{q_0\} = \{(x_0, x_0)\}$ // Creating the initial state of twin-plant

3 LIST_CURRENT $= \{(x_0, x_0)\}$

4 STACK.PUSH($\langle (x_0, x_0)$, ENABLE$(x_0) \times$ ENABLE$(x_0) \rangle$)

// Formally, ENABLE(x)= $\{\sigma \in \Sigma$ s.t. $\delta(x, \sigma) \in X\}$.

5 **while** STACK $\neq \emptyset$ **do**

6 $\langle (x, x'),$ ENABLE_SET$\rangle$ = STACK.TOP // Take the top element in the stack

7 **if** ENABLE_SET$= \emptyset$ **then**

8 $\langle (x, x'),$ ENABLE_SET$\rangle$ = STACK.POP // Remove the element from stack

9 **else**

10 (σ, σ') = REMOVELAST(ENABLE_SET) // Pick up an element

11 **if** $(\sigma = \sigma')$ **then**

// Only the shared observable event are handled

12 $x_1 \leftarrow \delta(x, \sigma)$ // Get back to the target state

13 $x'_1 \leftarrow \delta(x', \sigma')$

14 **if** $(x_1, x'_1) \notin \mathcal{Q}$ **then**

15 $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x_1, x'_1)\}$ // Update the set of states

16 $\Gamma \leftarrow \Gamma \cup \{(x, x') \xrightarrow{\sigma} (x_1, x'_1)\}$ // Update the transition function

17 LIST_CURRENT.ADD$((x_1, x'_1))$

18 STACK.PUSH($\langle (x_1, x'_1)$,ENABLE$(x_1) \times$ENABLE$(x'_1) \rangle$) // Pile up the current element

19 **else**

20 $\Gamma \leftarrow \Gamma \cup \{(x, x') \xrightarrow{\sigma} (x_1, x'_1)\}$

21 **if** $((x_1, x'_1) \in$ LIST_CURRENT$)$ **then**

22 **if** CHECK_NON-F-CYCLE*()* **then**

// Check for non-F-cycle

23 **return** "non-SF-diagnosable"

24 REMOVELAST(LIST_CURRENT)

First, a prime state-trace (noted by LIST_STATEPRIME) is generated, which is associated with event-trace LIST_EVENT. Then, LIST_STATEPRIME is compared with LIST_STATE (generated by Algorithm 10) in terms of F-state indices. We first pick from

the elementary cycle of List_State (resp. List_StatePrime), F-state indices in F-Indices (resp. F-IndicesP) using function Indices() (cf. Algorithm 10). Then, we check if List_StatePrime and List_State share some F-states indices. If no F-state index is shared (i.e., F-Indices $\cap$ F-IndicesP$= \emptyset$), then the procedure is stopped and we conclude that G is non-SF-diagnosable. Otherwise, the shared F-state indices are saved in F-Indices and the procedure is continued, to check other state-traces that have List_Event as an associated event-trace. This task is repeated for each List_Event generated by Algorithm 10.

Algorithm 10 Checking necessary and sufficient condition

Input: twin-plant $\mathcal{P}$

Output: Decision about diagnosability

```
1  q ← q0
2  List_State.Add (q0)
3  List_Event
4  F-Indices
5  GenEventTrace(G, q)
6  return 'G is SF-diagnosable'
1  Function GenEventTrace(G, q)
2      foreach σ ∈Enable(q) do
3          q' ← Γ(q, σ)
4          List_Event.Add(σ)
5          if q' ∉ List_State then
6              List_State.Add(q')
7              GenEventTrace(G, q')
8          else
9              F-Indices = Indices(List_State)
10             Check_Condition(P, F-Indices, List_Event)
                                                  // cf. Algorithm 3
11         List_State.Remove(q')
12         List_Event.Remove(σ)
```

It is worthwhile noticing that two cases are possible when comparing two prime state-traces that share the same associated event-trace in terms of F-states indices:

Algorithm 11 CHECK_CONDITION()

Input: twin-plant $\mathcal{P}$

Input: LIST_EVENT, F-INDICES

Output: Decision about non diagnosability

```
1  q ← q_0
2  index = 0, indexP = 0
3  LIST_STATEPRIME.ADD(q_0)   F-INDICESP
4  COMPUTE_Π_SET(G, q)
1  Function COMPUTE_Π_SET(G, q)
2      foreach σ ∈ENABLE(q) do
3          if (index ≤ | LIST_EVENT|) then
4              if (σ == LIST_EVENT [index]) then
5                  q' ← δ(q, σ)
6                  if (q' ∈ LIST_STATEPRIME) then
7                      if (index == | LIST_EVENT|) then
8                          F-INDICESP=INDICES(LIST_STATEPRIME)
9                          if (F-INDICES ∩ F-INDICESP = ∅) then
10                             return G is not SF-diagnosable
11                         else
12                             F-INDICES←F-INDICES∩F-INDICESP
13                 else
14                     index ← index + 1
15                     LIST_STATEPRIME.ADD(q')
16                     COMPUTE_Π_SET(G, q')
17         else
18             indexP=(| LIST_EVENT-|℘|)+(index− LIST_EVENT)_mod|℘|        // ℘:
                elementary cycle of LIST_STATE
19             if (σ == LIST_EVENT [indexP]) then
20                 q' ← δ(q, σ)
21                 if (q' ∈ LIST_STATEPRIME) then
22                     F-INDICESP = INDICES(LIST_STATEPRIME)
23                     if (F-INDICES ∩ F-INDICESP= ∅) then
24                         return G is not SF-diagnosable
25                     else
26                         F-INDICES←F-INDICES ∩F-INDICESP
27                 else
28                     index ← index + 1
29                     LIST_STATEPRIME.ADD(q')
30                     COMPUTE_Π_SET(G, q')
```

(1) *Elementary cycles of the prime state-traces have the same length:* In this case, checking F-state indices is performed on the real indices in F-INDICES and F-INDICESP (cf. Algorithm 11, Lines 6 - 12).

(2) *Elementary cycles of the prime state-traces have different lengths:* In this case, the elementary cycle, with smaller length, is extended by considering as many loops as possible to reach the same length of the other prime state-trace elementary cycle (i.e., lengths of elementary cycles in prime state-traces which share the same event-trace are multiples of the smallest cycle length). Then, checking F-state indices is performed on the new computed indices (cf. Algorithm 11, Lines 17 - 30). Finally, when all event-traces in $\mathcal{P}$ are investigated with no exit point has been encountered, then we conclude that automaton G is SF-diagnosable (cf. Algorithm 10, Line 6).

8.4 Diagnosability of Intermittent Faults as Model-Checking Problems

In the previous chapter, we have discussed the practical verification of diagnosability of permanent faults in the model-checking framework. The approach proposed is a reformulation of the twin-plant technique [Jiang and Huang, 2001]. In this work, the twin-plant structure is reformulated as a Kripke structure, while the necessary and sufficient condition is expressed as a CTL/LTL formula and tackled using a model-checking engine. The particularity of such an approach is that it can be practically implemented, applied and reused for various kinds of real-life systems. Moreover, a wide range of powerful and optimized model-checkers have been developed in the formal verification community and successfully used for the verification/validation of large scale industrial systems. In this section, we extend the approach in order to deal with the diagnosability verification of intermittent faults. That is, the necessary and sufficient conditions developed above are expressed using LTL formulas for the actual verification.

8.4.1 The Weak Diagnosability Conditions as LTL Formulas

In order to formulate the analysis issue of the weak diagnosability properties as Model-Checking problems, we first express each diagnosability condition as an LTL formula. For the sake of simplicity, we introduce these atomic propositions: N1, NF, and NR, which mean respectively: the state q in the twin plant is an $N1$-state, NF-state, NR-state.

8.4.1.1 WF-Diagnosability as a Model-Checking Problem

The LTL formula which characterizes each state of an F-confused cycle in the twin plant is,

$$\boldsymbol{\phi_1}: \ \boldsymbol{G}(\ \mathbf{N1}\ \wedge\ \boldsymbol{F}\ \mathbf{NF}\ )$$

The specification can be read as follows: "*a path from the current state in the twin-plant exists, where all states are $N1$-states and at least one state is an NF-state*". Therefore, property (N1 $\wedge F$ NF) is satisfied by each state in the cycle.

The Model-Checking problem expressing WF-diagnosability is:

$$\boldsymbol{K_{\mathcal{P}}}, \boldsymbol{S_{\mathcal{P}}} \models \neg\ \boldsymbol{F}(\ \boldsymbol{G}(\ \mathbf{N1}\ \wedge\ \boldsymbol{F}\ \mathbf{NF}\ )\ )$$

where $K_{\mathcal{P}}$ is the Kripke structure corresponding to the twin-plant $\mathcal{P}$ of model G, and $S_{\mathcal{P}}$ is the initial state in $K_{\mathcal{P}}$.

8.4.1.2 WR-Diagnosability as a Model-Checking Problem

The LTL formula that characterizes each state of an R-confused cycle in the twin plant is,

$$\boldsymbol{\phi_2}: \ \boldsymbol{G}(\ \mathbf{N1}\ \wedge\ \boldsymbol{F}\ \mathbf{NR}\ )$$

The specification can be read as follows: "*a path from the current state in the twin-plant exists, where all states are $N1$-states and at least one state is an NR-state*". Therefore, property (N1 $\wedge F$ NR) is satisfied by each state in the cycle.

The Model-Checking problem expressing WR-diagnosability is:

$$\boldsymbol{K_{\mathcal{P}}}, \boldsymbol{S_{\mathcal{P}}} \models \neg\ \boldsymbol{F}(\ \boldsymbol{G}(\ \mathbf{N1}\ \wedge\ \boldsymbol{F}\ \mathbf{NR}\ )\ )$$

8.4.2 Reformulation of the Strong Diagnosability Properties

Regarding the reformulation of the strong diagnosability properties, we notice that only the necessary conditions (presented in Section 8.3.1), can be reformulated as LTL formulas. The necessary and sufficient conditions cannot be reformulated as CTL/LTL formulas. This is due to the fact that such properties cannot be characterized by pairs of equivalent traces. They are rather global properties on sets of (observationally) equivalent traces [Fabre et al., 2016] and unfortunately CTL/LTL are no longer adequate to express such properties. We think that such properties can be dealt with via *Temporal Epistemic Logic* [Halpern and Vardi, 1989, Męski et al., 2012] which are capable to express global properties on sets of (observationally) equivalent traces [Cimatti et al., 2005, Gammie and Van Der Meyden, 2004].

8.5 Experimentation

In this section, some experimentations are performed in order to evaluate the diagnosability of intermittent faults using model-checking. We limit the experimentation to the weak diagnosability properties, since they are the only reformulation as model-checking problems. The experimentation are conducted through a benchmark that depicts the concept of intermittent faults with assumptions A1, A2, and A3 (See Section 5.5.1. For the verification, we use the symbolic model-checker NuSMV (version 2.5.4) [Cimatti et al., 2000], which is widely used for formal verification in both academia and industry.

8.5.1 Presentation of the DES Benchmark

The DES benchmark, depicted in Figure 8.4, describes a manufacturing system composed of a normal part and a faulty one. Each part contains several similar production lines modeled using a Labeled Petri Net (LPN). Many parameters can be taken into account, such as the number of tokens in place P_0, the line length, or the number of production lines. In our study, we consider only the number of production lines as a variable parameter (k). Transitions $t_1, t'_1, t_2, t'_2, t_{4,i}, t'_{4,i}$ are observable $\forall 1 \leq i \leq k$. t_1 is labeled with a, t'_1 can be labeled with a or b (we consider two tests as it will be detailed after), t_2, t'_2 are labeled with b, and $\forall 1 \leq i \leq k$, $t_{4,i}, t'_{4,i}$ are labeled with c. Transitions $t_{3.i}, t'_{3.i}, t_5, t'_5$ are unobservable $\forall 1 \leq i \leq k$. All the unobservable transitions are labeled with u excepted transition $t'_{3,1}$ and t'_5 that correspond respectively to the fault event (labeled with $f \in \Sigma_f$) and the corresponding reset event (labeled with $r \in \Sigma_r$).

As said before, two tests are performed: (Test 1) where t'_1 is labeled with a, and (Test 2) where t'_1 is labeled with b. As the benchmark is modeled by an LPN, we first generate its reachability graph with the help of TINA Tool [Berthomieu et al., 2007] and then, perform our technique based on the generated reachability graph. In order to assess the scalability, we increase the number of production lines k progressively for each test.

8.5.2 Results and Discussion

All the experiments were conducted on a 64-bit PC, Ubuntu 14.04 operating system, an Intel Core i5, 2.5 GHz Processor with 4 cores and 4 GB RAM.

Table 1 summarize the obtained results, for different number of production lines. Columns from left to right correspond to the different tests:

— k is the number of production lines;

— $|G_S|$, $|G_T|$ are respectively the number of states and the number of transitions in

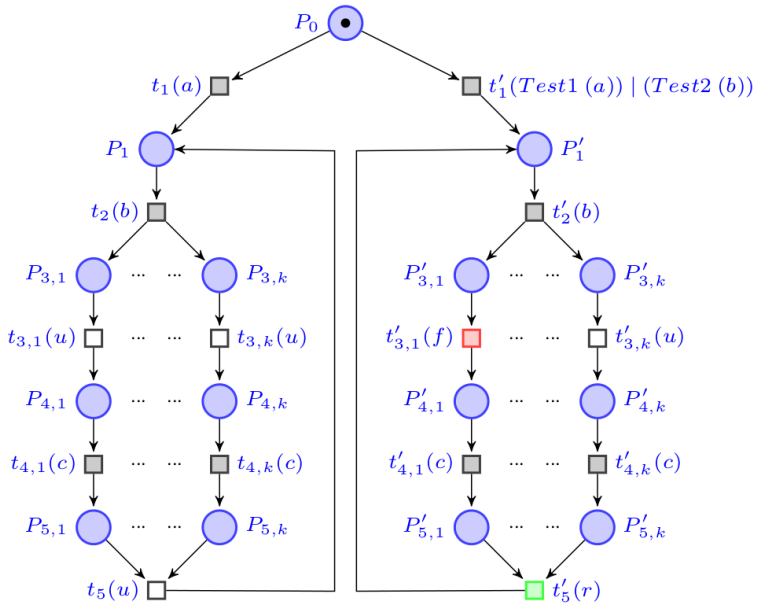


Figure 8.4 – The LPN Benchmark

automaton G (i.e., the reachability graph of the LPN model obtained using the TINA Tool);

— $|RS|$ is the number of reachable states in the Kripke structure corresponding to the twin-plant;

— T_{RS} is the time elapsed for generating the Kripke structure;

— $Diag$ is the diagnosability verdict;

— T_{Diag} : the time elapsed for verification.

8.5.2.1 Discussion

It can be seen that for *Test 1*, G is not diagnosable (not WF-diagnosable thus not WR-diagnosable). This is a logical result given the structure of the net, where the left part and the right part of the net depict the same structure (in terms of observable and unobservable transitions). Moreover, the left part is fault-free unlike the right one, which contains an intermittent fault. Thus, two executions that share the same observations exist in the twin-plant, where in the first execution an intermittent fault can occur. However, no fault event occurs in the second one. In **Test 2** (where transition t'_1 is labeled with b),

Table 8.1 – Experimental results for the LPN benchmark

	k	$\|G_S\|$	$\|G_T\|$	$\|RS\|$	T_{RS}	$Diag$	T_{Diag}
Test 1	3	131	294	2077	0.02s	No	0.02s
	4	515	1542	38285	0.17s	No	0.28s
	5	2015	7686	663453	3.04s	No	2.70s
	6	8195	36870	11048100	57s	No	361s
Test 2	3	131	294	1040	0.01s	Yes	0.02s
	4	515	1542	19144	0.17s	Yes	0.14s
	5	2015	7686	331728	4.22s	Yes	2.12s
	6	8195	36870	5524040	63s	Yes	290s

the model is WF-(and WR-)diagnosable, since there are no executions that share the same observations such that one execution contains intermittent fault f and the other is fault-free. The same reasoning can be considered for WR-diagnosability.

Regarding the scalability of the approach, we observe that the size of the reachability graph G significantly increases with the number of production lines, which affects the size of the Kripke structure that corresponding to the twin-plant. This is not surprising since, on the one hand, twin-plant computation is performed in a polynomial complexity regarding the size of model G. On the other hand, model-checking is very sensitive to the combinatorial explosion of the state space.

Finally, three remarks relative to the elapsed times for generating the twin-plant and checking diagnosability, can be emphasized:

1. The model-checker spends more time in verification than in generating the twin-plant.

2. Elapsed times for generating the twin-plant and verifying diagnosability remain in the order of milliseconds until 5 production lines, then it increases significantly. This is due to the size of the twin-plant which becomes considerable.

3. More time elapsed for verifying diagnosability when the system is diagnosable (cf. **Test 2**) than when the system is not diagnosable (cf. **Test 1**). This can be clearly observed in line 5 (**Test 1**: 2.70s $\rightarrow$ 663454 states, **Test 2**: 2.12s $\rightarrow$ 331728 states). This result is logical, since the model-checker needs to analyze the whole state-space to conclude that the system is diagnosable. However, when the system is not diagnosable, the verification process is stopped as soon as a counter-example is found, that is, only a part of the generated state-space is covered.

8.6 Discussion of the F_r-diagnosability

In Section 5.10 (Chapter 5), we have introduced the notion of F_r-diagnosability (see Definition 31), which consists in diagnosing each fault event occurrence within a finite delay, but also before its corresponding reset event occurs. In this section, we provide a necessary condition for F_r-diagnosability on the basis of the twin-plant structure. To do so, let us introduce the following twin-plant state-types

- $F\overline{F}$**-state**: is a state $q = (x, x') \in \mathcal{Q}$, such that $\Psi(q) = (F, \overline{F})$.
- F**1-state (resp.** R**1-state)**: is a state $q = (x, x') \in \mathcal{Q}$, such that $\Psi(q) = (F, \triangle)$ (resp. $\Psi(q) = (R, \triangle)$), with $\triangle \in \{N, F, R\}$.
- $\overline{F}$**1-state**: is a state $q = (x, x') \in \mathcal{Q}$, such that $\Psi(q) = (\overline{F}, \triangle)$.

with $\overline{F}$ means that the label of the corresponding state is different from F.
Now, we introduce a particular type of finite trace in the twin-plant called F_r-indicating sequence, that will be used in the sequel.

Définition 63 (***F_r-indicating sequence***)

Let $\pi = (q_1, q_2, \ldots, q_n)$ be a finite state-sequence in the twin-plant. π is an F_r-indicating sequence if q_1 is an $\overline{F}$1-state, $\forall 1 < i < n$, q_i is an $F\overline{F}$-state, and q_n $(n > 2)$ is an R1-state.

Figure 8.5 shows a path that contains a configuration of an F-indicating sequence represented by twin-plant states q_1, q_2, q_3, and q_4.

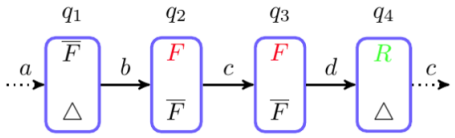


Figure 8.5 – Example of an F_r-indicating sequence

Proposition 15 (F_r-*diagnosability*)

A system model G is non-F_r-diagnosable, w.r.t a projection function P, a class of fault events Σ_f and its corresponding class of reset events Σ_r, if an F_r-indicating sequence exists in its corresponding twin-plant.

The proof of F_r-diagnosability is omitted since it can be derived directly from Proof 8.3.1.

Example 20 *Let us consider automaton G depicted in Figure 8.6. $\Sigma_o = \{a\}$, $\Sigma_u = \{f, r, u\}$, with $\Sigma_f = \{f\}$ and its corresponding reset event set $\Sigma_r = \{r\}$. Automaton $\mathcal{G}$ in Figure 8.7 is a part of the twin-plant corresponding to G, where only the composition of the top and central sequences, the bottom and top sequences, the central and the bottom sequences are represented. Let us consider finite path $\pi = q_1, a, q_2, a, q_3, a, q_4, a, q_5$ in $\mathcal{G}$. One can observe that the finite state-sequence q_3, q_4, q_5 is an F_r-indicating sequence since q_3 is an $\overline{F}1$-state, q_4 is an $F\overline{F}$-state and q_5 is an $R1$-state. Therefore, G is non-F_r-diagnosable.*

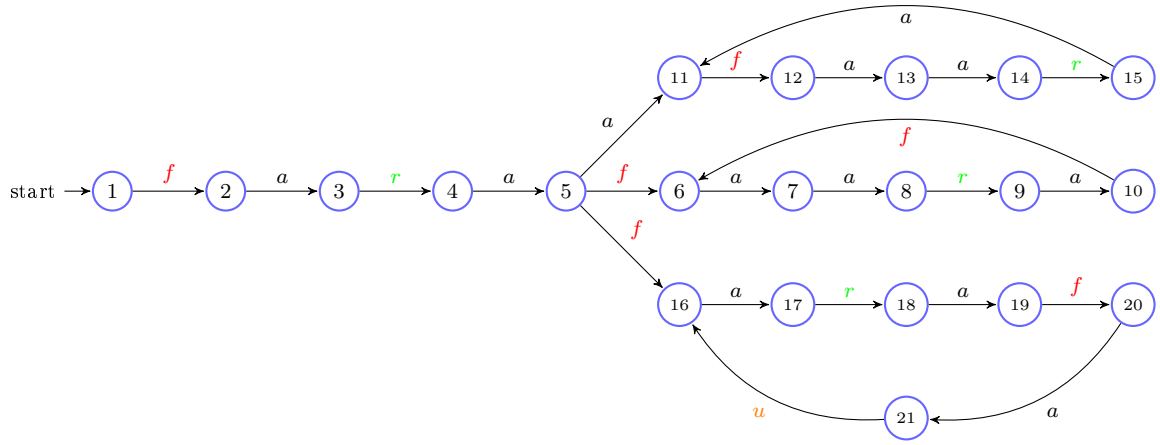


Figure 8.6 – Automaton G of Example 1

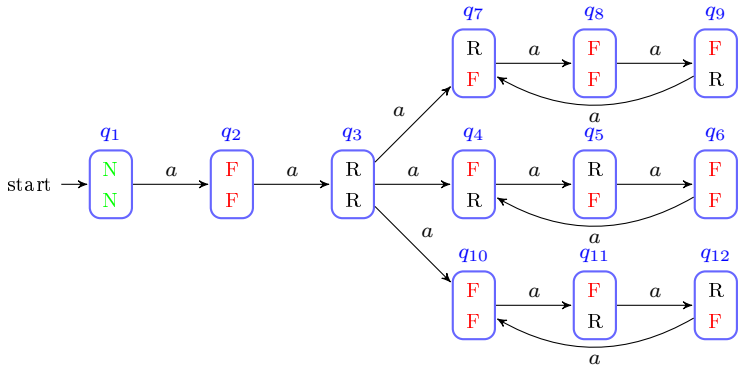


Figure 8.7 – A part of twin-plant $\mathcal{G}$ corresponding to automaton G

8.7 Conclusion

In this chapter, the twin-plant technique is investigated in order to deal with diagnosability of intermittent faults of DESs. Necessary and sufficient conditions for various notions of diagnosability properties are established and proved. Firstly, on-the-fly and incremental ad-hoc algorithms are developed to check these conditions. Secondly, the verification of (some of) these diagnosability properties is reformulated as LTL model-checking problems. Some experimentations are conducted through an LPN benchmark in order to illustrate the practicability of the developed reformulation.

Part IV

CONCLUSION

Chapter 9

Conclusions and Perspectives

Sommaire

9.1 Conclusions

This dissertation is focused on fault diagnosis of discrete-event systems modeled by finite state automata with some extensions to bounded Petri net models. Our contributions are discussed in Part II and Part III.

The second part (Part II:Chapter 4, 5 and 6) of the dissertation was devoted to fault diagnosis of DESs in the diagnoser-based context. The goal of this part is to provide some improvements to the classic diagnoser approaches.

In Chapter 4, we have developed a diagnoser variant with a new structure for representing the diagnoser nodes. Such a structure explicitly separates between the normal and the faulty states in each node. This feature allows us to separately track the normal and the faulty traces directly in the diagnoser. Moreover, the diagnoser is built directly from the original model, without needing to construct any intermediate model. On the basis of such a diagnoser structure, a necessary and sufficient condition for analyzing diagnosability is formulated and a systematic procedure for the actual verification of diagnosability, which does not require building any intermediate model is established. For the actual verification, we proposed an on-the-fly algorithm, for simultaneously constructing the diagnoser and verifying diagnosability.

In Chapter 5, we extended the diagnoser variant, introduced in Chapter 4, in order to deal with intermittent fault diagnosis. Various notions of diagnosability are discussed and necessary and sufficient condition for each notion is formulated on the basis of our diagnoser variant. In the same way as for the permanent faults, a systematic procedure for the actual verification of diagnosability properties is developed.

In Chapter 6, we established a hybrid version (in the sense of combining enumerative and symbolic representations) of the diagnoser variant that we have developed, in order to deal with fault diagnosis of bounded Petri nets. The main idea consisted in: (i) using binary decision diagrams (BDDs) to depict marking sets so as to compact the diagnoser nodes. This allows for reducing the memory consumption efficiently handling PN markings and (ii) using an explicit representation for the (observable) transitions that link the diagnoser nodes. A dedicated tool implementing the proposed approach is developed in order to assess the efficiency and the scalability of the approach. Moreover, some experimentations have been conducted through a PN benchmark. The obtained results are discussed with respect to a reference approach for fault diagnosis of LPNs, called MBRG/BRD technique [Cabasino et al., 2009a].

The third part (Part III:Chapter 7, 8) of the dissertation was dedicated to fault diagnosis of DESs through the twin-plan based technique [Jiang and Huang, 2001].

Chapter 7 discusses the actual verification of diagnosability of permanent failures using

model-checking. Namely, some extensions of the approach in [Cimatti et al., 2003] are developed to deal with the classic definition of diagnosability based on the twin-plant model. Thereafter, some quantitative versions of diagnosability, namely, $K/K_{min}-$diagnosability properties are investigated. These properties are expressed using CTL/LTL and RT-CTL specifications respectively to be tackled as model-checking problems. Some experimentations are conducted through a level crossing benchmark in order to show the applicability of the approach.

Chapter 5 deals with the diagnosability of intermittent faults. In particular, we developed necessary and sufficient conditions, for each notion of diagnosability introduced in Chapter 5 on the basis of the twin-plant approach [Jiang and Huang, 2001]. Some of the developed necessary and sufficient conditions were expressed in temporal logics (LTL/CTL) and then model-checking technique was used to perform the actual verification.

9.2 Perspectives

The work discussed in this dissertation raises several research direction as discussed below:

1) Modular diagnoser-based approach

The diagnoser-based approach we propose in the second part of this dissertation deals with *monolithic* system models for diagnosability analysis and implementation. However, in general, industrial systems are composed of several modules, local components, or subsystems that could themselves be formed by various sub-modules. Such modular systems tend to have very large state-spaces and therefore, they are complex to model and difficult to diagnose in a holistic manner, i.e., by means of a global diagnoser [Contant et al., 2006]. The challenge of modular diagnosis methodologies consists in performing diagnosis locally, i.e., at each module, while at the same time accounting for the coupling of each module with the rest of the system [Benveniste et al., 2003, Ricker and Fabre, 2000, Contant et al., 2006, Schmidt, 2013, Debouk et al., 2002b, Pandalai and Holloway, 2000]. In the near future, we intend to investigate the modular diagnosability by extending our diagnoser approach. In fact, we think that the various features of our approach, namely, the systematic procedure for the actual verification, the on-the-fly and partial construction/verification, can potentially improve and speed up the diagnosability analysis.

2) Sensor placement issues

When a system model is checked to be non-diagnosable, then two means are identified to make it diagnosable: i) *revisit sensors placement and possibly introduce new sensors* and ii) *redesign the controller*. The former arises sensor optimization/reconfiguration issues [Pan and Hashtrudi-Zad, 2007, Jiang et al., 2003a, Ru and Hadjicostis, 2010, Yoo and Lafortune, 2002a, Cabasino et al., 2013b, Debouk et al., 2002a, Dallal and Lafortune, 2011] and the synthesis of observability requirements [Bittner et al., 2012]. The diagnoser we develop in Chapter 4 has an interesting feature that can be investigate when one deals with sensor reconfiguration. As we distinguish between the normal and faulty states in each diagnoser node, it is possible to accurately identify the fault events that cause the non-diagnosablity of the system. In fact, from the various experimentations we have performed, we have seen that only one event fault is responsible for the existing of each indeterminate cycle. Moreover, such a fault event may occur either in the entering node of an indeterminate cycle or the finite trace which is linked to. These observations motivate us to explore in depth this issue.

3) Discriminating between fault types

In this dissertation, we have dealt with diagnosability of permanent and intermittent faults separately, i.e., the system model is assumed to contain either permanent faults or intermittent faults. However, in reality most of industrial systems, can show both types of failures. In such a case and from maintenance point of view, it can be critical to discriminate intermittent faults from permanent ones when a fault occurs. Namely, if the current fault is diagnosed to be an intermittent fault, a timely fault treatment actions could be required. In this way, a lot of maintenance cost can be saved by avoiding unnecessary shutdown and repair. Unfortunately, most of the existing approaches (including the contribution developed in this manuscript) consider only one type of faults, i.e., either permanent faults or intermittent faults. Dealing with this issue, i.e., discriminating intermittent from permanent faults in the same framework, is an interesting perspective that we wish to investigate.

4) An effective platform for the analysis of various system safety properties

The third part of the dissertation was devoted to the practical verification of diagnosability in a model-checking framework. In this part, we have extended the approach proposed by Cimatti et al. [Cimatti et al., 2003] to deal with various properties in fault diagnosis, namely, diagnosability, $K/K_{min}-$ diagnosability, $WF/WR-$diagnosability. As a future research direction, we intend to develop an effective platform for the analysis of vari-

ous properties related to system safety. In fact, we wish to reformulate the analysis of predictability/prognosability [Jéron et al., 2008, Takai and Kumar, 2012] and opacity [Saboori and Hadjicostis, 2007, Lin, 2011] as model-checking problems in the same way as for diagnosability analysis. A software platform, which gathers all these reformulations and integrate efficient model-checking tools, will be of a great help for the actual verification of the system safety properties. This can be the subject of some research action, for instance a research project covering these aspects.

5) Application to railway systems

This thesis was accomplished within the COSYS/ESTAS team (*Evaluation and Safety of Automated Transport Systems*) at IFSTTAR. In ESTAS, we develop methods, techniques and tools to analyze and improve the safety features of guided transport systems. Although the contributions of this thesis are mainly theoretical, as a part of our future works, we intend to bring into play the techniques developed in the thesis in order to deal with monitoring and fault diagnosis issues pertaining to embedded railway control systems. Besides, several railway case-studies have already been identified.

Bibliography

[Aghasaryan et al., 1998] Aghasaryan, A., Fabre, E., Benveniste, A., Boubour, R., and Jard, C. (1998). Fault detection and diagnosis in distributed systems: an approach by partially stochastic petri nets. *Discrete event dynamic systems*, 8(2):203–231. (Cited in page 85.)

[Akers, 1978] Akers, S. B. (1978). Binary decision diagrams. *IEEE Transactions on computers*, 100(6):509–516. (Cited in page 128.)

[Alur et al., 1990] Alur, R., Courcoubetis, C., and Dill, D. (1990). Model-checking for real-time systems. 5^{th} *Annual IEEE Symposium on Logic in Computer Science*, pages 414–425. (Cited in page 175.)

[Alur et al., 1993] Alur, R., Courcoubetis, C., and Dill, D. (1993). Model-checking in dense real-time. *Information and computation*, 104(1):2–34. (Cited in page 175.)

[Anderson and Aylward, 1993] Anderson, R. J. and Aylward, S. R. (1993). Lab testing of neural networks for improved aircraft onboard-diagnostics on flight-ready hardware. *Annual Reliability and Maintainability Symposium*, pages 404–410. (Cited in page 84.)

[Antoniotti, 1995] Antoniotti, M. (1995). *Synthesis and Verification of Discrete Controllers for Robotics and Manufacturing Devices with Temporal Logic and the Control-D System.* PhD thesis, New York University. (Cited in page 162.)

[Aydin et al., 2013] Aydin, İ., Karaköse, E., Karaköse, M., Gençoğlu, M. T., and Akın, E. (2013). A new computer vision approach for active pantograph control. *Innovations in Intelligent Systems and Applications (INISTA), 2013 IEEE International Symposium on*, pages 1–5. (Cited in page 84.)

[Baier et al., 2008] Baier, C., Katoen, J.-P., and Larsen, K. G. (2008). *Principles of model checking.* MIT press. (Cited in pages 159 and 161.)

[Bajwa et al., 2003] Bajwa, A., Sweet, A., and Korsmeyer, D. (2003). The livingstone model of a main propulsion system. (Cited in page 166.)

[Baldan et al., 2010] Baldan, P., Chatain, T., Haar, S., and König, B. (2010). Unfolding-based diagnosis of systems with an evolving topology. *Information and Computation*, 208(10):1169–1192. (Cited in page 32.)

[Ball and Hardie, 1969] Ball, M. and Hardie, F. (1969). Effects and detection of intermittent failures in digital systems. *Proceedings of the November 18-20, 1969, fall joint computer conference*, pages 329–335. (Cited in pages 6 and 83.)

[Banerjee and Khilar, 2010] Banerjee, N. and Khilar, P. (2010). Performance analysis of distributed intermittent fault diagnosis in wireless sensor networks using clustering. 5^{th} *International Conference on Industrial and Information Systems*, pages 13–18. (Cited in page 83.)

[Basile, 2014] Basile, F. (2014). Overview of fault diagnosis methods based on Petri net models. *European Control Conference (ECC)*, pages 2636–2642. (Cited in pages 24, 120, 121 and 123.)

[Basile et al., 2007] Basile, F., Chiacchio, P., and De Tommasi, G. (2007). Improving on-line fault diagnosis for discrete event systems using time. *IEEE International Conference on Automation Science and Engineering*, pages 26–32. (Cited in page 121.)

[Basile et al., 2009] Basile, F., Chiacchio, P., and De Tommasi, G. (2009). An Efficient Approach for Online Diagnosis of Discrete Event Systems. *IEEE Transactions on Automatic Control*, 54(4):748–759. (Cited in pages 24, 120 and 121.)

[Basile et al., 2010] Basile, F., Chiacchio, P., and De Tommasi, G. (2010). Diagnosability of labeled Petri nets via integer linear programming. *Discrete Event Systems*, 10(1):71–77. (Cited in pages 24, 120 and 121.)

[Basile et al., 2012a] Basile, F., Chiacchio, P., and De Tommasi, G. (2012a). On k-diagnosability of Petri nets via integer linear programming. *Automatica*, 48(9):2047–2058. (Cited in pages 24, 69, 121, 122 and 158.)

[Basile et al., 2012b] Basile, F., Chiacchio, P., and De Tommasi, G. (2012b). On K-diagnosability of Petri nets via integer linear programming. *Automatica*, 48(9):2047–2058. (Cited in pages 33 and 172.)

[Basile et al., 2008] Basile, F., Chiacchiot, P., and De Tommasi, G. (2008). Sufficient conditions for diagnosability of Petri nets. 9^{th} *International Workshop on Discrete Event Systems*, pages 370–375. (Cited in pages 24, 120 and 121.)

[Basilio and Lafortune, 2009] Basilio, J. C. and Lafortune, S. (2009). Robust codiagnosability of discrete event systems. In *2009 American Control Conference*, pages 2202–2209. IEEE. (Cited in pages 22 and 49.)

[Basu et al., 2003] Basu, S., Saha, D., Lin, Y., and Smolka, S. A. (2003). Generation of all counter-examples for Push-Down Systems. (Cited in page 175.)

[Bavishi and Chong, 1994] Bavishi, S. and Chong, E. K. (1994). Automated fault diagnosis using a discrete event systems framework. In *Intelligent Control, 1994., Proceedings of the 1994 IEEE International Symposium on*, pages 213–218. IEEE. (Cited in page 21.)

[Bellini et al., 2000] Bellini, P., Mattolini, R., and Nesi, P. (2000). Temporal logics for real-time system specification. *ACM Computing Surveys (CSUR)*, 32(1):12–42. (Cited in page 175.)

[Benveniste et al., 2003] Benveniste, A., Fabre, E., Haar, S., and Jard, C. (2003). Diagnosis of asynchronous discrete-event systems: a net unfolding approach. *IEEE Transactions on Automatic Control*, 48(5):714–727. (Cited in pages 23, 85 and 211.)

[Berthomieu et al., 2007] Berthomieu, B., Ribet, P.-O., and Vernadat, M. (2007). The tool TINA: Construction of abstract state spaces for Petri nets and time Petri nets. *International Journal of Production Research*, 42(14):2741–2756. (Cited in pages 74, 178 and 201.)

[Bertoli et al., 2007] Bertoli, P., Bozzano, M., and Cimatti, A. (2007). A symbolic model checking framework for safety analysis, diagnosis, and synthesis. *Model Checking and Artificial Intelligence*, pages 4428, 1–18. (Cited in page 167.)

[Biere et al., 1999a] Biere, A., Cimatti, A., Clarke, E., and Zhu, Y. (1999a). Symbolic model checking without bdds. In *International conference on tools and algorithms for the construction and analysis of systems*, pages 193–207. Springer. (Cited in pages 162 and 165.)

[Biere et al., 1999b] Biere, A., Cimatti, A., Clarke, E. M., Fujita, M., and Zhu, Y. (1999b). Symbolic model checking using sat procedures instead of bdds. In *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*, pages 317–320. ACM. (Cited in page 162.)

[Biere et al., 2003] Biere, A., Cimatti, A., Clarke, E. M., Strichman, O., and Zhu, Y. (2003). Bounded model checking. *Advances in computers*, 58:117–148. (Cited in pages 159 and 160.)

[Biswas, 2012] Biswas, S. (2012). Diagnosability of discrete event systems for temporary failures. *Computers & Electrical Engineering*, 38(6):1534–1549. (Cited in pages 26, 86 and 92.)

[Bittner et al., 2012] Bittner, B., Bozzano, M., Cimatti, A., and Olive, X. (2012). Symbolic synthesis of observability requirements for diagnosability. 27^{th} *Conference on Artificial Intelligence*, pages 712–718. (Cited in pages 21 and 212.)

[Bourgne et al., 2009] Bourgne, G., Dague, P., Nouioua, F., and Rapin, N. (2009). Diagnosability of Input Output Symbolic Transition Systems. 1^{st} *Int. Conference on Advances in System Testing and Validation Lifecycle*, pages 147–154. (Cited in pages 158 and 166.)

[Boussif and Ghazel, 2015a] Boussif, A. and Ghazel, M. (2015a). Diagnosability analysis of input/output discrete event system using model checking. 5^{th} *IFAC International Workshop on Dependable Control of Discrete Systems DCDS 2015*, 48(7):71 – 78. (Cited in pages 157 and 184.)

[Boussif and Ghazel, 2015b] Boussif, A. and Ghazel, M. (2015b). Une approche par décomposition de modèles pour l'analyse de la diagnosticabilité des seds par model-checking. 10^{eme} *Colloque sur la Modélisation des Systèmes Réactifs (MSR'15)*. (Cited in page 157.)

[Boussif and Ghazel, 2016a] Boussif, A. and Ghazel, M. (2016a). Analyzing various notions of diagnosability of intermittent faults in discrete event systems. *IEEE Transactions on Automation Science and Engineering (submitted)*. (Cited in page 181.)

[Boussif and Ghazel, 2016b] Boussif, A. and Ghazel, M. (2016b). Intermittent fault diagnosis of industrial systems in the model-checking framework. *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–6. (Cited in pages 81 and 181.)

[Boussif and Ghazel, 2016c] Boussif, A. and Ghazel, M. (2016c). A new variant of the diagnoser-based approach for fault diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control (submitted)*. (Cited in page 47.)

[Boussif and Ghazel, 2016d] Boussif, A. and Ghazel, M. (2016d). Using model-checking techniques for diagnosability analysis of intermittent faults -a railway case-study. *Proceedings of the* 10^{th} *International Workshop on Verification and Evaluation of Computer and Communication Systems*, pages 93–104. (Cited in pages 116 and 181.)

[Boussif and Ghazel, 2017] Boussif, A. and Ghazel, M. (2017). A diagnoser-based approach for intermittent fault diagnosis of discrete-event systems. *the 2017 American Control Conference (submitted)*. (Cited in page 81.)

[Boussif et al., 2015] Boussif, A., Ghazel, M., and Klai, K. (2015). Combining enumerative and symbolic techniques for diagnosis of discrete event systems. *in Proc. of the* 9^{th} *Int. Workshopon Verification and Evaluation of Computer and Communication Systems*. (Cited in pages 47 and 119.)

[Boussif et al., 2016a] Boussif, A., Ghazel, M., and Klai, K. (2016a). Fault diagnosis of bounded labeled petri nets using a semi-symbolic diagnoser. *IEEE Transactions on Systems, Man, and Cybernetics: Systems (submitted)*. (Cited in page 119.)

[Boussif et al., 2016b] Boussif, A., Ghazel, M., and Klai, K. (2016b). Fault diagnosis of discrete-event systems based on the symbolic observation graph. *The International Journal of Critical Computer-Based Systems (submitted)*. (Cited in pages 47 and 119.)

[Boussif et al., 2017] Boussif, A., Ghazel, M., and Klai, K. (2017). Dpn-sog: A software tool for fault diagnosis of labeled petri nets using the semi-symbolic diagnoser. 20^{th} *World Congress of the International Federation of Automatic Control, (submitted)*. (Cited in page 147.)

[Boussif et al., 2016c] Boussif, A., Liu, B., and Ghazel, M. (2016c). A twin plant based approach for diagnosability analysis of intermittent failure. 13^{th} *International Workshop on Discrete Event Systems*. (Cited in pages 81 and 181.)

[Bozzano et al., 2013a] Bozzano, M., Cimatti, A., Gario, M., and Tonetta, S. (2013a). A formal framework for the specification, verification and synthesis of diagnosers. In *AAAI (Late-Breaking Developments)*. (Cited in page 164.)

[Bozzano et al., 2013b] Bozzano, M., Cimatti, A., Gario, M., and Tonetta, S. (2013b). Formal specification and synthesis of fdi through an example. In *Workshop on Principles of Diagnosis (DX13)*, pages 174–179. (Cited in page 164.)

[Bozzano et al., 2014] Bozzano, M., Cimatti, A., Gario, M., and Tonetta, S. (2014). Formal design of fault detection and identification components using temporal epistemic logic. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 326–340. Springer. (Cited in pages 164 and 165.)

[Brusoni et al., 1998] Brusoni, V., Console, L., Terenziani, P., and Dupré, D. T. (1998). A spectrum of definitions for temporal model-based diagnosis. *Artificial Intelligence*, 102(1):39–79. (Cited in page 19.)

[Bryant, 1992] Bryant, R. E. (1992). Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Comput. Surv.*, 24(3):293–318. (Cited in pages 128 and 130.)

[Cabasino et al., 2009a] Cabasino, M. P., Giua, A., Lafortune, S., and Seatzu, C. (2009a). Diagnosability analysis of unbounded Petri nets. *Proceedings of the* 48^{th} *IEEE Conference on Decision and Control (CDC) held jointly with* 28^{th} *Chinese Control Conference*, pages 1267–1272. (Cited in pages 8, 24, 99, 120, 121 and 210.)

[Cabasino et al., 2012a] Cabasino, M. P., Giua, A., Lafortune, S., and Seatzu, C. (2012a). A new approach for diagnosability analysis of Petri nets using verifier nets. *IEEE Transactions on Automatic Control*, 57(12):3104–3117. (Cited in pages 33, 35, 69 and 158.)

[Cabasino et al., 2012b] Cabasino, M. P., Giua, A., Marcias, L., and Seatzu, C. (2012b). A comparison among tools for the diagnosability of discrete event systems. In *IEEE International Conference on Automation Science and Engineering*, pages 218–223. IEEE. (Cited in page 147.)

[Cabasino et al., 2013a] Cabasino, M. P., Giua, A., Paoli, A., and Seatzu, C. (2013a). Decentralized diagnosis of discrete-event systems using labeled Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–9. (Cited in pages 22, 49 and 158.)

[Cabasino et al., 2009b] Cabasino, M. P., Giua, A., and Seatzu, C. (2009b). Diagnosability of bounded Petri nets. *Proceedings of the* 48^{th} *IEEE Conference on Decision and Control, held jointly with the* 28^{th} *Chinese Control Conference*, pages 1254–1260. (Cited in pages 5, 24, 49, 120, 122, 139, 152 and 158.)

[Cabasino et al., 2010] Cabasino, M. P., Giua, A., and Seatzu, C. (2010). Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 46(9):1531–1539. (Cited in pages 24, 120 and 122.)

[Cabasino et al., 2014] Cabasino, M. P., Giua, A., and Seatzu, C. (2014). Diagnosability of discrete-event systems using labeled petri nets. *IEEE Transactions on Automation Science and Engineering*, 11(1):144–153. (Cited in pages 49, 68, 124, 133, 135, 147, 148 and 158.)

[Cabasino et al., 2013b] Cabasino, M. P., Lafortune, S., and Seatzu, C. (2013b). Optimal sensor selection for ensuring diagnosability in labeled Petri nets. *Automatica*, 49(8):2373–2383. (Cited in pages 21 and 212.)

[Carvalho et al., 2012] Carvalho, L., Basilio, J., and Moreira, M. V. (2012). Robust diagnosis of discrete event systems against intermittent loss of observations. *Automatica*, 48(9):2068–2078. (Cited in page 90.)

[Carvalho et al., 2010] Carvalho, L. K., Basilio, J. C., and Moreira, M. V. (2010). Robust diagnosability of discrete event systems subject to intermittent sensor failures. *International Workshop on Discrete Event Systems*, pages 84–89. (Cited in pages 86 and 90.)

[Carvalho et al., 2013] Carvalho, L. K., Basilio, J. C., Moreira, M. V., and Clavijo, L. B. (2013). Diagnosability of intermittent sensor faults in discrete event systems. *American Control Conference*, pages 929–934. (Cited in pages 86 and 90.)

[Cassandras and Lafortune, 2009] Cassandras, C. and Lafortune, S. (2009). Introduction to discrete event systems. *Springer Science and Business Media*. (Cited in pages 3, 20, 26, 27, 29, 131 and 244.)

[Cassez, 2009] Cassez, F. (2009). A note on fault diagnosis algorithms. *Proceedings of the* 48^{th} *IEEE Conference on Decision and Control, held jointly with the* 28^{th} *Chinese Control Conference. CDC/CCC*, pages 6941–6946. (Cited in pages 31, 35 and 36.)

[Cassez and Tripakis, 2008] Cassez, F. and Tripakis, S. (2008). Fault diagnosis with static and dynamic observers. *Fundamenta Informaticae*, 88(4):497–540. (Cited in pages 31, 35, 36 and 166.)

[Chandra and Kumar, 2002] Chandra, V. and Kumar, R. (2002). A event occurrence rules based compact modeling formalism for a class of discrete event systems. *Mathematical and computer modelling of dynamical Systems*, 8(1):49–73. (Cited in page 163.)

[Chang and McCluskey, 1997] Chang, J. T. Y. and McCluskey, E. J. (1997). Detecting bridging faults in dynamic cmos circuits. *IEEE International Workshop on IDDQ Testing*, pages 106–109. (Cited in page 84.)

[Chanthery and Pencolé, 2009] Chanthery, E. and Pencolé, Y. (2009). Monitoring and active diagnosis for discrete-event systems. *IFAC Proceedings Volumes*, 42(8):1545–1550. (Cited in page 21.)

[Chédor et al., 2015] Chédor, S., Morvan, C., Pinchinat, S., and Marchand, H. (2015). Diagnosis and opacity problems for infinite state systems modeled by recursive tile systems. *Discrete Event Dynamic Systems*, 25(1-2):271–294. (Cited in page 31.)

[Chen and Patton, 2012] Chen, J. and Patton, R. J. (2012). *Robust model-based fault diagnosis for dynamic systems*, volume 3. Springer Science & Business Media. (Cited in pages 17, 19 and 20.)

[Chen et al., 2014] Chen, Z., Lin, F., Wang, C., Le Wang, Y., and Xu, M. (2014). Active diagnosability of discrete event systems and its application to battery fault diagnosis. *IEEE Transactions on control systems technology*, 22(5):1892–1898. (Cited in page 21.)

[Chung, 2005] Chung, S.-L. (2005). Diagnosing PN-based models with partial observable transitions. *International Journal of Computer Integrated Manufacturing*, 18(2-3):158–169. (Cited in pages 24, 120 and 122.)

[Cimatti et al., 2000] Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M. (2000). NUSMV: a new symbolic model checker. *Int. Journal on Software Tools for Technology Transfer*. (Cited in pages 175, 176 and 201.)

[Cimatti et al., 2003] Cimatti, A., Pecheur, C., and Cavada, R. (2003). Formal verification of diagnosability via symbolic model checking. *Int. Joint Conference on Artificial Intelligence*.

(Cited in pages 6, 7, 9, 22, 120, 121, 158, 159, 165, 166, 167, 168, 169, 170, 171, 173, 182, 211, 212 and 246.)

[Cimatti et al., 2005] Cimatti, A., Pecheur, C., and Lomuscio, A. (2005). Applications of model checking for multi-agent systems: Verification of diagnosability and recoverability. In *Proceedings of the International Workshop on Concurrency, Specification, and Programming (CS&P 2005). Ruciane-Nida.* (Cited in pages 164 and 200.)

[Clarke et al., 2001] Clarke, E., Grumberg, O., Jha, S., Lu, Y., and Veith, H. (2001). Progress on the state explosion problem in model checking. In *Informatics*, pages 176–194. Springer. (Cited in page 162.)

[Clarke et al., 1996] Clarke, E., McMillan, K., Campos, S., and Hartonas-Garmhausen, V. (1996). Symbolic model checking. In *International Conference on Computer Aided Verification*, pages 419–422. Springer. (Cited in page 162.)

[Clarke and Emerson, 1981] Clarke, E. M. and Emerson, E. A. (1981). Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pages 52–71. Springer. (Cited in pages 159 and 161.)

[Clarke et al., 1999] Clarke, E. M., Grumberg, O., and Peled, D. A. (1999). *Model Checking*. MIT Press. (Cited in page 159.)

[Contant, 2005] Contant, O. (2005). On monitoring and diagnosing classes of discrete event systems. *PhD Thesis, University of Michigan.* (Cited in pages 6, 85, 90, 94 and 95.)

[Contant et al., 2002] Contant, O., Lafortune, S., and Teneketzis, D. (2002). Failure diagnosis of discrete event system: The case of intermittent faults. *International Conference on Decision and Control*, pages 4006–4017. (Cited in pages 85 and 90.)

[Contant et al., 2004] Contant, O., Lafortune, S., and Teneketzis, D. (2004). Diagnosis of Intermittent Faults. *Discrete Event Dynamic Systems*, 14(2):171–202. (Cited in pages 84, 85, 86, 88, 89, 90, 91, 98, 99, 104, 105, 106, 109, 112, 113 and 114.)

[Contant et al., 2006] Contant, O., Lafortune, S., and Teneketzis, D. (2006). Diagnosability of discrete event systems with modular structure. *Discrete Event Dynamic Systems*, 16(1):9–37. (Cited in pages 23, 49, 158 and 211.)

[Cook, 1971] Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM. (Cited in page 165.)

[Cordier et al., 2004] Cordier, M.-O., Dague, P., Lévy, F., Montmain, J., Staroswiecki, M., and Travé-Massuyès, L. (2004). Conflicts versus analytical redundancy relations: a comparative analysis of the model based diagnosis approach from the artificial intelligence and automatic control perspectives. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2163–2177. (Cited in pages 4 and 19.)

[Cordier and Largouet, 2001] Cordier, M. O. and Largouet, C. (2001). Using model-checking techniques for diagnosing discrete-event systems. 12^{th} *Int. Workshop on Principles of Diagnosis.* (Cited in page 163.)

[Correcher et al., 2003] Correcher, A., Garcia, E., Morant, F., and Quiles, E. (2003). Intermittent failure diagnosis in industrial processes. *IEEE International Symposium on Industrial Electronics*, 2:723–728. (Cited in pages 83 and 86.)

[Correcher et al., 2010] Correcher, A., García, E., Quiles, E., Morant, F., and Rodríguez, L. (2010). Diagnosis of intermittent faults and its dynamics. *INTECH Open Access Publisher.* (Cited in page 83.)

[Dallal and Lafortune, 2010] Dallal, E. and Lafortune, S. (2010). On most permissive observers in dynamic sensor optimization problems for discrete event systems. 48^{th} *Annual Allerton Conference on Communication, Control, and Computing*, pages 318–324. (Cited in pages 33 and 69.)

[Dallal and Lafortune, 2011] Dallal, E. and Lafortune, S. (2011). Efficient computation of most permissive observers in dynamic sensor activation problems. *Int. Workshop on Logical Aspects of Fault-Tolerance*, pages 1–28. (Cited in pages 21, 33, 34, 172 and 212.)

[Darwiche and Provan, 1996] Darwiche, A. and Provan, G. (1996). Exploiting system structure in model-based diagnosis of discrete-event systems. In *Proceedings of the Seventh International Workshop on Principles of Diagnosis*, pages 95–105. (Cited in pages 19 and 163.)

[De Kleer, 2009] De Kleer, J. (2009). Diagnosing multiple persistent and intermittent faults. *The International Joint Conference on Artificial Intelligence*, pages 733–738. (Cited in page 87.)

[de Kleer and Kurien, 2004] de Kleer, J. and Kurien, J. (2004). Fundamentals of model-based diagnosis. In *Proc. 5th IFAC Symposium on Fault Detection, Supervision, and Safety of Technical Processes (Safeprocess)*, pages 25–36. (Cited in page 20.)

[De Kleer et al., 2008] De Kleer, J., Price, B., Kuhn, L., Do, M., and Zhou, R. (2008). A framework for continuously estimating persistent and intermittent failure probabilities. *19th International Workshop on Principles of Diagnosis*, pages 63–70. (Cited in page 87.)

[De Nicola and Vaandrager, 1990] De Nicola, R. and Vaandrager, F. (1990). Action versus state based logics for transition systems. In *Semantics of Systems of Concurrent Processes*, pages 407–419. Springer. (Cited in page 160.)

[De Nicola and Vaandrager, 1995] De Nicola, R. and Vaandrager, F. (1995). Three logics for branching bisimulation. *Journal of the ACM (JACM)*, 42(2):458–487. (Cited in page 160.)

[De Vries, 1990] De Vries, R. C. (1990). An automated methodology for generating a fault tree. *IEEE transactions on reliability*, 39(1):76–86. (Cited in page 18.)

[Debouk et al., 1998] Debouk, R., Lafortune, S., and Teneketzis, D. (1998). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *IEEE International Conference on Systems, Man, and Cybernetics*, 3:3010–3011. (Cited in pages 22, 49 and 158.)

[Debouk et al., 2000] Debouk, R., Lafortune, S., and Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems*, 10(1-2):33–86. (Cited in pages 22 and 49.)

[Debouk et al., 2002a] Debouk, R., Lafortune, S., and Teneketzis, D. (2002a). On an optimization problem in sensor selection. *Discrete Event Dynamic Systems*. (Cited in pages 21, 49 and 212.)

[Debouk et al., 2002b] Debouk, R., Malik, R., and Brandin, B. (2002b). A modular architecture for diagnosis of discrete event systems. *Proceedings of the* 41^{st} *IEEE Conference on Decision and Control*, 1:417–422. (Cited in pages 23, 158 and 211.)

[Deng et al., 2014] Deng, G., Qiu, J., Liu, G., and Lyu, K. (2014). A discrete event systems approach to discriminating intermittent from permanent faults. *Chinese Journal of Aeronautics*, 27(2):390–396. (Cited in pages 84 and 87.)

[Deshpande and Varaiya, 1996] Deshpande, A. and Varaiya, P. (1996). Semantic tableau for control of PLTL formulae. In *Decision and Control, 1996., Proceedings of the 35th IEEE Conference on*, volume 2, pages 2243–2248. IEEE. (Cited in page 162.)

[Dotoli et al., 2009] Dotoli, M., Fanti, M. P., Mangini, A. M., and Ukovich, W. (2009). On-line fault detection in discrete event systems by Petri nets and integer linear programming. *Automatica*, 45(11):2665–2672. (Cited in pages 24 and 121.)

[Eén and Sörensson, 2003] Eén, N. and Sörensson, N. (2003). An extensible sat-solver. In *International conference on theory and applications of satisfiability testing*, pages 502–518. Springer. (Cited in page 165.)

[Elliott, 1994] Elliott, M. S. (1994). Computer-assisted fault-tree construction using a knowledge-based approach. *IEEE Transactions on Reliability*, 43(1):112–120. (Cited in page 18.)

[Emerson, 1990] Emerson, E. A. (1990). Temporal and modal logic. *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, 995(1072):5. (Cited in page 164.)

[Emerson, 2008] Emerson, E. A. (2008). The beginning of model checking: a personal perspective. In *25 Years of Model Checking*, pages 27–45. Springer. (Cited in page 162.)

[Eser Kart and Schmidt, 2015] Eser Kart, B. and Schmidt, K. (2015). Computation of reduced diagnosers for the fault diagnosis of discrete event systems. *Engineering and Technology Symposium*, pages 1–6. (Cited in page 35.)

[Fabre et al., 2002] Fabre, E., Benveniste, A., and Jard, C. (2002). Distributed diagnosis for large discrete event dynamic systems. *15th IFAC World Congress, Barcelona*. (Cited in page 23.)

[Fabre et al., 2016] Fabre, E., Hélouet, L., Lefaucheux, E., and Marchand, H. (2016). Diagnosability of repairable faults. 13^{th} *International Workshop on Discrete Event Systems (WODES)*, pages 230–236. (Cited in pages 29, 87, 90, 116 and 200.)

[Farmakis and Moskowitz, 2013] Farmakis, I. and Moskowitz, M. A. (2013). *Fixed point theorems and their applications*. World Scientific. (Cited in page 61.)

[Frank, 1996] Frank, P. (1996). Analytical and qualitative model-based fault diagnosis: A survey and some new results. *European Journal of Control*, 2(1):6 – 28. (Cited in page 19.)

[Fromherz et al., 2004] Fromherz, M. P. J., Bobrow, D. G., and de Kleer, J. (2004). Model-based computing for design and control of reconfigurable systems. *AI Magazine*, 24(4):120–130. (Cited in pages 6 and 83.)

[Gammie and Van Der Meyden, 2004] Gammie, P. and Van Der Meyden, R. (2004). MCK: Model checking the logic of knowledge. In *International Conference on Computer Aided Verification*, pages 479–483. Springer. (Cited in pages 164 and 200.)

[García et al., 2005] García, E., Correcher, A., Morant, F., Quiles, E., and Blasco, R. (2005). Modular fault diagnosis based on discrete event systems. *Discrete Event Dynamic Systems*, 15(3):237–256. (Cited in pages 23 and 49.)

[Garcia et al., 2002] Garcia, E., Morant, F., Blasco-Giménez, R., Correcher, A., and Quiles, E. (2002). Centralized modular diagnosis and the phenomenon of coupling. In *Discrete Event*

Systems, 2002. Proceedings. Sixth International Workshop on, pages 161–168. IEEE. (Cited in page 23.)

[Garcia and Yoo, 2005] Garcia, H. E. and Yoo, T.-S. (2005). Model-based detection of routing events in discrete flow networks. *Automatica*, 41(4):583–594. (Cited in pages 85 and 116.)

[Gario, 2016] Gario, M. E. G. (2016). A formal foundation of fdi design via temporal epistemic logic. *PhD Thesis*. (Cited in pages 18 and 164.)

[Genc and Lafortune, 2003] Genc, S. and Lafortune, S. (2003). Distributed diagnosis of discrete-event systems using petri nets. In *International Conference on Application and Theory of Petri Nets*, pages 316–336. Springer. (Cited in page 23.)

[Germanos et al., 2015] Germanos, V., Haar, S., Khomenko, V., and Schwoon, S. (2015). Diagnosability under weak fairness. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(4):69. (Cited in pages 24, 120 and 121.)

[Gertler, 1991] Gertler, J. (1991). Analytical redundancy methods in fault detection and isolation. In *Preprints of IFAC/IMACS Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS91*, pages 9–21. (Cited in page 19.)

[Ghazel and El-Koursi, 2014] Ghazel, M. and El-Koursi, E.-M. (2014). Two-half-barrier level crossings versus four-half-barrier level crossings: A comparative risk analysis study. *IEEE Transactions on Intelligent Transportation Systems*, 15(3):1123–1133. (Cited in page 178.)

[Ghazel and Liu, 2016] Ghazel, M. and Liu, B. (2016). A customizable railway benchmark to deal with fault diagnosis issues in des. *2016 13th International Workshop on Discrete Event Systems (WODES)*, pages 177–182. (Cited in pages 176 and 178.)

[Giua, 2008] Giua, A. (2008). A benchmark for diagnosis. *9th International Workshop on Discrete Event Systems*. (Cited in page 72.)

[Grastien, 2009] Grastien, A. (2009). Symbolic testing of diagnosability. *DX'09*. (Cited in pages 22, 42 and 158.)

[Grastien et al., 2008] Grastien, A. et al. (2008). Incremental diagnosis of des by satisfiability. In *Proceedings of the* 18^{th} *European Conference on Artificial Intelligence*, pages 787–788. IOS Press. (Cited in page 165.)

[Grastien et al., 2007] Grastien, A., Rintanen, J., and Kelareva, E. (2007). Modeling and solving diagnosis of discrete-event systems via satisfiability. In *Eighteenth International Workshop on Principles of Diagnosis–DX*, volume 7, pages 114–121. (Cited in page 165.)

[Grastien and Anbulagan, 2010] Grastien, Al. and Anbulagan, An. (2010). Diagnostic de systèmes à événements discrets à base de cohérence par SAT. *Revue d'Intelligence Artificielle (RIA)*, 24(6):757–786. (Cited in page 165.)

[Grastien and Anbulagan, 2013] Grastien, Al. and Anbulagan, An. (2013). Diagnosis of discrete event systems using satisfiability algorithms: a theoretical and empirical study. *IEEE Transactions on Automatic Control (TAC)*, 58(12):3070–3083. (Cited in page 165.)

[Haddad et al., 2004] Haddad, S., Ilié, J. M., Klai, K., and Wang, F. (2004). Design and evaluation of a symbolic and abstraction-based model checker. 2^{nd} *International Symposium on Automated Technology for Verification and Analysis (ATVA'04)*, pages 198–210. (Cited in pages 123, 127, 130, 131 and 147.)

[Halpern and Vardi, 1989] Halpern, J. Y. and Vardi, M. Y. (1989). The complexity of reasoning about knowledge and time. i. lower bounds. *Journal of Computer and System Sciences*, 38(1):195–237. (Cited in pages 164 and 200.)

[Hamscher, 1992] Hamscher, W. (1992). Readings in model-based diagnosis. (Cited in pages 18 and 19.)

[Hardie and Suhocki, 1967] Hardie, F. H. and Suhocki, R. J. (1967). Design and use of fault simulation for saturn computer design. *IEEE Transactions on Electronic Computers*, (4):412–429. (Cited in pages 6 and 83.)

[Henzinger et al., 1994] Henzinger, T. A., Nicollin, X., Sifakis, J., and Yovine, S. (1994). Symbolic model checking for real-time systems. *Information and computation*, 111(2):193–244. (Cited in page 175.)

[Holloway and Chand, 1994] Holloway, L. E. and Chand, S. (1994). Time templates for discrete event fault monitoring in manufacturing systems. In *American Control Conference, 1994*, volume 1, pages 701–706. IEEE. (Cited in page 23.)

[Hosseini et al., 2013] Hosseini, M., Lennartson, B., Cabasino, M., and Seatzu, C. (2013). A survey on efficient diagnosability tests for automata and bounded petri nets. *IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA),*, pages 1–6. (Cited in pages 42, 72 and 158.)

[Huang, 2003] Huang, Z. (2003). Rules based modeling of discrete event systems with faults and their diagnosis. *PhD Thesis*. (Cited in pages 18, 20, 83, 164 and 246.)

[Huang et al., 2004] Huang, Z., Bhattacharyya, S., Chandra, V., Jiang, S., and Kumar, R. (2004). Diagnosis of discrete event systems in rules-based model using first-order linear temporal logic. In *American Control Conference, 2004. Proceedings of the 2004*, volume 6, pages 5114–5119. IEEE. (Cited in page 164.)

[Hughes and Cresswell, 1996] Hughes, G. E. and Cresswell, M. J. (1996). *A new introduction to modal logic*. Psychology Press. (Cited in page 164.)

[Isermann, 2006] Isermann, R. (2006). *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media. (Cited in pages 16 and 82.)

[Isermann and Ballé, 1997] Isermann, R. and Ballé, P. (1997). Trends in the application of model-based fault detection and diagnosis of technical processes. *Control engineering practice*, 5(5):709–719. (Cited in pages 16 and 19.)

[Ismaeel and Bhatnagar, 1997] Ismaeel, A. A. and Bhatnagar, R. (1997). Test for detection and location of intermittent faults in combinational circuits. *IEEE transactions on reliability*, 46(2):269–274. (Cited in page 84.)

[Jéron et al., 2008] Jéron, T., Marchand, H., Genc, S., and Lafortune, S. (2008). Predictability of sequence patterns in discrete event systems. *{IFAC} Proceedings Volumes*, 41(2):537 – 543. (Cited in page 213.)

[Jéron et al., 2006] Jéron, T., Marchand, H., Pinchinat, S., and Cordier, M.-O. (2006). Supervision patterns in discrete event systems diagnosis. *Discrete Event Systems, 2006 8th International Workshop on*, pages 262–268. (Cited in pages xiii, 31, 33, 36, 87, 91 and 166.)

[Jiang, 2002] Jiang, S. (2002). Supervisory control and failure diagnosis of discrete event systems: A temporal logic approach. *PhD Thesis*. (Cited in pages 85 and 163.)

[Jiang and Huang, 2001] Jiang, S. and Huang, Z. (2001). A polynomial algorithm for testing diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321. (Cited in pages 5, 7, 22, 36, 40, 48, 120, 121, 158, 164, 165, 170, 182, 183, 184, 186, 193, 199, 210, 211 and 246.)

[Jiang and Kumar, 2004] Jiang, S. and Kumar, R. (2004). Failure diagnosis of discrete-event systems with linear-time temporal logic specifications. *IEEE Transactions on Automatic Control,*, 49(6):934–945. (Cited in pages 85, 163 and 171.)

[Jiang and Kumar, 2006] Jiang, S. and Kumar, R. (2006). Diagnosis of repeated failures for discrete event systems with linear-time temporal-logic specifications. *IEEE Transactions on Automation Science and Engineering*, 3(1):47–59. (Cited in pages 85 and 163.)

[Jiang et al., 2003a] Jiang, S., Kumar, R., and Garcia, H. (2003a). Optimal sensor selection for discrete-event systems with partial observation. *IEEE Transactions on Automatic Control*, 48(3):369–381. (Cited in pages 21 and 212.)

[Jiang et al., 2003b] Jiang, S., Kumar, R., and Garcia, H. E. (2003b). Diagnosis of repeated / intermittent failures in discrete event systems. *IEEE Transactions on Robotics and Automation*, 19(2):1–27. (Cited in pages 85, 116 and 163.)

[Jiroveanu and Boel, 2004] Jiroveanu, G. and Boel, R. (2004). Contextual distributed diagnosis for very large systems. 7^{th} *IFAC Workshop on Discrete Event Systems*, pages 343–348. (Cited in pages 24, 120 and 122.)

[Jiroveanu and Boel, 2010] Jiroveanu, G. and Boel, R. K. (2010). The diagnosability of Petri net models using minimal explanations. *IEEE Transactions on Automatic Control*, 55(7):1663–1668. (Cited in pages 24, 120 and 121.)

[Jiroveanu et al., 2008] Jiroveanu, G., Boel, R. K., and Bordbar, B. (2008). On-line monitoring of large Petri net models under partial observation. *Discrete Event Dynamic Systems*, 18(3):323–354. (Cited in page 49.)

[Kautz and Selman, 1996] Kautz, H. and Selman, B. (1996). Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1194–1201. (Cited in page 165.)

[Kim, 2009] Kim, C. J. (2009). Electromagnetic radiation behavior of low-voltage arcing fault. *IEEE Transactions on Power Delivery*, 24(1):416–423. (Cited in page 84.)

[Klai and Petrucci, 2008] Klai, K. and Petrucci, L. (2008). Modular construction of the symbolic observation graph. *The* 8^{th} *International Conference on Application of concurrency to System Design*, pages 23–27. (Cited in pages 123, 131 and 147.)

[Kleer, 1992] Kleer, J. D. (1992). Diagnosing Multiple Persistent and Intermittent Faults. pages 733–738. (Cited in page 19.)

[Knight, 2002] Knight, J. C. (2002). Safety critical systems: challenges and directions. In *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, pages 547–550. IEEE. (Cited in page 2.)

[Kumar et al., 1993] Kumar, R., Garg, V., and Marcus, S. I. (1993). Predicates and predicate transformers for supervisory control of discrete event dynamical systems. *IEEE Transactions on Automatic Control*, 38(2):232–247. (Cited in page 163.)

[Kumar and Takai, 2010] Kumar, R. and Takai, S. (2010). Decentralized prognosis of failures in discrete event systems. *IEEE Transactions on Automatic Control*, 55(1):48–59. (Cited in pages 22 and 49.)

[Kupferman and Vard, 2001] Kupferman, O. and Vard, M. Y. (2001). Model checking of safety properties. *Formal Methods in System Design*. (Cited in page 175.)

[Lafortune, 2000] Lafortune, S. (2000). UMDES Software library. *http://wiki.eecs.umich.edu/umdes/index.php /UMDES Group*. (Cited in page 72.)

[Lafortune et al., 2005] Lafortune, S., Wang, Y., and Yoo, T.-S. (2005). Diagnostic décentralisé des systèmes à événements discrets. *Journal Européen des Systèmes Automatisés*, 39(1-3):95–110. (Cited in pages 22, 49 and 158.)

[Lamperti and Zanella, 2013] Lamperti, G. and Zanella, M. (2013). *Diagnosis of active systems: principles and techniques*, volume 741. Springer Science & Business Media. (Cited in page 19.)

[Lapp and Powers, 1977] Lapp, S. A. and Powers, G. J. (1977). Computer-aided synthesis of fault-trees. *IEEE Transactions on Reliability*, 1:2–13. (Cited in page 18.)

[Larsen et al., 1995] Larsen, K. G., Pettersson, P., and Yi, W. (1995). Model-checking for real-time systems. *International Symposium on Fundamentals of Computation Theory*, pages 62–88. (Cited in page 175.)

[Lee et al., 1985] Lee, W.-S., Grosh, D. L., Tillman, F. A., and Lie, C. H. (1985). Fault tree analysis, methods, and applications – a review. *IEEE transactions on reliability*, 34(3):194–203. (Cited in page 18.)

[Lefebvre, 2014] Lefebvre, D. (2014). On-line fault diagnosis with partially observed petri nets. *IEEE Transactions on Automatic Control*, 59(7):1919–1924. (Cited in page 120.)

[Lefebvre and Delherm, 2007] Lefebvre, D. and Delherm, C. (2007). Diagnosis of des with petri net models. *IEEE Transactions on Automation Science and Engineering*, 4(1):114–118. (Cited in page 120.)

[Lefebvre and Leclercq, 2015] Lefebvre, D. and Leclercq, E. (2015). Diagnosability of petri nets with observation graphs. *Discrete Event Dynamic Systems*, pages 1–21. (Cited in pages 120 and 122.)

[Lefebvre et al., 2013] Lefebvre, D., Leclercq, E., and Guerin, F. (2013). Design of observations graphs for partially observed petri nets: Application to the diagnosability analysis of des1. In *52nd IEEE Conference on Decision and Control*, pages 6329–6334. IEEE. (Cited in pages 120 and 122.)

[Li et al., 2015a] Li, B., Guo, T., Zhu, X., and Li, Z. (2015a). Reverse twin plant for efficient diagnosability testing and optimizing. *Engineering Applications of Artificial Intelligence*, 38:131–137. (Cited in page 5.)

[Li et al., 2015b] Li, B., Khlif-Bouassida, M., and Toguyéni, A. (2015b). On-the-fly diagnosability analysis of labeled Petri nets using T-invariants. 5^{th} *IFAC International Workshop on Dependable Control of Discrete Systems*, 48(7):64–70. (Cited in pages 24 and 123.)

[Li et al., 2015c] Li, B., Liu, B., and Toguyeni, A. (2015c). On-the-fly diagnosability analysis of labeled Petri nets using minimal explanations. 9^{th} *IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes*, 48(21):326–331. (Cited in pages 24, 49, 120, 123 and 158.)

[Lin, 1991] Lin, F. (1991). Analysis and synthesis of discrete event systems using temporal logic. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 140–145. IEEE. (Cited in page 162.)

[Lin, 1994] Lin, F. (1994). Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems*, 4(2):197–212. (Cited in pages 3, 19, 21, 22, 26, 27, 48, 120, 158 and 166.)

[Lin, 2011] Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, 47(3):496–503. (Cited in page 213.)

[Lin and Wonham, 1988] Lin, F. and Wonham, W. M. (1988). On observability of discrete-event systems. *Information sciences*, 44(3):173–198. (Cited in page 28.)

[Ling and Ionescu, 1994] Ling, J.-Y. and Ionescu, D. (1994). A reachability synthesis procedure for discrete event systems in a temporal logic framework. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(9):1397–1406. (Cited in page 162.)

[Liu, 2014] Liu, B. (2014). An efficient approach for diagnosability and diagnosis of des based on LPN in untimed and timed contexts. *PhD Thesis, Ecole Centrale de Lille.* (Cited in pages 5, 24, 32, 33, 35, 43, 48, 49, 122, 124, 139, 158, 178 and 179.)

[Liu et al., 2014a] Liu, B., Ghazel, M., and Toguyéni, A. (2014a). OF-PENDA: A software tool for fault diagnosis of discrete event systems modeled by labeled Petri nets. *International Worshop Petri Nets for Adaptive Discrete-Event Control Systems.* (Cited in page 178.)

[Liu et al., 2014b] Liu, B., Ghazel, M., and Toguyéni, A. (2014b). Toward an efficient approach for diagnosability analysis of DES modeled by labeled Petri nets. In *The 13th European Control Conference (ECC'14).* (Cited in pages 24, 49, 69, 120 and 122.)

[Liu et al., 2016] Liu, B., Ghazel, M., and Toguyéni, A. (2016). Model-based diagnosis of multi-track level crossing plants. *IEEE Transactions on Intelligent Transportation Systems*, 17(2):546–556. (Cited in page 178.)

[Madalinski and Khomenko, 2010] Madalinski, A. and Khomenko, V. (2010). Diagnosability verification with parallel LTL-X model checking based on Petri net unfoldings. *Conference on Control and Fault-Tolerant Systems (SysTol).* (Cited in pages 24, 120 and 121.)

[Madden and Nolan, 1999] Madden, M. G. M. and Nolan, P. J. (1999). Monitoring and diagnosis of multiple incipient faults using fault tree induction. *IEE Proceedings - Control Theory and Applications*, 146(2). (Cited in page 84.)

[McMillan, 1993] McMillan, K. L. (1993). Symbolic model checking. pages 25–60. (Cited in pages 162 and 176.)

[McMillan, 2003] McMillan, K. L. (2003). Interpolation and sat-based model checking. In *International Conference on Computer Aided Verification*, pages 1–13. Springer. (Cited in page 162.)

[Męski et al., 2012] Męski, A., Penczek, W., Szreter, M., Woźna-Szcześniak, B., and Zbrzezny, A. (2012). Two approaches to bounded model checking for linear time logic with knowledge. In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pages 514–523. Springer. (Cited in pages 164 and 200.)

[Moreira et al., 2011] Moreira, M., Jesus, T., and Basilio, J. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 56(7):1679–1684. (Cited in pages 5, 22, 42, 49, 90, 158 and 183.)

[Morvan and Pinchinat, 2009] Morvan, C. and Pinchinat, S. (2009). Diagnosability of pushdown systems. *Haifa Verification Conference*, pages 21–33. (Cited in pages 31, 32, 33, 35, 36 and 158.)

[Murata, 1989] Murata, T. (1989). Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580. (Cited in pages 23, 120 and 125.)

[Noori Hosseini, 2011] Noori Hosseini, M. (2011). Diagnosis of discrete event systems. (Cited in page 246.)

[Nunes et al., 2016] Nunes, C. E., Moreira, M. V., Alves, M. V., and Basilio, J. C. (2016). Network codiagnosability of discrete-event systems subject to event communication delays. In *2016 13th International Workshop on Discrete Event Systems (WODES)*, pages 217–223. IEEE. (Cited in pages 22 and 49.)

[Ostroff, 1989] Ostroff, J. S. (1989). Synthesis of controllers for real-time discrete event systems. In *Proceedings of the* 28^{th} *IEEE Conference on Decision and Control*, pages 138–144. IEEE. (Cited in page 162.)

[Ostroff and Wonham, 1990] Ostroff, J. S. and Wonham, W. M. (1990). A framework for real-time discrete event control. *IEEE Transactions on Automatic control*, 35(4):386–397. (Cited in page 162.)

[Pan and Hashtrudi-Zad, 2007] Pan, J. and Hashtrudi-Zad, S. (2007). Diagnosability analysis and sensor selection in discrete-event systems with permanent failures. *IEEE International Conference on Automation Science and Engineering*, pages 869–874. (Cited in pages 21 and 212.)

[Pan et al., 2012] Pan, S., Hu, Y., and Li, X. (2012). Ivf: Characterizing the vulnerability of microprocessor structures to intermittent faults. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(5):777–790. (Cited in page 84.)

[Pandalai and Holloway, 2000] Pandalai, D. N. and Holloway, L. E. (2000). Template languages for fault monitoring of timed discrete event processes. *IEEE transactions on automatic control*, 45(5):868–882. (Cited in pages 23 and 211.)

[Patton and Chen, 1991] Patton, R. and Chen, J. (1991). A review of parity space approaches to fault diagnosis. In *IFAC Safeprocess Conference*, pages 65–81. (Cited in page 19.)

[Pecheur et al., 2002] Pecheur, C., Cimatti, A., and Cimatti, R. (2002). Formal verification of diagnosability via symbolic model checking. In *Workshop on Model Checking and Artificial Intelligence (MoChArt-2002), Lyon, France.* (Cited in pages 6, 7, 158 and 159.)

[Pencole, 2000] Pencole, Y. (2000). Decentralized diagnoser approach: application to telecommunication networks. *proceedings of the International Workshop on Principles of Diagnosis*, pages 185–192. (Cited in pages 22 and 49.)

[Pencolé and Cordier, 2005] Pencolé, Y. and Cordier, M.-O. (2005). A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(1):121–170. (Cited in page 23.)

[Pencolé et al., 2004] Pencolé, Y. et al. (2004). Diagnosability analysis of distributed discrete event systems. In *ECAI*, volume 16, page 43. (Cited in page 23.)

[Pencolé et al., 2005] Pencolé, Y. et al. (2005). Assistance for the design of a diagnosable component-based system. In *ICTAI*, pages 549–556. (Cited in page 23.)

[Peres and Ghazel, 2014] Peres, F. and Ghazel, M. (2014). An operative formulation of the diagnosability of discrete event systems using a single logical framework. pages 1–11. (Cited in page 166.)

[Petri, 1966] Petri, C. A. (1966). Communication with automata. *Ph.D. Thesis*. (Cited in page 120.)

[Philippot et al., 2013] Philippot, A., Marangé, P., and Riera, R. (2013). Decentralized diagnosis and diagnosability by Model-Checking. *Universal Journal of Control and Automation*, 1(2):28–33. (Cited in pages 22, 49, 158 and 166.)

[Picardi,] Picardi, C. A short tutorial on model-based diagnosis. (Cited in page 20.)

[Plath and Ryan, 2000] Plath, M. and Ryan, M. D. (2000). The feature construct for SMV: Semantics. *FIW*, pages 129–144. (Cited in page 176.)

[Pnueli, 1977] Pnueli, A. (1977). The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE. (Cited in page 161.)

[Pocci, 2012] Pocci, M. (2012). A toolbox for diagnosability of Petri nets. *http://www.diee.unica.it/giua/ TESI/09_Marco .Pocci/*. (Cited in page 147.)

[Poole, 1989] Poole, D. (1989). Explanation and prediction: an architecture for default and abductive reasoning. *Computational Intelligence*, 5(2):97–110. (Cited in page 20.)

[Prasad et al., 2005] Prasad, M. R., Biere, A., and Gupta, A. (2005). A survey of recent advances in sat-based formal verification. *International Journal on Software Tools for Technology Transfer*, 7(2):156–173. (Cited in page 162.)

[Prasad, 1990] Prasad, V. B. (1990). Computer networks reliability evaluations and intermittent faults. *Proceedings of the 33rd Midwest Symposium on Circuits and Systems*, pages 327–330. (Cited in page 84.)

[Provan, 2002] Provan, G. (2002). On the diagnosability of decentralized, timed discrete event systems. In *Proceedings of the 41st IEEE Conference on Desision and Control*, pages 405–410. (Cited in pages 22, 49 and 158.)

[Qiu and Kumar, 2006] Qiu, W. and Kumar, R. (2006). Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 36(2):384–395. (Cited in pages 5, 22, 49 and 158.)

[Qiu et al., 2009] Qiu, W., Wen, Q., and Kumar, R. (2009). Decentralized diagnosis of event-driven systems for safely reacting to failures. *IEEE Transactions on Automation Science and Engineering*, 6(2):362–366. (Cited in page 158.)

[Queille and Sifakis, 1982] Queille, J.-P. and Sifakis, J. (1982). Specification and verification of concurrent systems in cesar. In *International Symposium on Programming*, pages 337–351. Springer. (Cited in pages 159 and 161.)

[Ramadge and Wonham, 1987] Ramadge, P. J. and Wonham, W. M. (1987). Supervisory control of a class of discrete event processes. *SIAM journal on control and optimization*, 25(1):206–230. (Cited in pages 21 and 27.)

[Ramírez-Treviño et al., 2007] Ramírez-Treviño, A., Ruiz-Beltrán, E., Rivera-Rangel, I., and Lopez-Mellado, E. (2007). Online fault diagnosis of discrete event systems. a Petri net-based approach. *IEEE Transactions on Automation Science and Engineering*, 4(1):31–39. (Cited in pages 24 and 120.)

[Reiter, 1987] Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95. (Cited in pages 19 and 20.)

[Reniers and Willemse, 2011] Reniers, M. A. and Willemse, T. A. (2011). Folk theorems on the correspondence between state-based and event-based systems. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 494–505. Springer. (Cited in page 160.)

[Ricker and Fabre, 2000] Ricker, S. and Fabre, E. (2000). On the construction of modular observers and diagnosers for discrete-event systems. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, volume 3, pages 2240–2244. IEEE. (Cited in pages 23 and 211.)

[Rintanen et al., 2007a] Rintanen, J. et al. (2007a). Diagnosers and diagnosability of succinct transition systems. *International Joint Conference on Artificial Intelligence*, pages 538–544. (Cited in pages 32 and 36.)

[Rintanen et al., 2007b] Rintanen, J., Grastien, A., et al. (2007b). Diagnosability testing with satisfiability algorithms. In *IJCAI*, pages 532–537. (Cited in page 165.)

[Roberts, 1989] Roberts, M. (1989). A fault-tolerant scheme that copes with intermittent and transient faults in sequential circuits. *Proceedings of the* 32^{nd} *Midwest Symposium on Circuits and Systems*, pages 36–39. (Cited in page 83.)

[Roth et al., 2012] Roth, M., Schneider, S., Lesage, J.-J., and Litz, L. (2012). Fault Detection and Isolation in Manufacturing Systems with an Identi ed Discrete Event Model. *International Journal of Systems Science*, 43(10):1826–1841. (Cited in page 19.)

[Ru and Hadjicostis, 2010] Ru, Y. and Hadjicostis, C. N. (2010). Sensor selection for structural observability in discrete event systems modeled by Petri nets. *IEEE Transactions on Automatic Control*, 55(7):1–14. (Cited in pages 21 and 212.)

[Saboori and Hadjicostis, 2007] Saboori, A. and Hadjicostis, C. N. (2007). Notions of security and opacity in discrete event systems. In *Decision and Control, 2007 46th IEEE Conference on*, pages 5056–5061. IEEE. (Cited in page 213.)

[Salvatore et al., 2003] Salvatore, J. B., Elizabeth, R., Joanne Bechta, D., Kishor S, T., Nitin, M., Robert M, G., and Mark D, S. (2003). Hybrid automated reliability predictor (HARP) integrated reliability tool system (version 7.0) HARP introduction and user's guide. *NASA Langley Technical Report Server*. (Cited in page 84.)

[Sampath et al., 1998] Sampath, M., Lafortune, S., and Teneketzis, D. (1998). Active diagnosis of discrete-event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929. (Cited in page 21.)

[Sampath et al., 1995] Sampath, M., Sengupta, R., and Lafortune, S. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, pages 1555–1575, 40(9). (Cited in pages 4, 5, 19, 21, 22, 27, 30, 31, 32, 33, 36, 37, 38, 39, 48, 49, 51, 62, 68, 70, 72, 74, 86, 88, 89, 90, 94, 99, 105, 113, 120, 121, 124, 133, 135, 136, 139, 147, 152, 158, 163, 166, 168, 169, 170, 171, 179 and 244.)

[Sampath et al., 1996] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, D. C. (1996). Failure diagnosis using discrete-event models. *IEEE transactions on control systems technology*, 4(2):105–124. (Cited in pages 5, 21, 30, 31, 32, 36, 49 and 158.)

[Sayed-Mouchaweh and Carre-Menetrier, 2008] Sayed-Mouchaweh, M., a. P. A. and Carre-Menetrier, V. (2008). Decentralized diagnosis based on boolean discrete event models: application on manufacturing systems. *International Journal of Production Research*, 46(9):5469–5490. (Cited in pages 22, 49 and 158.)

[Scherer and White III, 1989] Scherer, W. T. and White III, C. C. (1989). A survey of expert systems for equipment maintenance and diagnostics. *Knowledge-Based System Diagnosis, Supervision, and Control*, pages 285–300. (Cited in page 17.)

[Schmidt, 2010] Schmidt, K. (2010). Abstraction-based verification of codiagnosability for discrete event systems. *Automatica*, 46(9):1489–1494. (Cited in page 158.)

[Schmidt, 2013] Schmidt, K. W. (2013). Verification of modular diagnosability with local specifications for discrete-event systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(5):1130–1140. (Cited in pages 23, 158 and 211.)

[Schneider et al., 2012] Schneider, S., Litz, L., and Lesage, J.-J. (2012). Determination of Timed Transitions in Identified Discrete-Event Models for Fault Detection. In 51^{st} *IEEE Annual Conference on Decision and Control*, pages 5816–5821. (Cited in page 19.)

[Schumann and Pencolé, 2007] Schumann, A. and Pencolé, Y. (2007). Scalable diagnosability checking of event-driven system. *20th Int. Joint Conference on Artificial Intelligence*. (Cited in pages 23 and 158.)

[Schumann et al., 2010] Schumann, A., Pencole, Y., and Thiebaux, S. (2010). A decentralised symbolic diagnosis approach. 19^{th} *European Conference on Artificial Intelligence*, pages 99–104. (Cited in pages 22 and 49.)

[Sharma et al., 2015] Sharma, R., Dewan, L., and Chatterji, S. (2015). ault diagnosis methods in dynamic systems: A review. *International Journal of Electronics and Electrical Engineering*, 3(6):465–471. (Cited in page 82.)

[Shen et al., 2016] Shen, Q., Qiu, J., Liu, G., and Lv, K. (2016). Intermittent faults parameter framework and stochastic Petri net based formalization model. *Eksploatacja I neizawodnosc*, 18(2):1–210. (Cited in pages 6 and 83.)

[Shewhart and Deming, 1939] Shewhart, W. A. and Deming, W. E. (1939). *Statistical method from the viewpoint of quality control*. Courier Corporation. (Cited in page 18.)

[Soldani et al., 2006] Soldani, S., Combacau, M., Subias, A., and Thomas, J. (2006). Intermittent fault detection through message exchanges: a coherence based approach. *International Workshop Principles Diagnosis*, pages 251–257. (Cited in page 86.)

[Soldani et al., 2007] Soldani, S., Combacau, M., Subias, A., and Thomas, J. (2007). Intermittent fault diagnosis: a diagnoser derived from the normal behavior. pages 391–399. (Cited in page 86.)

[Sorensen et al., 1994] Sorensen, B., Kelly, G., Sajecki, A., and Sorensen, P. (1994). An analyzer for detecting intermittent faults in electronic devices. *IEEE Systems Readiness Technology Conference.'Cost Effective Support Into the Next Century', Conference Proceedings.*, pages 417–421. (Cited in pages 6, 83 and 84.)

[Srinivasan and Gluch, 1998] Srinivasan, G. R. and Gluch, D. P. (1998). A study of practice issues in model-based verification using the symbolic model verifier (SMV). (Cited in page 176.)

[Steadman et al., 2008] Steadman, B., Berghout, F., Olsen, N., and Sorensen, B. (2008). Intermittent fault detection and isolation system. *IEEE AUTOTESTCON*, pages 37–40. (Cited in pages 6 and 83.)

[Steadman et al., 2005] Steadman, B., Sievert, S., Sorensen, B., and Berghout, F. (2005). Attacking" bad actor" and" no fault found" electronic boxes. *IEEE Autotestcon*, pages 821–824. (Cited in pages 6 and 83.)

[Stéphane Lafortune and Paoli, 2014] Stéphane Lafortune, R. H. and Paoli, A. (2014). Fault diagnosis of manufacturing systems using finite state automata. pages 601–626. (Cited in pages xiv, 244 and 245.)

[Storey, 1996] Storey, N. R. (1996). *Safety critical computer systems*. Addison-Wesley Longman Publishing Co., Inc. (Cited in page 2.)

[Su and Wonham, 2005] Su, R. and Wonham, W. M. (2005). Global and local consistencies in distributed fault diagnosis for discrete-event systems. *IEEE Transactions on Automatic Control*, 50(12):1923–1935. (Cited in page 23.)

[Syed et al., 2013] Syed, W. A., Khan, S., Phillips, P., and Perinpanayagam, S. (2013). Intermittent fault finding strategies. *Procedia CIRP*, 11:74–79. (Cited in pages 83 and 84.)

[Takai and Kumar, 2012] Takai, S. and Kumar, R. (2012). Distributed failure prognosis of discrete event systems with bounded-delay communications. *IEEE Transactions on Automatic Control*, 57(5):1259–1265. (Cited in page 213.)

[Takai and Kumar, 2016] Takai, S. and Kumar, R. (2016). Delay bound of inference-based decentralized diagnosis in discrete event systems. In *2016 13th International Workshop on Discrete Event Systems (WODES)*, pages 224–229. IEEE. (Cited in pages 22 and 49.)

[Thistle and Wonham, 1986] Thistle, J. and Wonham, W. (1986). Control problems in a temporal logic framework. *International Journal of Control*, 44(4):943–976. (Cited in page 162.)

[Tripakis, 2002] Tripakis, S. (2002). Fault diagnosis for timed automata. *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 205–221. (Cited in pages 32 and 42.)

[Tsitsiklis, 1989] Tsitsiklis, J. N. (1989). On the control of discrete-event dynamical systems. *Mathematics of Control, Signals and Systems*, 2(2):95–107. (Cited in page 32.)

[Tzafestas and Watanabe, 1990] Tzafestas, S. and Watanabe, K. (1990). Modern approaches to system sensor fault detection and diagnosis. *Journal A*, 31(4):42–57. (Cited in page 17.)

[Ushio et al., 1998] Ushio, T., Onishi, I., and Okuda, K. (1998). Fault detection based on Petri net models with faulty behaviors. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 1, pages 113–118. (Cited in pages 24, 49, 120, 122 and 139.)

[Varpaaniemi et al., 1997] Varpaaniemi, K., Heljanko, K., and Lilius, J. (1997). Prod 3.2 an advanced tool for efficient reachability analysis. *International Conference on Computer Aided Verification*, pages 472–475. (Cited in page 148.)

[Venkatasubramanian et al., 2003] Venkatasubramanian, V., Rengaswamy, R., Yin, K., and Kavuri, S. N. (2003). A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & chemical engineering*, 27(3):293–311. (Cited in pages 17, 18 and 19.)

[Wang et al., 2004] Wang, Y., Yoo, T.-S., and Lafortune, S. (2004). New results on decentralized diagnosis of discrete event systems. *Proceedings of 2004 Annual.* (Cited in page 158.)

[Wang et al., 2007] Wang, Y., Yoo, T. S., and Lafortune, S. (2007). Diagnosis of discrete event systems using decentralized architectures. *Discrete Event Dynamic Systems: Theory and Applications*, 17(2):233–263. (Cited in pages 22, 49 and 158.)

[Wen et al., 2005] Wen, Y., Li, C., and Jeng, M. (2005). A polynomial algorithm for checking diagnosability of Petri nets. In *IEEE International Conference on Systems, Man and Cybernetics (SMC'05)*, volume 3, pages 2542–2547. IEEE. (Cited in pages 24, 120 and 121.)

[Willsky, 1976] Willsky, A. S. (1976). A survey of design methods for failure detection in dynamic systems. *Automatica*, 12(6):601–611. (Cited in page 19.)

[Yan et al., 2015] Yan, R., He, X., and Zhou, D. (2015). Robust detection of intermittent faults for linear discrete-time stochastic systems with parametric perturbations. 34^{th} *Chinese Control Conference (CCC)*, pages 6308–6313. (Cited in page 84.)

[Ye and Dague, 2010] Ye, L. and Dague, P. (2010). Diagnosability analysis of discrete event systems with autonomous components. In *ECAI*, pages 105–110. (Cited in page 23.)

[Ye and Dague, 2012] Ye, L. and Dague, P. (2012). A general algorithm for pattern diagnosability of distributed discrete event systems. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, volume 1, pages 130–137. IEEE. (Cited in page 23.)

[Ye and Dague, 2013] Ye, L. and Dague, P. (2013). Undecidable case and decidable case of joint diagnosability in distributed discrete event systems. *International Journal on Advances in Systems and Measurements*, 6(3). (Cited in pages 49 and 158.)

[Ye et al., 2009] Ye, L., Dague, P., and Yan, Y. (2009). A distributed approach for pattern diagnosability. In *Proceedings of the 20th International Workshop on Principles of Diagnosis (DX-09)*, pages 179–186. (Cited in page 23.)

[Yoo and Garcia, 2003] Yoo, T. and Garcia, H. (2003). Computation of fault detection delay in discrete-event systems. *Proceedings of the* 14^{th} *International Workshop on Principles of Diagnosis*, pages 207–212. (Cited in page 36.)

[Yoo and Garcia, 2004] Yoo, T.-S. and Garcia, H. (2004). Event diagnosis of discrete-event systems with uniformly and nonuniformly bounded diagnosis delays. *American Control Conference*, 6:5102–5107. (Cited in pages 85, 116 and 158.)

[Yoo and Garcia, 2009] Yoo, T.-S. and Garcia, H. E. (2009). Event counting of partially-observed discrete-event systems with uniformly and nonuniformly bounded diagnosis delays. *Discrete Event Dynamic Systems*, 19(2):167–187. (Cited in pages 85 and 116.)

[Yoo and Lafortune, 2002a] Yoo, T.-S. and Lafortune, S. (2002a). NP-completeness of sensor selection problems arising in partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, 47(9):1495–1499. (Cited in pages 21 and 212.)

[Yoo and Lafortune, 2002b] Yoo, T. S. and Lafortune, S. (2002b). Polynomial-time verification of diagnosability of partially observed discrete-event systems. 47(9):1491–1495. (Cited in pages 5, 22, 36, 40, 41, 42, 48, 72, 77, 86, 120, 121, 158, 165 and 182.)

[Zad, 1999] Zad, S. H. (1999). *Fault diagnosis in discrete-event and hybrid systems*. PhD thesis, University of Toronto. (Cited in page 244.)

[Zad et al., 2003] Zad, S. H., Kwong, R. H., and Wonham, W. M. (2003). Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Transactions on Automatic Control*, pages 1199–1212. (Cited in pages 22, 49, 68, 86, 99, 120, 121, 139, 158, 163 and 166.)

[Zaytoon and Lafortune, 2013] Zaytoon, J. and Lafortune, S. (2013). Overview of fault diagnosis methods for Discrete Event Systems. *Annual Reviews in Control*, pages 308–320. (Cited in pages 4, 6, 22, 23, 27, 30, 33, 34, 82 and 123.)

[Zhou and Kumar, 2009] Zhou, C. and Kumar, R. (2009). Computation of diagnosable fault-occurrence indices for systems with Rrepeatable-faults. *Proceedings of the* 44^{th} *IEEE Conference on Decision and Control*, pages 6311–6316. (Cited in pages 85, 116 and 191.)

[Zhou et al., 2008] Zhou, C., Kumar, R., and Sreenivas, R. (2008). Decentralized modular diagnosis of concurrent discrete event systems. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*, pages 388–393. IEEE. (Cited in pages 22, 23, 49 and 158.)

Appendix A

An Illustrative Example using NuSMV

A.1 Heating Ventilation and Air-Conditioning System

Let us consider the HVAC system discussed in [Zad, 1999, Sampath et al., 1995, Cassandras and Lafortune, 2009, Stéphane Lafortune and Paoli, 2014] and shown in Figure A.1. It consists of a pump, a vale, and a controller, represented by FSA. Considering the valve model, the nominal behavior of the valve is that it is initially closed (VC), where the initial state is indicated by the short arrow, and the valve transitions to its open position (VO) under the open valve command (ov). Likewise, the valve returns to the closed position under the close valve command (cv). This model also includes faulty behavior in that the valve can transition to a stuck-closed (SC) or stuck-open (SO) state via the occurrence of fault events sc and so, respectively. For simplicity, the pump model does not include any faulty behavior and transitions between the pump off-state (POFF) and the pump-on state (PON) under the commands poff and pon, respectively. The controller runs the system through the cycle of events: open valve, turn pump on, turn pump off and close valve. Such a cycle could be initiated by a higher level controller where the material output of this subsystem is being employed for cooling or is feeding a larger manufacturing process [Stéphane Lafortune and Paoli, 2014].

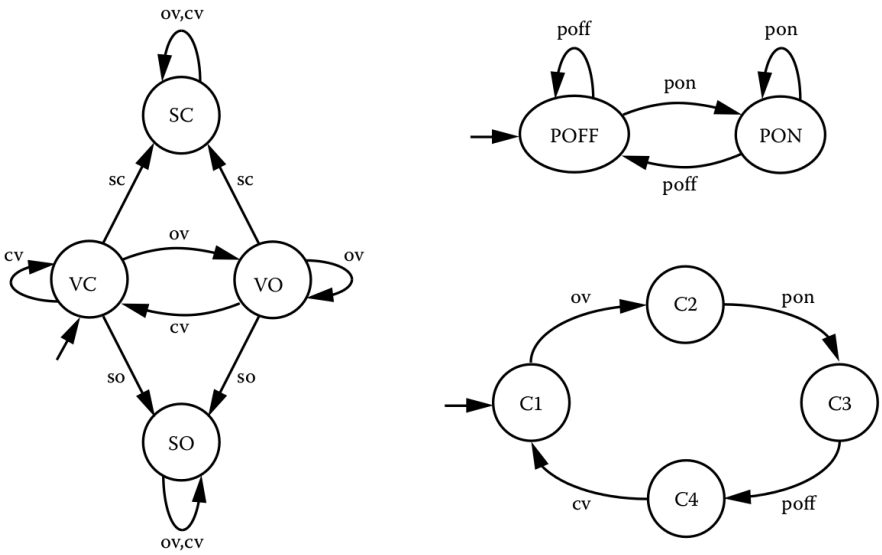


Figure A.1 – HVAC system components [Stéphane Lafortune and Paoli, 2014]

The global model of the HVAC system is obtained by performing the parallel composition of the component models (valve, pump, controller). The global model is illustrated in Figure A.2.

Formally, the HVAC system is an FSA $G = \langle X, \Sigma, \delta, x_0 \rangle$, where $X = \{$VC_POFF_C1,

VO_POFF_C2,VO_PON_C3, VO_POFF_C4, SO_POFF_C1, SO_POFF_C2, SO_PON_C3, SO_POFF_C4, SC_POFF_C1, SC_POFF_C2, SC_PON_C3, SC_POFF_C4 } (as illustrated in Figure A.2). Set of events Σ = {CV, OV, SO, SC, POFF, PON } and x_0 = VC_POFF_C1, which is indicated by a short arrow in Figure A.2. Two fault event exist in the model, SC and SO. For simplicity, we consider that both events SC and SO belong to the same fault class, i.e., Σ_f = {SC, SO}. The rest of events are considered to be observable. It is worth noticing that G can be seen as an event-based model, i.e., the occurrence of the faulty behavior is modeled by the occurrence of faulty events SC and SO, or as a state-based automaton, i.e., the occurrence of the faulty behavior is modeled by the reachability of faulty states. In fact, set of states X can be partitioned into to disjoint subset of states: $X = X_N \biguplus X_N$, with X_N = { VC_POFF_C1, VO_POFF_C2,VO_PON_C3, VO_POFF_C4, } and X_F = { SO_POFF_C1, SO_POFF_C2, SO_PON_C3, SO_POFF_C4, SC_POFF_C1, SC_POFF_C2, SC_PON_C3, SC_POFF_C4 }.

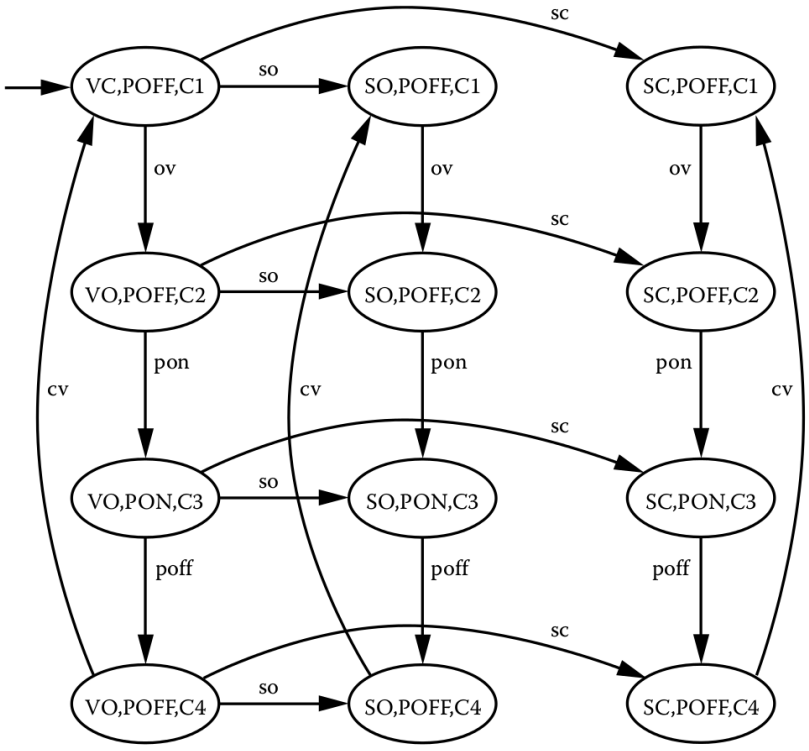


Figure A.2 – HVAC system model [Stéphane Lafortune and Paoli, 2014]

A.2 The Diagnosability Analysing Using NuSMV

In this section, we provide the NuSMV program for analyzing diagnosability. The program is composed of two modules: HVAC and main. Module HVAC contains the observable behavior of the HVAC system, i.e., a description of the generator G' of G obtained by the ε-reduction. Module main contains the instantiation of two copies of generator G' (plant_L and plant_R) which will be used to construct the twin-plant as a Kripke structure. Variable SameObs_Trace is used as an invariant to ensure that only the indistinguishable traces are generated. Finally, the CTL specification, which expresses the necessary and sufficient condition for diagnosability of permanent faults, is integrated.

It is worth noticing that the NUSMV program we present here, is not the optimized one (regarding the generated state-space of the Kripke structure). However, it allows for better illustrating the concept of the twin-plant approach [Jiang and Huang, 2001, Cimatti et al., 2003]. In Script 4 (Appendix A.2), we give another manner of coding the two copies of generator G', where the synchronization regarding observable event is performed in the main module, i.e., the two copies (plant_L and plant_R) shares the same observable events. Various manners of encoding the twin-plant in NuSMV can be found in [Cimatti et al., 2003, Huang, 2003, Noori Hosseini, 2011].

Script 3 (Appendix A.2), give the result output by executing The NuSMV model-checker. The system is checked to be non-diagnosable and a counter-example is generated with a witnessed trace.

Script 1

```
-----------------------------------------------------
--        Script I: HVAC NuSMV module
-----------------------------------------------------
MODULE HVAC
VAR
         State :{VC_POFF_C1, VO_POFF_C2,VO_PON_C3, VO_POFF_C4,
                  SO_POFF_C1, SO_POFF_C2, SO_PON_C3, SO_POFF_C4,
                  SC_POFF_C1, SC_POFF_C2, SC_PON_C3,  SC_POFF_C4};
         Event :{CV, OV, PON, POFF};
ASSIGN
 Event:= case
        State  in {VC_POFF_C1, SO_POFF_C1, SC_POFF_C1}        : OV;
        State  in {VO_POFF_C2, SO_POFF_C2, SC_POFF_C2}        : PON;
        State  in {VO_PON_C3, SO_PON_C3, SC_PON_C3}           : POFF;
        State  in {VO_POFF_C4, SO_POFF_C4, SC_POFF_C4}        : CV;
--      TRUE                                                  :Event;
         esac;
 init(State)                          :=VC_POFF_C1;
 next(State)                          := case
        State = VC_POFF_C1  & Event = OV:{VO_POFF_C2, SO_POFF_C2, SC_POFF_C2};
        State = VO_POFF_C2  & Event = PON:{VO_PON_C3, SO_PON_C3, SC_PON_C3};
        State = VO_PON_C3   & Event = POFF:{VO_POFF_C4, SO_POFF_C4, SC_POFF_C4};
        State = VO_POFF_C4  & Event = CV:{VC_POFF_C1, SO_POFF_C1, SC_POFF_C1};

        State = SO_POFF_C1 & Event = OV         :SO_POFF_C2;
        State = SO_POFF_C2 & Event = PON        :SO_PON_C3;
        State = SO_PON_C3 & Event = POFF        :SO_POFF_C4;
        State = SO_POFF_C4 & Event = CV         :SO_POFF_C1;

        State = SC_POFF_C1 & Event = OV         :SC_POFF_C2;
        State = SC_POFF_C2 & Event = PON        :SC_PON_C3;
        State = SC_PON_C3 & Event = POFF        :SC_POFF_C4;
        State = SC_POFF_C4 & Event = CV         :SC_POFF_C1;
        TRUE                                     : State;
         esac;
```

Script 2

```
-----------------------------------------------------
--        Script II:  module main
-----------------------------------------------------
MODULE main
VAR
plant_L                : HVAC;
plant_R                : HVAC;

DEFINE         NormalState := {VC_POFF_C1, VO_POFF_C2,VO_PON_C3, VO_POFF_C4};
DEFINE         FaultyState := {SO_POFF_C1, SO_POFF_C2, SO_PON_C3, SO_POFF_C4,
                              SC_POFF_C1, SC_POFF_C2, SC_PON_C3,  SC_POFF_C4};

DEFINE  SameObs := ((plant_L.Event = plant_R.Event));
VAR
SameObs_Trace : boolean ;

ASSIGN
        init (SameObs_Trace)                :=SameObs;
        next (SameObs_Trace)                :=case
                !SameObs                    :FALSE;
                 SameObs                    :TRUE ; --next(SameObs);
           esac;
INVAR SameObs_Trace

CTLSPEC !EF(EG((plant_R.State in NormalState)&(plant_L.State in FaultyState)))
```

Script 3

```
NuSMV > read_model -i HVAC_Thesis.smv
NuSMV > go
NuSMV > print_reachable_states
###########################################################################
system diameter: 5
reachable states: 36 (2^5.16993) out of 4608 (2^12.1699)
###########################################################################

NuSMV > check_ctlspec
-- specification !(EF (EG (plant_R.State in NormalState &
 plant_L.State in FaultyState)))  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
  plant_L.State = VC_POFF_C1
  plant_L.Event = OV
  plant_R.State = VC_POFF_C1
  plant_R.Event = OV
  SameObs_Trace = TRUE
-- Loop starts here
-> State: 1.2 <-
  plant_L.State = SO_POFF_C2
  plant_L.Event = PON
  plant_R.State = VO_POFF_C2
  plant_R.Event = PON
-> State: 1.3 <-
  plant_L.State = SO_PON_C3
  plant_L.Event = POFF
  plant_R.State = VO_PON_C3
  plant_R.Event = POFF
-> State: 1.4 <-
  plant_L.State = SO_POFF_C4
  plant_L.Event = CV
  plant_R.State = VO_POFF_C4
  plant_R.Event = CV
-> State: 1.5 <-
  plant_L.State = SO_POFF_C1
  plant_L.Event = OV
  plant_R.State = VC_POFF_C1
  plant_R.Event = OV
-> State: 1.6 <-
  plant_L.State = SO_POFF_C2
  plant_L.Event = PON
  plant_R.State = VO_POFF_C2
  plant_R.Event = PON
```

Script 4

```
MODULE main
VAR
Event          :{CV, OV, PON, POFF};
plant_L        : HVAC(Event);
plant_R        : HVAC(Event);

DEFINE         NormalState := {VC_POFF_C1, VO_POFF_C2,VO_PON_C3, VO_POFF_C4};
DEFINE         FaultyState := {SO_POFF_C1, SO_POFF_C2, SO_PON_C3, SO_POFF_C4,
                              SC_POFF_C1, SC_POFF_C2, SC_PON_C3,  SC_POFF_C4};
ASSIGN
         Event                                  := case
        plant_L.State  in {VC_POFF_C1, SO_POFF_C1, SC_POFF_C1} : OV;
        plant_L.State  in {VO_POFF_C2, SO_POFF_C2, SC_POFF_C2} : PON;
        plant_L.State  in {VO_PON_C3, SO_PON_C3, SC_PON_C3}    : POFF;
        plant_L.State  in {VO_POFF_C4, SO_POFF_C4, SC_POFF_C4} : CV;
        -- TRUE                                                :Event;
         esac;

CTLSPEC !EF(EG((plant_R.State in NormalState)&(plant_L.State in FaultyState)))
```

Contributions to Model-Based Diagnosis of Discrete-Event Systems

Abstract : This PhD thesis deals with fault diagnosis of discrete-event systems modeled as finite state automata with some extensions to bounded Petri net models. The developed contributions can be classified regarding two pioneering approaches from the literature: the diagnoser-based technique and the twin-plant based technique.

Regarding the diagnoser-based technique, we propose a new diagnoser variant with some interesting features that allow us to separately track the normal and the faulty traces directly in the diagnoser. On the basis of the developed diagnoser, we reformulate the necessary and sufficient condition for diagnosability of permanent faults and we propose a systematic procedure for checking such a condition without building any intermediate model. An on-the-fly algorithm, for simultaneously constructing the diagnoser and verifying diagnosability is developed. The algorithm aims to generate as less state-space as possible, particularly when the system is non-diagnosable, which improves the memory/time consumption. The developed diagnoser is then extended to deal with fault diagnosis of intermittent faults. Various notions of diagnosability are addressed and necessary and sufficient conditions are formulated on the basis of the new diagnoser structure.

A Hybrid version (in the sense of combining enumerative and symbolic representations) of the diagnoser variant is established in order to deal with fault diagnosis of labeled bounded Petri nets. The main idea consists in using the symbolic representation (Binary Decision Diagrams) to compact and handle the Petri net markings inside the diagnoser nodes and using explicit representation for the (observable) transitions that link the diagnoser nodes. Such a combination serves, on one hand, to reduce the memory required for constructing the diagnoser and, on the other hand, to easily explore the diagnoser paths. The developed approaches are implemented in dedicated tools and evaluated through benchmarks with respect to the reference approaches in the domain.

Regarding twin-plant based technique, the first contribution consists in elaborating a model-checking framework, that extends the Cimatti's work, for the actual verification of various diagnosability concepts pertaining to permanent failures based on the twin-plant structure. The main idea is to reformulate and express the diagnosability issues as temporal logics and then to tackle them using the model-checking engines. The second contribution is pertaining to the diagnosis of intermittent faults, and consists in addressing various notions of diagnosability while establishing their necessary and sufficient conditions on the basis of the twin-plant structure.

Keywords : Fault diagnosis, Discrete-event systems, Intermittent faults, Permanent faults, Finite state automata, Labeled Petri nets, Model-checking.