Diagnosis and Diagnosability of Complex Discrete Event Systems Modeled by Labeled Petri Nets

Ben Li

N° d'ordre: 3 1 8

CENTRALE LILLE

THÈSE

présentée en vue d'obtenir le grade de

DOCTEUR

En

Spécialité : Automatique, Génie informatique, Traitement du signal et des images

Par

Ben LI

DOCTORAT DÉLIVRÉ PAR CENTRALE LILLE

Titre de la thèse :

Diagnosis and Diagnosability of Complex Discrete Event Systems Modeled by Labeled Petri Nets

Diagnostic et Diagnosticabilité des Systèmes à Événements Discrets Complexes Modélisés par des Réseaux de Petri Labellisés

---

Soutenue le 3 mai 2017 devant le jury d'examen :

Président	Prof. Rochdi MERZOUKI	Polytech Lille
Rapporteur	Prof. Eric NIEL	INSA de Lyon
Rapporteur	Prof. Dimitri LEFEBVRE	Université du Havre
Examinateur	Prof. João Dos Santos BASILIO	Université Fédérale de Rio
Examinatrice	MCF Ramla SADDEM	Université de Reims Champagne Ardennes
Examinateur	CR-HDR Mohamed GHAZEL	IFSTTAR
Directeur de thèse	Prof. Armand TOGUYÉNI	Centrale Lille
Encadrante	MCF Manel KHLIF-BOUASSIDA	Centrale Lille

Thèse préparée dans le Centre de Recherche en Informatique, Signal et Automatique de Lille, CRIStAL, CNRS UMR 9189

École Doctorale Sciences pour l'Ingénieur (SPI) 072 (Centrale Lille)

To my parents,
to all my family,
to my professors,
and to all my friends.

ACKNOWLEDGEMENTS

This research work has been realized at "Centre de Recherche en Informatique, Signal et Automatique de Lille (CRIStAL)" in Centrale Lille, with the research team "Modèles et Outils formels pour des Systèmes à Evénements discrets Sûrs (MOSES)" from March 2014 to May 2017. This work is financially supported by China Scholarship Council (CSC). I would never have this wonderful living and working experience in France.

First and foremost, I offer my sincerest gratitude to my supervisor Prof. Armand TOGUYENI and my co-supervisor Dr. Manel KHLIF-BOUASSIDA. They have provided their supervision, valuable guidance, continuous encouragement as well as given me extraordinary experiences throughout my working experience.

Besides my supervisors, I would like to thank Prof. Rochdi MERZOUKI for his kind acceptance to be the president of my PhD Committee. Then, I would like to express my sincere gratitude to Prof. Eric NIEL and Prof. Dimitri LEFEBVRE, who have kindly accepted the invitation to be reviewers of my PhD thesis, for their encouragement, insightful comments and helpful questions. I would also like to give my gratitude to Prof. João Dos Santos BASILIO, Dr. Ramla SADDEM and Dr. Mohamed GHAZEL for their kind acceptance to take part in the jury of the PhD defense.

I am very grateful to the staff in Centrale Lille, Mrs. Vanessa FLEURY, Mrs. Brigitte FONCEZ, Mrs. Christine YVOZ and Mrs. Salima HALITIM have helped me in the administrative work. Many thanks go also to Mr. Patrick GALLAIS, Mr. Gilles MARGUERITE and Mr. Jacques LASUE, for their kind help and hospitality in the laboratory. Special thanks go to Mrs. Christine VION, Mrs. Martine MOUVAUX for their support in my lodgment life.

My sincere thanks goes to Dr. Baisi LIU, for his useful advices during my study in the laboratory as well as after his graduation. My sincere thanks also goes to Prof. João Dos Santos BASILIO, for his kind supervision of my research work.

All my gratitude goes to Ms. Hélène CATSIAPIS, my French teacher, who showed us the French language and culture. She organized some interesting and unforgettable voyages in France, which inspired my knowledge and interest in the French culture, opened my appetite for art, history and cuisine/wine and enriched my life in France.

I would like to take the opportunity to express my gratitude and to thank my fellow workmates in CRIStAL: Hongchang ZHANG, Lijuan ZHANG, Yuchen XIE, Jianxin FANG and Paul CAZENAVE for the stimulating discussions for the hard teamwork. Many thanks go to my friends who helped me a lot during the past three years. I mention Jian ZHANG, Yue Wang, Qi SUN, Yihan LIU, Chen XIA, Daji TIAN, Qi GUO, Xuemei LIU, Abderaouf Boussif, Rahma Lahyani and many others whose names only by lack of memory I failed to include in this list.

My acknowledgements to all the professors and teachers in École Centrale de Pékin, Beihang University. The engineer education there not only gave me solid knowledge but also made it easier for me to live in France.

A special acknowledgment should be shown to Prof. Zhang REN and Dr. Qingdong LI at the School of Automation Science and Electrical Engineering, Beihang University, who enlightened me at the first glance of research. I always benefit from the abilities that I obtained in his research team.

I would like to offer my gratitude to thank my friend at the table tennis club LYS LILLE METROPOLE CP: Valentin, Dario, Eric, Frédéric, Jérémy, Dominic, Loic, etc. I spent a lot of joyful time with them.

Thanks to my parents, Junshan LI and Ling ZHANG, for living in my heart and mind despite being on the other side of the ocean. Especial thanks to my girl friend Jiaqian YU for her support accompany, patience, and encouragement.

Villeneuve d'Ascq, France *Ben LI*

May, 2017

Contents

List of Figures

List of Tables

ABBREVIATIONS

BFG	Basis Fault Marking graph
BFM	Basis Fault Marking
BFS	Basis Fault Marking Set
BFST	Basis Fault Marking Set Tree
BRD	Basis Reachability Diagnoser
CG	Coverability Graph
DES	Discrete Event System
ELOT	Exclusively Labeled Observable Transition
FM	Fault Marking
FM-graph	Fault Marking graph
FM-set	Fault Marking Set
FM-set tree	Fault Marking Set Tree
ILP	Integer Linear Programming
LPN	Labeled Petri Net
MBRG	Modified Basis Reachability Graph
MRG	Modified Reachability Graph
RG	Reachability Graph
VN	Verifier Net

CHAPTER 1

INTRODUCTION

Contents

This thesis deals with fault diagnosis of complex discrete event systems (DES) modeled by labeled Petri nets (LPN). This work is accomplished in the research team MOSES (Modèles et Outils formels pour des Systèmes à Evénements discrets Sûrs) of CRIStAL (Centre de Recherche en Informatique, Signal et Automatique de Lille, UMR 9189) laboratory, co-supervised by Prof. Armand Toguyéni and Dr. Manel Khlif-Bouassida.

1.1 Background

At the beginning of this century, with the advancement of new technologies, the competition between companies is increasing quickly to produce powerful automated and autonomous systems in many fields (manufacturing, transportation, health, space, etc.) Meanwhile, the complexity of automated and autonomous systems is increasing since more performance requirements become mandatory. When the operation of a complex system is related o human life or human safety, the system is called "critical" and it needs to impose automated fault diagnosis to avoid catastrophes.

A fault is defined as any deviation of system from its specified behavior. The fault diagnosis is the process that detects and identifies the fault and its type based on the observable symptoms.

Effective methods for fault diagnosis are required, because the efficiency of these

methods enhances the safety, the reliability, the availability and the competitiveness of systems.

In the literature, fault diagnosis approaches were classified in [Ven+03] as follows: model-based approaches; knowledge-based approaches; and data-based approaches. In addition to the nature of the information used to make the diagnosis, the structure of diagnosis itself is an important factor to be taken into consideration. We distinguish: monolithic (or centralized) diagnosis; decentralized diagnosis; modular diagnosis; and distributed diagnosis. Whatever the nature of the system (continuous systems, discrete event systems (DES) and hybrid systems) and the structure of the diagnosis, the classification in [Ven+03] remains valid. This thesis focuses on the model-based fault diagnosis of DES that is modeled by labeled Petri nets (LPN) and we address monolithic and modular diagnosis problems.

A DES is informally defined as a discrete-state and event-driven system. In detail, the definition of DES contains two main points: (1) the state space is discrete, e.g., a lamp has two states: "ON" and "OFF"; (2) the transition mechanism of states is driven by event e.g., the state of the lamp is changed from "OFF" to "ON" by the event: pressing the button of the lamp. In practice, many systems can be categorized and analyzed as DES such as embedded systems [Edw+97; Haj+13; ST15; Sch+00], transportation systems [Gha17; Haj+12; Liu+16; Paq+14] and manufacturing systems [Far+11; NN04; Tog+03; Zho+92]. The behavior of a DES is monitored by observing some events that can be detected by the sensors. These events are called "observable events". Nevertheless, there exist also some events that are undetectable by the sensors and they are called "unobservable events". The existence of unobservable events produces the ambiguity of the system, since the state of the system cannot be determined after the occurrence of an unobservable event. In this thesis, the faulty behaviors are modeled by unobservable events.

Two main issues of fault diagnosis of DES models are:

1. on-line diagnosis of the system;
2. off-line diagnosability analysis of the system.

The on-line diagnosis is to deduce the occurrence of faults (represented by unobservable events) and their types by using the observable events, while the system is running. The diagnosability represents the ability to detect a fault in a finite delay after its occurrence based on observations. The diagnosability is checked "off-line" (at the design stage of the system) and must be ensured before implementing the system. The on-line diagnosis is executable if and only if all the faults are "diagnosable". The most intuitive method is to enumerate the entire state space and then to verify certain formal conditions for diagnosability property. However, if the system is huge and complex, computational complexity and combinatorial explosion problems prevent the diagnosability analysis.

To overcome these problems, a lot of works have been proposed. An overview of the DES-based diagnosis in literature is provided in Appendix A.

This thesis proposes efficient approaches for diagnosability analysis of DES modeled by labeled Petri nets (LPN), dealing with the computational complexity and the combinatorial explosion to help to develop, in the future, software tools of diagnosability analysis for industrial use. It is worth noticing that LPN is a powerful formal modeling tool that gives a compact representation of DES and it is increasingly applied in industry.

1.2 Contributions

This research work focuses on the fault diagnosis of DES using LPN modeling formalism. Some new approaches for monolithic diagnosability analysis (in Chapter 3) and modular diagnosability analysis (in Chapter 4) are presented. The contributions are summarized as follows:

1. **Monolithic diagnosability analysis**: New diagnosability analysis techniques for LPN models with different assumptions are proposed. These techniques are based on structural properties of LPN model, in particular, T-invariants, reduction rules, minimal explanations.

   a) Some reduction rules are proposed to simplify the LPN model of system before analyzing its diagnosability. Some transitions and places could be suppressed. It is proved that the diagnosability property is preserved after using these reduction rules. These rules are strong complement for most of diagnosability analysis techniques using LPN existing in the literature.

   b) A new sufficient condition for the diagnosability of a *safe* and *live* LPN is proposed, which supplements the defect of the sufficient condition in [Wen+05]. A method is proposed by using linear programming technique to check this sufficient condition.

   c) The on-the-fly diagnosability analysis, previously developed in our research team [Liu+14] is improved using minimal explanations. By using minimal explanations, the state space of the system and the model for diagnosis are built in a compact manner to reduce the combinatorial explosion of diagnosability analysis.

   d) The on-the-fly diagnosability analysis is also improved by using T-invariant: by using the T-invariants, the priorities of investigating branches are defined. For a non-diagnosable LPN, the efficiency of on-the-fly diagnosability analysis is particularly improved.

   e) The on-the-fly diagnosability analysis using Verifier Nets is proposed, which can be used for both *bounded* and *unbounded* LPN model. The computational

complexity is polynomial for diagnosability analysis of *bounded* LPN. This contribution achieves a compromise between computation efficiency and combinatorial explosion limitation.

2. **Modular diagnosability analysis**: the strong assumption on liveness in [Con+06] is removed. An approach with lower computational complexity is proposed.

   a) Some reduction rules are applied to simplify the LPN model before analyzing the modular diagnosability. It is proved that the modular diagnosability property is preserved after using these reduction rules.

   b) While analyzing the local diagnosability of local modules, a new approach is proposed based on the Verifier Net (VN) approach in [Cab+12]. A new structure called Modified Reachability Graph (MRG) of the VN is developed. A sufficient and necessary condition for local diagnosability is given. The complexity of this approach is the same with that of the VN approach.

   c) A new approach for modular diagnosability analysis is proposed. The parallel composition of MRG and the Reachability Graph (RG) of the composed LPN module is built, in order to check the modular diagnosability property. A sufficient and necessary condition for modular diagnosability is proposed. The $\varepsilon-$reduction technique is used to simplify the structures before building the parallel composition, in order to reduce the combinatorial explosion problem. The complexity of this approach is polynomial and lower than other approaches in literature.

1.3 Manuscript's structure

This thesis is structured as follows:

- In Chapter 2, the problem statement of the diagnosability analysis is provided. The positioning of our works on monolithic diagnosability analysis and modular diagnosability analysis in presented. Then, some useful basic notions of this thesis are given.

- In Chapter 3, the literature review of the approaches for monolithic diagnosability analysis is given. Some automata-based approaches and PN-based approaches are analyzed in detail. Thus, some contributions on monolithic diagnosability analysis are proposed.

- In Chapter 4, the literature review of decentralized diagnosis, modular diagnosis and distributed diagnosis is presented. A new modular diagnosability analysis approach is proposed.

- In Chapter 5, some experimental evaluations are provided to test our proposed approaches. A manufacturing benchmark [Hos+13] and a $n-$multi track level crossing benchmark [Liu+16] are taken into consideration. The monolithic diagnosability and modular diagnosability of the two benchmarks are analyzed.
- In Chapter 6, the conclusions and some perspectives are given.

CHAPTER 2

PROBLEM STATEMENT AND POSITIONING OF THE WORKS

Contents

2.1 Problem statement

The first widely accepted DES-based fault diagnosis was proposed in [Sam+95] using automata. An automaton structure called "Diagnoser" is introduced for both on-line diagnosis and diagnosability analysis. However, this approach (called "diagnoser" approach) brings two principle problems for fault diagnosis: computational complexity and combinatorial explosion. The complexity of building a diagnoser is exponential in the number of the states of the automaton system. The diagnosability analysis is based on the enumeration of the state space of the diagnoser, which could lead to a combinatorial explosion problem. Because of these problems, the "diagnoser" approach is not applicable while dealing with a large-scale system. Afterwards, researchers focus on the Petri net (PN), which gives a compact representation of DES. The pioneer work of PN-based fault diagnosis was proposed in [Ush+98]. However, the computational complexity and combinatorial explosion problems still exist to a certain extent. To overcome these problems of

diagnosability analysis using automata and using PN, many techniques has been proposed in the literature. A literature review is given in Section 3.1.

Moreover, instead of analyzing the monolithic system, the architectures of decentralized diagnosis, modular diagnosis and distributed diagnosis have been proposed in the literature, in order to reduce the complexity of diagnosis. The literature review of these approaches is given in Section 4.1. The main objective is to achieve the same diagnosis performance with the monolithic approaches without building the monolithic diagnoser. However, before using these architectures for fault diagnosis, the corresponding diagnosability properties need to be verified (codiagnosability for decentralized architecture, modular diagnosability for modular architecture and monolithic diagnosability for distributed architecture). There does not exist a best approach for all kinds of DES system and each architecture has its own problems. This thesis focuses on the modular architecture in order to deal with computational complexity and combinatorial explosion problems of modular diagnosis in literature.

2.2 Positioning of the works

This thesis is in the framework of labeled Petri net (LPN) and deals with the diagnosability analysis of DES. The contributions of this thesis consists of two parts: (1) monolithic diagnosability analysis; (2) modular diagnosability analysis. The positioning of the works are discussed separately.

2.2.1 Monolithic diagnosability analysis

The diagnosability property needs to be analyzed at the design stage of the DES. The online diagnosis is executed if and only if the diagnosability of the DES is verified. However, there is no guaranty that the diagnosability property of system is initially fulfilled. The aim of this study is to propose some methods of engineering that allow iterating the diagnosability analysis. When a DES is determined to be non-diagnosable, the model of DES must be modified and the diagnosability needs to be reanalyzed until the DES is diagnosable.

Most of the approaches in literature (in Section 3.1) build the whole state space *a priori* and then analyze the diagnosability using the entirely constructed state space. The process of these approaches in shown in Figure 2.1. For most of approaches, the diagnosability analysis of a given LPN model is transferred to the study of its reachability graph (RG) or some kinds of modified RG (e.g. MBRG in [Cab+14]). Therefore, the whole RG is necessarily built. Then, a diagnoser is built based on the RG for the diagnosability analysis. If the LPN model is non-diagnosable, the model is modified and then the RG and the diagnoser of the modified model are built once again. The previous process is iterated until the modified LPN is diagnosable. These approaches are not favorable for industrial

use, because the state space of the monolithic model is rebuilt each time when the system is modified. For a large-scale LPN, these approaches are not efficient for iterating the diagnosability analysis and there exists combinatorial explosion problem.

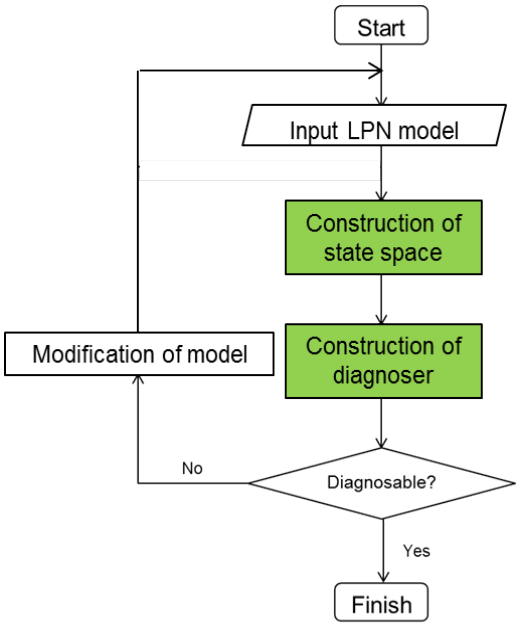


Figure 2.1 – Process of most approaches in literature

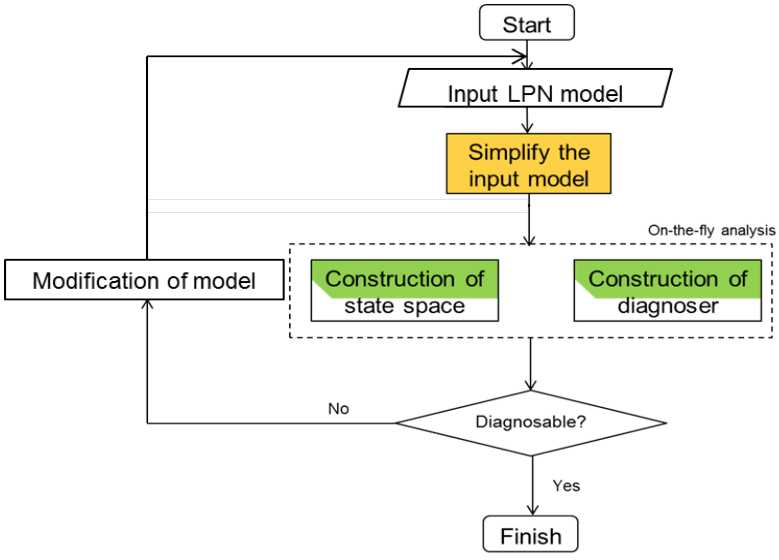


Figure 2.2 – Process of our approaches

In this thesis, to overcome the combinatorial explosion problem while iterating the diagnosability analysis, the process of the proposed technique is different. As it is shown in Figure 2.2, there are two different points:

1. Instead of building directly the RG of the given LPN model, the LPN model is simplified *a priori*: some reduction rules are proposed with the proof that the diagnosability property is preserved. By using these reduction rules, some transitions and places are reduced. The diagnosability analysis using the reduced LPN model leads to the same result of the diagnosability analysis of the initial LPN model. The state space of the reduced LPN model is smaller than that of the initial one so as to reduce the combinatorial explosion problem.

2. The on-the-fly diagnosability analysis is applied. The on-the-fly diagnosability analysis does not build entirely the state space *a priori*, nor the model for diagnosis: they are built on the fly and in parallel with some stop conditions, which stop building some of their branches. Along with their on-the-fly construction, the diagnosability is analyzed. When the condition of undiagnosability is satisfied, the result that the LPN model is not diagnosable, is immediately given. This technique shows that a part of state space could suffice for diagnosability analysis, particularly when the system is non-diagnosable, in order to reduce the memory cost and solve the combinatorial explosion problem. In this work, two structure properties of LPN are used: the minimal explanation and T-invariant, and the efficiency of the on-the-fly diagnosability analysis is improved.

2.2.2 Modular diagnosability analysis

For some large-scale systems which are more complex, the monolithic diagnosis is not feasible because the monolithic diagnoser is required i.e., it is impossible to build a monolithic diagnoser of the system because of the computational complexity and the combinatorial explosion problem. Many approaches such as decentralized diagnosis, modular diagnosis and distributed diagnosis are proposed to achieve the same diagnosis performance of the monolithic diagnosis without building the monolithic diagnoser.

In literature, the approaches of modular diagnosability are based on the automata models. For a modularly designed system (it is assumed that the communication between modules is via common events), if automata are used to model the system, it is not easy to give directly the monolithic model. Usually, the model of each module is generated, then the monolithic model can be obtained by building the parallel composition of the modules. However, for a modularly designed system, it is not favorable to build the monolithic model and analyze the monolithic diagnosability because of the combinatorial explosion problem. In this case, if the modular diagnosability of the system is fulfilled, the on-line diagnosis can be implemented by using only the local diagnoser of each module.

However, if the modularly designed system is modeled by Petri net (PN), the monolithic model may be directly given due to the advantage of PN (the concurrent processes are well represented by using PN). The monolithic diagnosability remains infeasible. In this thesis, we focus on system that is modeled by a collection of PN modules or by a

monolithic model that can be decomposed under certain conditions into a collection of PN modules. A new modular diagnosability verification is proposed based on the PN model by removing some assumptions of the approaches in the literature. The aim of this approach is to reduce the computational complexity and combinatorial explosion problem.

2.3 Basic notions

This section recalls some notions that will be used in the rest of this manuscript.

2.3.1 Automata

An automaton is a graphical device that is capable to describe state spaces and state transtions of a DES. A language according to well-defined rules can be represented by using an automaton. Particularly, one type of automata is widely used, which is called Final State Machine (or Final State Automaton).

A *deterministic* automaton model is denoted as a six-tuple $G = (X, \Sigma, \delta, \Gamma, x_0, X_m)$, where

- X is the finite set of states;
- Σ is the set of events;
- $\delta : X \times \Sigma \to X$ is the transition function. $\delta(x, e) = y$ denote that there is a transition labeled by event e from state x to state y;
- $\Gamma : X \to 2^{\Sigma}$ is the feasible event function (or active event function). $\Gamma(x)$ is called the feasible event set (or active event set) of G at state x. It is the set of all events e for which $\delta(x, e)$ is defined;
- x_0 is the initial state;
- $X_m \subseteq X$ is the set of marked states.

The automaton is said to be *deterministic* because δ is a function from $X \times \Sigma$ to X, namely, there canont be two transitions with the same event label out of a state. For the sake of convenience, δ is always extended from domain $X \times \Sigma$ to domain $X \times \Sigma^*$, where Σ^* is the *Kleene closure* of the set of events Σ.

For the sake of simplicity, unless specifically stated, the feasible event function Γ and the set of marked states X_m are omitted.

Example 1 *Let us consider the automaton $G = (X, \Sigma, \delta, x_0)$ shown in Figure 2.3.*

- *The set of states is* $X = \{1, 2, 3, 4, 5, 6\}$;
- *The set of events is* $\Sigma = \{a, b, c, f, u\}$;
- *The transition mapping is shown by following the arrows, e.g.* $\delta(1, c) = \{2\}$, $\delta(3, b) = \{2\}$ *and* $\delta(6, a) = \{6\}$;
- *The initial state is* 1.

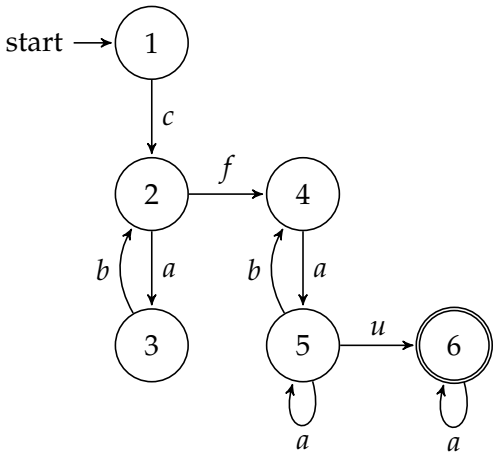


Figure 2.3 – An example of FSM [ZL13]

The behavior of a DES is described by the *prefix-closed language* $L(G) = \{s \in \Sigma^* | \ \delta(x_0, s) \text{ is defined}\}$. The language $L(G)$ represents all the directed paths that can be followed along the state transition diagram, starting from the initial state. $L(G)$ is a subset of Σ^*. The *post-language* of $L(G)$ after s is denoted as $L(G)/s = \{s' \in \Sigma^* | \ ss' \in L(G)\}$.

The accessible part of an automaton G is obtained by removing all states that are not accessible from the initial state x_0 and their related transitions. This operation is denoted as $Ac(G) = (X_{Ac}, \Sigma, \delta_{Ac}, x_0)$, where $X_{Ac} = \{x \in X | (\exists s \in \Sigma^*)[\delta(x_0, s) = x]\}$ and $\delta_{Ac} : X_{Ac} \times \Sigma \to X_{Ac}$ is the restricted transition function on X_{Ac}.

The coaccessible part of G is obtained by removing all states from which it is not possible to reach a marked state. This operation is denoted as $CoAc(G) = (X_{CoAc}, \Sigma, \delta_{CoAc}, x_{0_{CoAc}}, X_m)$, where $X_{CoAc} = \{x \in X | (\exists s \in \Sigma^*)[\delta(x_0, s) \in X_m]\}$, $\delta_{CoAc} : X_{CoAc} \times \Sigma \to X_{CoAc}$ is the restricted transition function on X_{CoAc} and $x_{0_{CoAc}} = x_0$ if $x_0 \in X_{CoAc}$ or $x_{0_{CoAc}}$ is undefined if $x_0 \notin X_{CoAc}$.

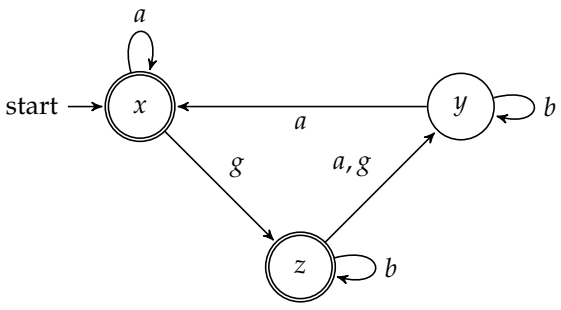
The product (or completely synchronous composition) of two given automata $G_1 = (X_1, \Sigma_1, \delta_1, \Gamma_1, x_{0,1}, X_{m,1})$ and $G_2 = (X_2, \Sigma_2, \delta_2, \Gamma_2, x_{0,2}, X_{m,2})$ is the automaton $G_1 \times G_2 := Ac(X_1 \times X_2, \Sigma_1 \cap \Sigma_2, \delta, \Gamma_{1\times 2}, (x_{0,1}, x_{0,2}), X_{m,1} \times X_{m,2})$, where

$$\delta((x_1, x_2), e)) := \begin{cases} (\delta_1(x_1, e), \delta_2(x_2, e)) & \text{if } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ \text{undefined} & \text{otherwise} \end{cases}$$

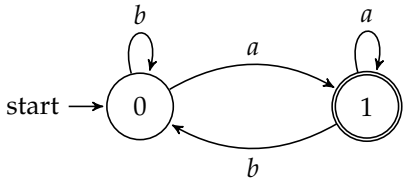
The parallel composition (or synchronous composition) of G_1 and G_2 is the automaton $G_1||G_2 := Ac(X_1 \times X_2, \Sigma_1 \cup \Sigma_2, \delta, \Gamma_{1||2}, (x_{0,1}, x_{0,2}), X_{m,1} \times X_{m,2})$, where

$$\delta((x_1, x_2), e)) := \begin{cases} (\delta_1(x_1, e), \delta_2(x_2, e)) & \text{if } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (\delta_1(x_1, e), x_2) & \text{if } e \in \Gamma_1(x_1) \backslash \Sigma_2 \\ (x_1, \delta_2(x_2, e)) & \text{if } e \in \Gamma_2(x_2) \backslash \Sigma_1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Example 2 *Given two automata (shown in Figure 2.4)* $G_1 = (X_1, \Sigma_1, \delta_1, \Gamma_1, x_{0,1}, X_{m,1})$ *and* $G_2 = (X_2, \Sigma_2, \delta_2, \Gamma_2, x_{0,2}, X_{m,2})$, *where* $X_1 = \{x, y, z\}$, $\Sigma_1 = \{a, b, g\}$, $X_{m,1} = \{x, z\}$, $X_2 = \{0, 1\}$, $\Sigma_2 = \{a, b\}$ *and* $X_{m,2} = \{1\}$. *The product automaton of* G_1 *and* G_2 *is shown in Figure 2.5 and the parallel composition of* G_1 *and* G_2 *is shown in Figure 2.6.*



(a) G_1



(b) G_2

Figure 2.4 – Two given automata G_1 and G_2

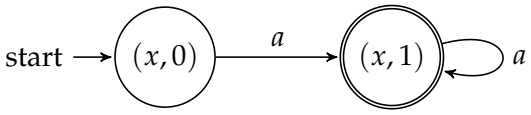


Figure 2.5 – The product automaton of G_1 and G_2

To deal with the automata-based diagnosis problems, the DES is model by an automaton containing normal and faulty behaviors. The set of events Σ are partitioned into two sets: $\Sigma = \Sigma_o \dot{\cup} \Sigma_u$ (the symbol $\dot{\cup}$ is used to represent that the two sets are disjoint), where Σ_o is the set of observable events and Σ_u is the set of unobservable events. Σ_f is denoted as the set of fault events to be diagnosed. Σ_f can be partitioned into different fault classes: $\Sigma_f = \Sigma_{f_1} \dot{\cup} \Sigma_{f_2} \dot{\cup} \cdots \Sigma_{f_k}$. The partition can be denoted by Π_f.

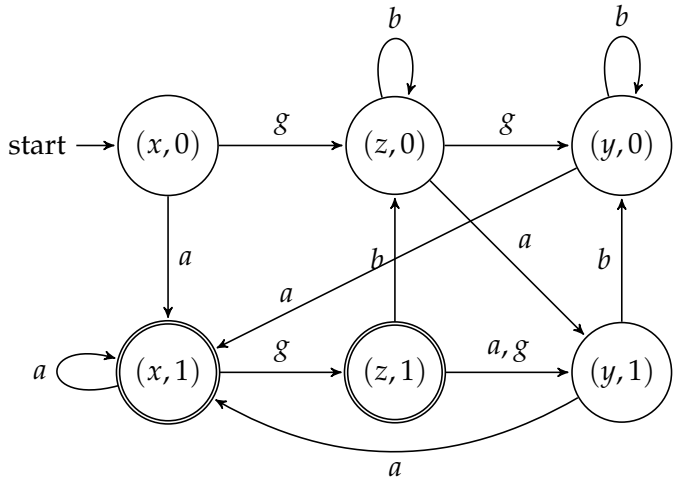


Figure 2.6 – The parallel composition of G_1 and G_2

The projection operator $P_{o,e} : \Sigma^* \rightarrow \Sigma_o^*$ is defined as follows:

$$\begin{cases} P_{o,e}(\varepsilon) = \varepsilon & (\varepsilon \text{ is an empty string}) \\ P_{o,e}(e) = \varepsilon & if\ e \in \Sigma_u \\ P_{o,e}(e) = e & if\ e \in \Sigma_o \\ P_{o,e}(se) = P_{o,e}(s)P_{o,e}(e) & where\ s \in \Sigma^*\ and\ e \in \Sigma \end{cases}$$

In other terms, by using the projection operator, the unobservable events in one sequence are removed and the obtained sequence contains only observable events.

The inverse projection operator $P_{o,e}^{-1}$ for $\forall y \in \Sigma_o^*$ is defined as

$$P_{o,e}^{-1}(y) = \{s \in \Sigma^* |\ P_{o,e}(s) = y\}$$

Example 3 *Let us consider again the automaton $G = (X, \Sigma, \delta, x_0)$ shown in Figure 2.3. Assuming that $\Sigma_o = \{a, b, c\}$ and $\Sigma_u = \{u, f\}$. For $s \in L(G)$ where $s = cabfau$, $P_{o,e}(s) = caba$.*

Normally, an automaton with $\varepsilon-$transition as introduced before is not necessarily deterministic, because there exist unobservable events, which cause the uncertainty of the states reached after firing un observable event. However, for any automaton, we can build an observer $Obs(G)$ which is a deterministic automaton. An observer is built by regrouping all the states that can be reached after firing an observable events.

Example 4 *Let us consider again the automaton $G = (X, \Sigma, \delta, x_0)$ shown in Figure 2.3. The observer $Obs(G)$ is built in Figure 2.7.*

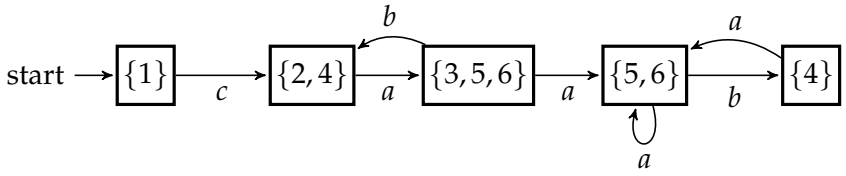


Figure 2.7 – The observer $Obs(G)$ of the automaton in Figure 2.3

2.3.2 Petri Nets (PNs)

Petri nets (PN) were invented in the PhD thesis of Carl Adam Petri in 1962 [Pet62]. The theory of PNs has been developed in the following years [Sil12]. Comparing to automata, PNs have some advantages:

1. PNs give a graphical and mathematical representation of DESs. The analysis of PNs can be proposed by using the graphical structure or mathematical calculation;
2. The states of PNs are represented by the distribution of tokens in places, which gives a more compact representation of DESs and allows them representing an infinite state space by a finite graphical structure;
3. The concurrent processes are well represented by using PNs., in order to model a system with shared resources;
4. The composition and decomposition operations can be more conveniently done by using PNs. PNs give a more natural structure of systems. The communication between modules can be modeled by common places or transitions. The process of decomposing a modular system is more intuitive using PNs

The definition of a PN is given as follows:

Definition 1 *A PN is defined as a 4-tuple $N = (P, T, Pre, Post)$, where:*

- *P is a finite set of places. A place is represented by a circle;*
- *T is a finite set of transitions. A transition is represented by a bar or box;*
- *$Pre : P \times T \to \mathbb{N}$ is the pre-incidence matrix that represents the weight of the arcs from places to transitions in the PN graph;*
- *$Post : P \times T \to \mathbb{N}$ is the post-incidence matrix that represents the weight of the arcs from transitions to places in the PN graph.*

Example 5 *An example of PN is shown in Figure 2.8.*

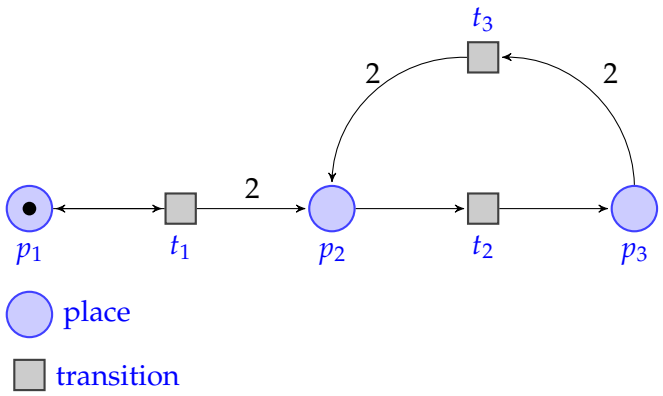


Figure 2.8 – An example of PN

- *The set of places is* $P = \{p_1, p_2, p_3\}$;
- *The set of transitions is* $T = \{t_1, t_2, t_3\}$;
- *The pre-incidence matrix is:*

$$Pre = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} \end{array};$$

- *The post-incidence matrix is:*

$$Post = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 2 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix} \end{array};$$

A state of a PN is called a "*Marking*", which is a vector $M \in \mathbb{N}^{|P|}$ that assigns a non-negative integer to each place. A marking is the distribution of tokens in the places of the PN. The tokens are represented by the dots in the places. We denote that (N, M_0) is a marked PN with the initial marking M_0.

The behavior of a PN model is represented by the redistribution after firing a transition. A transition $t \in T$ is enabled at a marking M iff $M \geq Pre(\cdot, t)$. The set of enabled transitions is denoted as $T_{enabled}(M) = \{t | t \in T, M \geq Pre(\cdot, t)\}$.

An enabled transition t at a marking M can be fired. The obtained marking M' is computed by $M' = M + Post \cdot \overrightarrow{t} - Pre \cdot \overrightarrow{t}$, where $\overrightarrow{t} \in \{0,1\}^{|T|}$ is the elementary vector of transition t, in which only the column associated to the transition t is equal to 1. The incidence matrix C is denoted as $C = Post - Pre$. Therefore, the obtained marking M' can be computed by $M' = M + C \cdot \overrightarrow{t}$. The marking M' is said to be a *reachable* marking from M, and it is denoted as $M[t > M'$.

Example 6 *In the PN example in Figure 2.8:*

- *The initial marking is:*

$$M_0 = \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix};$$

- *The incidence matrix is:*

$$C = Post - Pre = \begin{matrix} \\ p_1 \\ p_2 \\ p_3 \end{matrix} \begin{matrix} t_1 & t_2 & t_3 \\ \end{matrix} \begin{bmatrix} 0 & 0 & 0 \\ 2 & -1 & 2 \\ 0 & 1 & -2 \end{bmatrix};$$

- *At the initial marking, the set of enabled transitions is $T_{enabled}(M_0) = \{t_1\}$. The elementary vector of transition t_1 is:*

$$\overrightarrow{t_1} = \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix};$$

- *At the initial marking, after firing the transition t_1, the obtained marking is:*

$$M_1 = M_0 + C \cdot \overrightarrow{t_1} = \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix};$$

A sequence of transitions $\sigma = t_1 t_2 \cdots t_k$ is firable (executable) at marking M, if $M[t_1 > M_1[t_2 > \cdots M_{k-1}[t_k >$ and it is denoted as $M[\sigma >$. σ^j is the $j-$th transition in σ. The reached marking M' is computed by the *state equation*: $M' = M + C \cdot \pi(\sigma)$, where $\pi(\sigma) = \sum_{i=1}^{k} \overrightarrow{t}_i$ is the *firing vector* of σ, where $\pi(\sigma) \in \mathcal{N}^{|T|}$ and the value of the row associated to transition t_i is equal to the number of occurrence of t_i in σ. For a firing vector $\overrightarrow{y} = \pi(\sigma)$, $\overrightarrow{y}(t) = k$ means that transition t is contained k times in σ. $T_M(\overrightarrow{y}) = \{t \in T : \textit{the cardinality of } t \textit{ in } T_M(\overrightarrow{y}) \textit{ is } \overrightarrow{y}(t)\}$ is defined as the multiset of transitions corresponding to the firing vector $\overrightarrow{y}$. For example, the set of transitions contained by the firing vector $\overrightarrow{y}(t) = [1\ 2\ 0]^\tau$ is $T_M(\overrightarrow{y}) = \{t_1, t_2, t_2\}$. A transition $t_i \in \sigma$, iff $\overrightarrow{y} = \pi(\sigma)$ and $\overrightarrow{y}(t_i) \neq 0$; otherwise, $t_i \notin \sigma$. $\sigma\lambda = \sigma$, $\forall \sigma \in T^*$, where λ denotes the empty transition.

A marking M is *reachable* in (N, M_0) iff a sequence σ exists such that $M_0[\sigma > M$. The set of all the reachable markings from M_0 is denoted by $R(N, M_0)$ and called the reachability set of (N, M_0).

Example 7 *In the PN example in Figure 2.8:*

- *At the initial marking M_0, the sequence of transitions $\sigma = t_1t_2t_2$ is a firable, i.e., $M_0[\sigma >$. The firing vector of σ is:*

$$\overrightarrow{y} = \pi(\sigma) = \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix};$$

- *At the initial marking M_0, after firing the sequence $\sigma = t_1t_2t_2$, the obtained marking is:*

$$M' = M_0 + C \cdot \pi(\sigma) = \begin{matrix} p_1 \\ p_2 \\ p_3 \end{matrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix};$$

- *The marking M' is reachable in (N, M_0), because $M_0[\sigma > M'$, i.e., $M' \in R(N, M_0)$.*

A PN (N, M_0) is said to be *bounded* (or *m-bounded*) if there exists a positive number m such that $\forall M \in R(N, M_0),\ \forall p \in P,\ M(p) \leq m$. The reachability space of a bounded PN is finite and it is represented by a graph called reachability graph (RG). A PN is call *safe*, if it is 1-*bounded*.

If the number of tokens in one or more places can be arbitrarily large, the PN is *unbounded* and the coverability graph (CG) is used to represent the infinite state space.

Definition 2 *Given a PN (N, M_0), a transition t is:*

- dead*: if there does not exist a reachable marking $M \in R(N, M_0)$ that enables t;*
- semi-live*: if there exists at least one reachable marking $M \in R(N, M_0)$ that enables t;*
- live*: if for each reachable marking $M \in R(N, M_0)$, t is semi-live in (N, M);*

A PN (N, M_0) is live *if each transition $t \in T$ is live. In other words, a PN is live if, from any marking in $R(N, M_0)$, it is possible to fire any transition by progressing through some further firing sequences. A* deadlock *occurs at marking M if no transition can be enabled at M.*

A T-invariant of PNs is a positive integer solution of homogeneous equation: $C \cdot \overrightarrow{\Omega} = \overrightarrow{0}$, where $\overrightarrow{\Omega}$ is a firing vector as defined before.

For two markings $M_i, M_j \in R(N, M_0)$, $M_i > M_j$ iff $\forall p \in P, M_i(p) > M_j(p)$.

Definition 3 *A sequence $\sigma \in T^*$ is called* repetitive *if there exists a marking $M_1 \in R(N, M_0)$ s.t. $M_1[\sigma > M_2[\sigma > \cdots$, i.e., if it can fire infinite times starting from M_1. A repetitive sequence is called:*

- stationary*: if $M_{i+1} = M_i$ for all $i = 1, 2 \cdots$. A stationary repetitive sequence is associated to a T-invariant.*
- increasing*: if $M_{i+1} > M_i$ for all $i = 1, 2 \cdots$. If there exists an increasing repetitive sequence, the PN system is unbounded.*

Example 8 *In the example of PN in Figure 2.8.*

- *This PN is* unbounded *because there exists an increasing repetitive sequence. The number of tokens in p_2 can be infinite if the transition t_1 is fired infinite times at M_0.*
- *This PN is* live*. In addition, each transition is* live*.*
- *There exist a T-invariant*

$$\overrightarrow{\Omega} = \begin{matrix} t_1 \\ t_2 \\ t_3 \end{matrix} \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix};$$

  because $C \cdot \overrightarrow{\Omega} = \overrightarrow{0}$
- *$\sigma_1 = t_1$ is an increasing repetitive sequence at marking M_0, because $M_0[\sigma_1 > M_1$ and $M_1 > M_0$. $\sigma_2 = t_2 t_2 t_3$ is a stationary repetitive sequence at marking M_1 because $M_1[\sigma_2 > M_1$ and this stationary repetitive sequence is associated to the T-invariant $\overrightarrow{\Omega}$.*

Definition 4 *An LPN, an extension of PN, is a tuple $LPN = (N, M_0, \Sigma, \mathcal{L})$,*

- *(N, M_0) is a marked PN;*
- *Σ is a finite set of events;*
- *$\mathcal{L}: T \to \Sigma$ is the transition labeling function which assigns a label to each transition.*

In event-based diagnosis of DESs using LPN models, the set of transitions is partitioned into two disjoint sets, $T = T_o \dot{\cup} T_u$, where T_o is the set of observable transitions, and T_u is the set of unobservable transitions. The label of an observable transition can be observed when it fires. The fault transitions are unobservable. The set of unobservable transitions is partitioned into two disjoint sets, $T_u = T_f \dot{\cup} T_{reg}$, where T_f includes all fault transitions, while $T_{reg} = T_u \backslash T_f$ is the set of regular unobservable transitions. The set T_f can be further

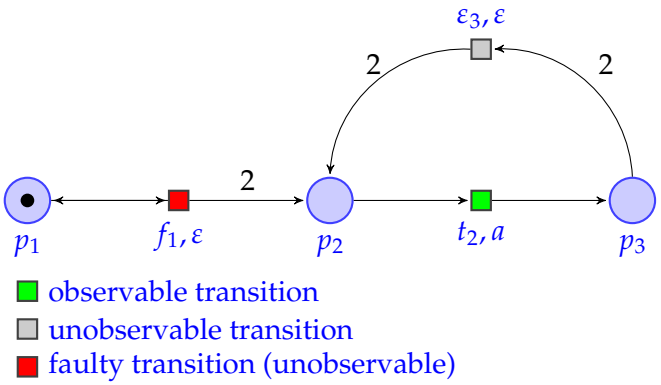


Figure 2.9 – An example of LPN

partitioned into k different subsets T_f^i, where $i = 1, \cdots, k$, represents different classes of faulty transitions.

$P_{o,t}: T^* \rightarrow T_o^*$ is the projection which removes the unobservable transitions in a sequence $\sigma \in T^*$ and $P_{u,t}: T^* \rightarrow T_u^*$ is the projection which removes the observable transitions of $\sigma \in T^*$.

The set of events is $\Sigma = \Sigma_o \dot{\cup} \{\varepsilon\}$. Σ_o is the set of observable events that are associated with observable transitions and the label of all unobservable transitions is ε. The same label could be shared by different transitions. The labeling function can be extended to $\mathcal{L}: T^* \rightarrow \Sigma^*$. The inverse projection operator $\mathcal{L}^{-1}$ is defined by $\mathcal{L}^{-1}(\omega) = \{\sigma \in T^* \mid \mathcal{L}(\sigma) = \omega\}$. $\omega = \omega_1\omega_2 \cdots \omega_n$ is the concatenation of $\omega_1, \omega_2, \cdots, \omega_n$ with $\omega_1, \omega_2, \cdots, \omega_n \in \Sigma^*$.

The language generated by LPN is $L(LPN) = \{\mathcal{L}(\sigma) \in \Sigma^* \mid \sigma \in T^*, M_0 [\sigma >\}$. Let ω be an observed word, the set of firing sequences of transitions corresponding to ω is $\mathcal{FS}(\omega) = \{\sigma \in L(LPN) | \mathcal{L}(\sigma) = \omega\}$.

Example 9 *Let us consider the LPN model $LPN = (N, M_0, \Sigma, \mathcal{L})$ in Figure 2.9.*

- *The set of events is $\Sigma = \Sigma_o \dot{\cup} \{\varepsilon\}$, where $\Sigma_o = \{a\}$;*
- *$\mathcal{L}(f_1) = \mathcal{L}(\varepsilon_3) = \varepsilon$ and $\mathcal{L}(t_2) = a$.*

Definition 5 *Given two LPN models $LPN_1 = (N_1, M_{0,1}, \Sigma_1, \mathcal{L}_1)$ and $LPN_2 = (N_2, M_{0,2}, \Sigma_2, \mathcal{L}_2)$, where $N_1 = (P_1, T_1, Pre_1, Post_1)$, $N_2 = (P_2, T_2, Pre_2, Post_2)$, $T_1 = T_{o1} \dot{\cup} T_{u1}$, $T_2 = T_{o2} \dot{\cup} T_{u2}$ and $(P_1 \cup T_1) \cap (P_2 \cup T_2) = \emptyset$. The parallel composition of LPN_1 and LPN_2 is denoted as $LPN = LPN_1 || LPN_2$, where $LPN = (N, M_0, \Sigma, \mathcal{L})$, $N = (P, T, Pre, Post)$ and $P = P_1 \cup P_2$. The transitions in T are defined as follows:*

1. *For any pair of transitions $t_i \in T_{o1}$ and $t_j \in T_{o2}$ s.t. $\mathcal{L}_1(t_1) = \mathcal{L}_2(t_2)$, add a transition $t_{i,j} \in T$. For all $p \in P_1$, let $Pre(p, t_{i,j}) = Pre_1(p, t_i)$ and $Post(p, t_{i,j}) = Post_1(p, t_i)$;*

for all $p' \in P_2$, *let* $Pre(p', t_{i,j}) = Pre_2(p', t_i)$ *and* $Post(p', t_{i,j}) = Post_2(p', t_i)$. *Label the transition* $t_{i,j}$ *with* $\mathcal{L}_1(t_i)$; *(The transition* $t_{i,j}$ *is called a* shared transition.*)*

2. *For all the other transitions* $t_k \in T_1$, *add a transition* $t'_k \in T$. *For all* $p \in P_1$, *let* $Pre(p, t'_k) = Pre_1(p, t_k)$ *and* $Post(p, t'_k) = Post_1(p, t_k)$; *for all* $p' \in P_2$, *let* $Pre_2(p', t'_k) = Post_2(p', t'_k) = 0$. *Label the transition* t'_k *with* $\mathcal{L}_1(t_k)$;

3. *For all the other transition* $t_h \in T_2$, *add a transition* $t'_h \in T$. *For all* $p \in P_1$, *let* $Pre_1(p, t'_h) = Post_1(p, t'_h) = 0$; *for all* $p' \in P_2$, *let* $Pre(p, t'_h) = Pre_2(p, t_h)$ *and* $Post(p, t'_h) = Post_2(p, t_h)$. *Label the transition* t'_h *with* $\mathcal{L}_2(t_h)$.

Example 10 *Given two LPN models in Figure 2.10. The parallel composition of the two LPN is shown in Figure 2.11.*

(a) LPN_1 (b) LPN_2

Figure 2.10 – Two given LPN: LPN_1 and LPN_2

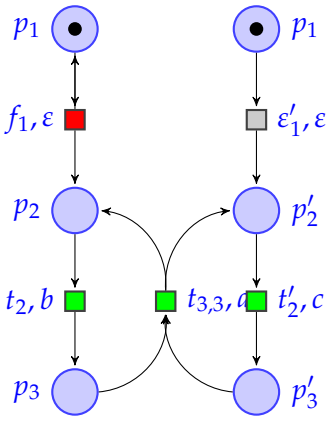


Figure 2.11 – The parallel composition of LPN_1 and LPN_2: $LPN = LPN_1||LPN_2$

CHAPTER 3

MONOLITHIC DIAGNOSABILITY ANALYSIS USING LPN

Contents

This chapter deals with the monolithic diagnosability analysis, which is the classic diagnosability analysis as it was originally defined in [Sam+95; Sam+96]. In the literature, the monolithic diagnosability is analyzed both in the framework of automata and that of Petri nets (PNs). This thesis focuses on PNs which have advantages for modeling DES. Precisely, we use Labeled Petri nets (LPN) as input models. This chapter stars with a literature review of the approaches for diagnosability analysis and then presents new contributions.

Section 3.2.1 presents some reduction rules that preserve the diagnosability property of the system (publication on topic [Li+16a; Li+17d]). Section 3.2.2 gives a new sufficient condition for *safe* and *live* LPN. Section 3.2.3 and Section 3.2.4 improve the on-the-fly diagnosability analysis for *bounded* LPN by using minimal explanations and T-invariants

(publications on topic [Li+15a; Li+15b; Li+15c; Li+17a]). Section 3.2.5 proposes the on-the-fly diagnosability analysis for both *bounded* and *unbounded* LPN (publication on topic [Li+16b; Li+17b]).

3.1 Literature review

The diagnosability was initially defined in the framework of automata and regular languages. Afterwards, researches focused on PNs which give a more expressive and compact representation of DESs. The critical problems of diagnosability analysis are combinatorial explosion and computational complexity. This literature review distinguishes automata-based approaches and PNs-based approaches and discusses the combinatorial explosion and computational complexity of the approaches.

3.1.1 Automata-based approaches

The definition of diagnosability is proposed in the framework of automata. It is assumed that the language of automata is *prefix-closed* and *live*.

Definition 6 *[Sam+95] A prefix-closed and live language $L(G)$ is said to be diagnosable w.r.t. the projection $P_{o,e}$ and w.r.t. the partition Π_f on Σ_f if the following condition holds:*

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})[\forall s \in \Psi(\Sigma_{F_i})][\forall r \in L(G)/s]$$

$$[|r| > n_i \Rightarrow D]$$

where $\Psi(\Sigma_{F_i})$ is the set of all the traces in $L(G)$ that end in a faulty event belonging to the class Σ_{F_i} and the condition D is:

$$\omega \in [P_{o,e}^{-1} P_{o,e}(sr)] \cap L(G) \Rightarrow \Sigma_{F_i} \in \omega$$

In other words, the diagnosability requires that after the occurrence of a fault, in a finite delay (after finite number of observable events), the fault can be detected based on distinct observations.

Definition 7 *[CL07] Unobservable event f is* not diagnosable *in live language $L(G)$, if there exist two strings s_N and s_F, such that:*

1. *s_F contains f and s_N does not;*
2. *s_F is arbitrarily long after the occurrence of f;*
3. *$P_{o,e}(s_N) = P_{o,e}(s_F)$.*

When no such pair of strings exists, f is said to be diagnosable *in $L(G)$.*

In other words, the unobservable event f is diagnosable if there is no pair of sequences with the same observation: one contains a fault and can be arbitrarily long after the fault; the second one does not contain a fault.

3.1.1.1 Diagnoser approach

The classic diagnoser approach is referred as the pioneer study on diagnosability analysis of DES. For a given automaton $G = (X, \Sigma, \delta, x_0)$, a diagnoser is an FSM defined as

$$G_d = (Q_d, \Sigma_0, \delta_d, q_0)$$

where Q_d, Σ_0, δ_d and q_0 have the usual definition. Each state of the diagnoser $q_i \in Q_d$ is a subset of $X \times \{N, F\}$, where N and F are fault tags: N denotes that the state is reached by firing a sequence without any fault and F denotes that the state is reached by firing a sequence with a fault (If the fault belongs to the fault class Σ_i, the tag is denoted as F_i). The initial state of the diagnoser is $q_0 = \{(x_0, N)\}$.

In order to build the diagnoser of a given automaton G, the generator ([Sam+95]) of the given automaton $G' = (X_o, \Sigma_o, \delta_{G'}, x_0)$ must be built by removing transition labeled by the unobservable event. G' is the $\varepsilon-$*reduced* automaton of G.

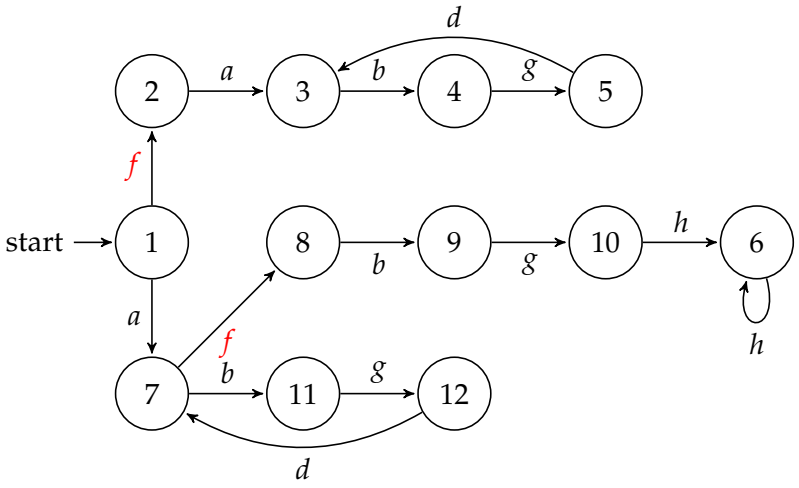


Figure 3.1 – An example of automaton G

Example 11 *Let us consider the automaton G in Figure 3.1. $\Sigma_o = \{a, b, d, g, h\}$ and $\Sigma_u = \Sigma_f = \{f\}$. The $\varepsilon-$reduced automaton of G is G' shown in Figure 3.2.*

The diagnoser is built based on the structure of G'. From the initial state (x_0, N), the following state of the diagnoser contains all the possible states after firing an observable event. The principle idea is based on *state estimation*.

Example 12 *The diagnoser of the automaton G in Figure 3.1 is shown in Figure 3.3.*

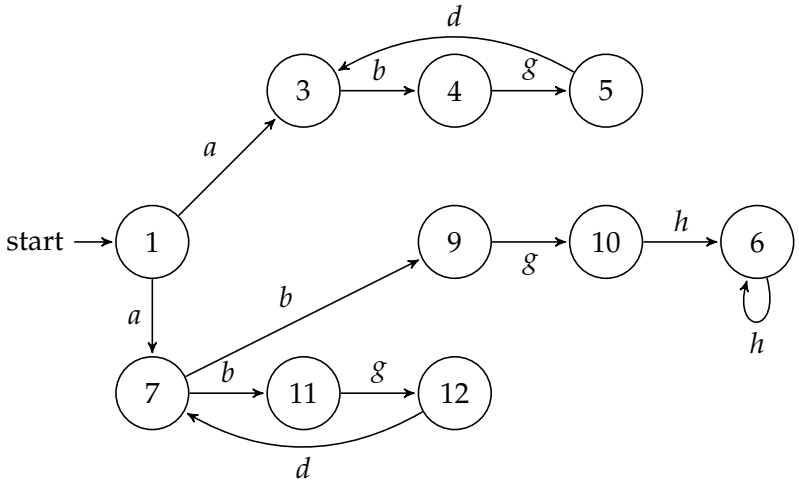


Figure 3.2 – The $\varepsilon-$reduced automaton G' of the automaton G in Figure 3.1

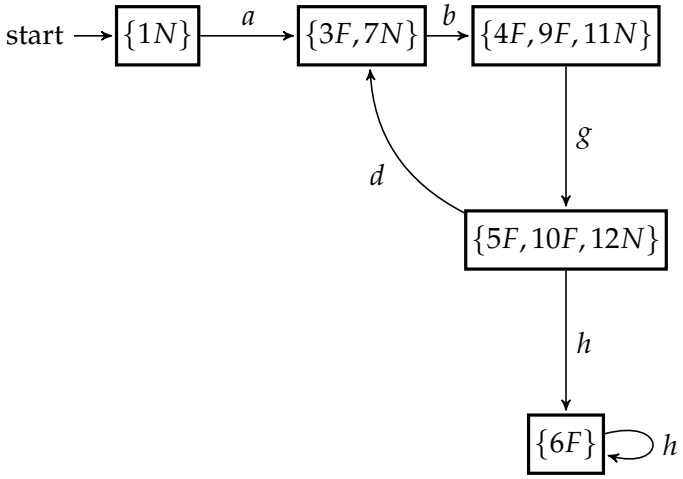


Figure 3.3 – The diagnoser G_d of the automaton G in Figure 3.1

A state in the diagnoser $q \in Q_d$ is said to be:

- Normal: if $\forall(x,l) \in q, l = N$ (e.g. the initial state $\{1N\}$ of the diagnoser in Figure 3.3);
- F-certain: if $\forall(x,l) \in q, l = F$ (e.g. the state $\{6F\}$of the diagnoser in Figure 3.3);
- F-uncertain: if $\exists(x,l)(x',l') \in q, (l = N) \wedge (l' = F)$ (e.g. the state $\{3F,7N\}$of the diagnoser in Figure 3.3).

In the diagnoser of a given automata G, a cycle formed by F-uncertain states is called an indeterminate cycle (the formal definition is presented in [Sam+95]), if there exists two corresponding cycles in G':

- The first one in G' is formed by the states labeled by N in the F-uncertain cycle;

– The second one is formed by the states labeled by F in the F-uncertain cycle;

The sufficient and necessary condition for diagnosability is:

Theorem 1 *A system is diagnosable iff there does not exist any F-indeterminate cycle.*

Example 13 *According to Theorem 1, the automaton G in Figure 3.1 is not diagnosable, because there exists an indeterminate cycle. The F-uncertain cycle in G_d is:*

$$3F, 7N \xrightarrow{b} 4F, 9F, 11N \xrightarrow{g} 5F, 10F, 12N \xrightarrow{d} 3F, 7N \cdots$$

The two corresponding cycles in G' are:

$$Normal\ cycle : 7 \xrightarrow{b} 11 \xrightarrow{g} 12 \xrightarrow{d} 7 \cdots$$

$$Faulty\ cycle : 3 \xrightarrow{b} 4 \xrightarrow{g} 5 \xrightarrow{d} 3 \cdots$$

Example 14 *Let us consider the automata G in Figure 3.4. The $\varepsilon-$reduced automaton G' is shown in Figure 3.5. The diagnoser of G is shown in Figure 3.6. The system is diagnosable, because there does not exist any indeterminate cycle. There exists an F-uncertain cycle in the diagnoser:*

$$3F, 7N \xrightarrow{b} 4F, 9F, 11N \xrightarrow{g} 5F, 10F, 12N \xrightarrow{d} 3F, 7N \cdots$$

However, there is only one corresponding normal cycle in G':

$$Normal\ cycle : 7 \xrightarrow{b} 11 \xrightarrow{g} 12 \xrightarrow{d} 7 \cdots$$

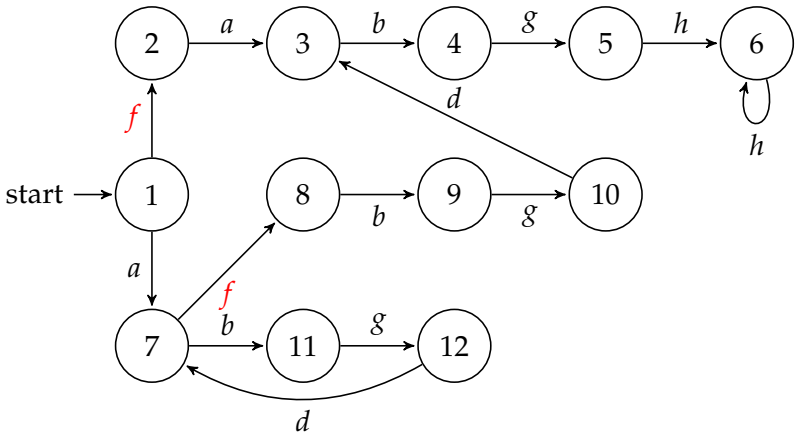


Figure 3.4 – An example of automaton G

It is worth noticing that there is a new version of diagnoser approach in [CL07]. The new version is obtained by $G_d = Obs(G||A_{label})$, where A_{label} is called a "label automaton" (shown in Figure 3.7) and denoted as $A_{label} = (\{N, F\}, \{f\}, \Sigma_f, N)$.

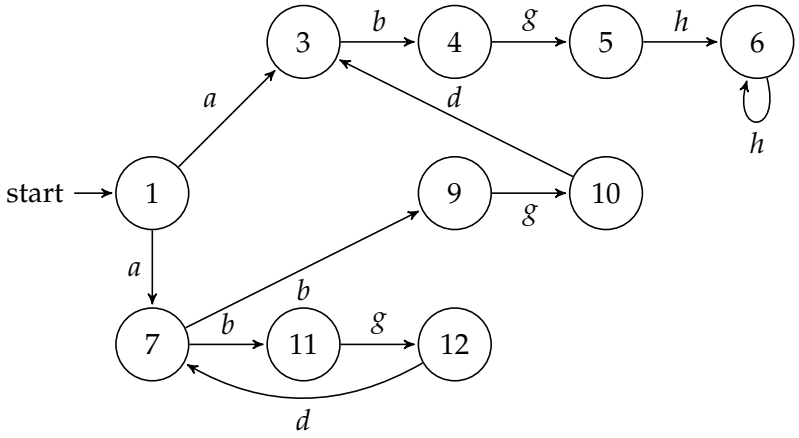


Figure 3.5 – The $\varepsilon-$reduced automaton G' of the automaton G in Figure 3.1

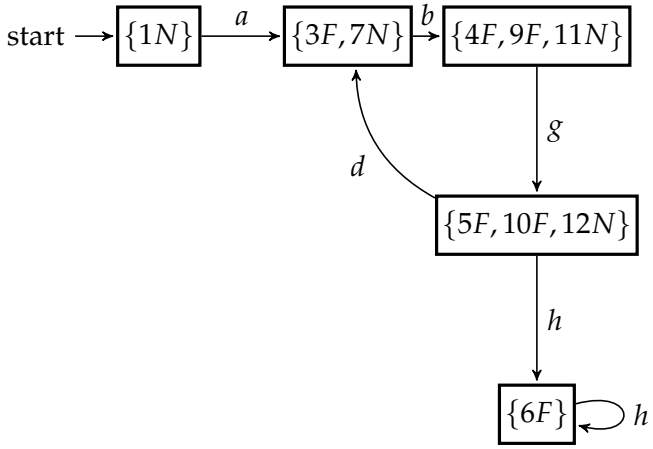


Figure 3.6 – The diagnoser G_d of the automaton G in Figure 3.4

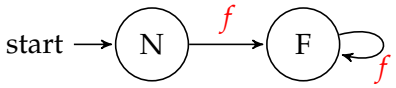


Figure 3.7 – The label automaton A_{label}

Example 15 *Let us consider again the automata G in Figure 3.1. The new version of diagnoser in [CL07] is shown in Figure 3.8. The initial state is not* $\{1N\}$ *but* $\{1N, 2F\}$ *where* $2F$ *is obtained by firing the unobservable fault* f *from* $1N$.

The difference between the two versions of diagnoser is the order to treat the unobservable event. The state of diagnoser in [Sam+95] contains only the states reached after firing an observable event e. The state of diagnoser in [CL07] contains the states reached by firing an observable event e and the states reached by firing all possible unobservable events after e.

The diagnoser approach is the first study for diagnosability analysis. The combinatorial explosion problem exists if the whole diagnoser is built. The complexity of building a

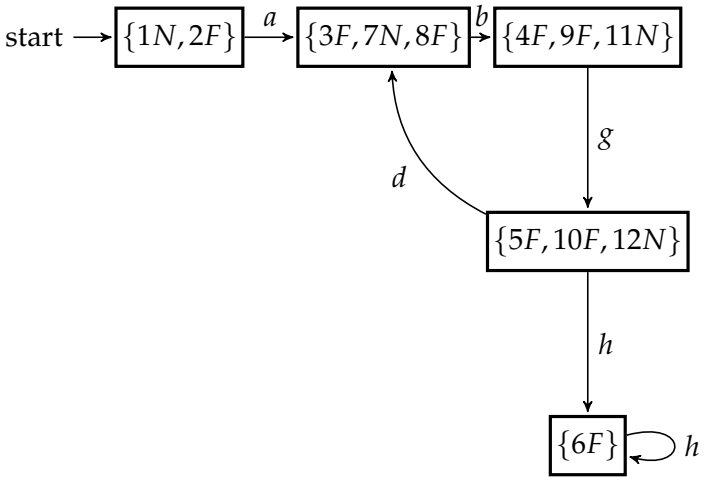


Figure 3.8 – The new version of diagnoser G_d of the automaton G in Figure 3.1

diagnoser is exponential. Assuming that the number of states of the given automaton is $|X|$ and the number of the fault classes is $|\Pi_f|$. The complexity of the diagnoser approach is $2^{|X|\times 2^{|\Pi_f|}}$.

3.1.1.2 Twin-plant approach

In order to reduce the complexity for diagnosability analysis, the twin-plant approach was proposed [Jia+01]. The main idea is based on building the parallel composition of the systems. This section introduces the Twin-plant approach for a given automaton $G = (X, \Sigma, \delta, x_0)$ and the projection operator $P_{o,e}$ in the following steps:

- Build a nondeterministic FSM $G_o = (X_o, \Sigma_o, \delta_o, x_0^o)$ with the language $L(G_o) = P_{o,e}(L(G))$, where:
  - $X_o = \{(x, l)|x \in X_1 \cup \{x_o\}$, l is the fault tag $\}$. $X_1 = \{x \in X|\exists(x', e, x) \in \delta, P_{o,e}(e) \neq \varepsilon\}$ is the set of states in G that can be reached after firing an observable event. l contains the fault information and l^i denotes the fault tag w.r.t. the fault class Σ_f^i;
  - Σ_o is the set of observable events;
  - $\delta_o \subseteq X_o \times \Sigma_o \times X_o$ is the set of transitions;
  - $x_o^0 = (x_0, \emptyset) \in X_o$ is the initial state.
- Build the twin-plant $Twin(G) = (G_o \parallel G_o)$, the parallel composition of G_o with itself. $Twin(G) = (X_{Twin}, \Sigma_o, \delta_{Twin}, x_{Twin}^0)$, where:
  - $X_{Twin} = \{(x_1^o, x_2^o)|x_1^o, x_2^o \in X_o\}$ is the set of states;
  - Σ_o is a set of observable events;
  - $\delta_{Twin} \subseteq X_{Twin} \times \Sigma_o \times X_{Twin}$ is the set of transitions;

- $x^0_{Twin} = (x^o_0, x^o_0) \in X_o$ is the initial state.

– Check whether there exists a cycle in $Twin(G)$, e.g. $(x^1_{Twin}, e_1, x^2_{Twin}, \cdots, x^n_{Twin}, e_n, x^1_{Twin})$, $n \geq 1, x^i_{Twin} = ((x^i_1, l^i_1)(x^i_2, l^i_2)), i \in \{1, 2, \cdots n\}$ such that $l^i_1 \neq l^i_2$. If $l^i_1 \neq l^i_2$, the system is not diagnosable. Otherwise the fault is diagnosable.

The principle idea is that if such a cycle presented above is found (e.g. $l^i_1 \neq l^i_2$), this cycle implies two cycles in G_o (in G as will because $L(G_o) = P_{o,e}(L(G))$), such that: one cycle formed by the states labeled by F_k (the fault in Σ^k_f has occurred); the other one formed by the normal states w.r.t. Σ^k_f. According to the definition of diagnosability, the system is not diagnosable w.r.t. Σ^k_f.

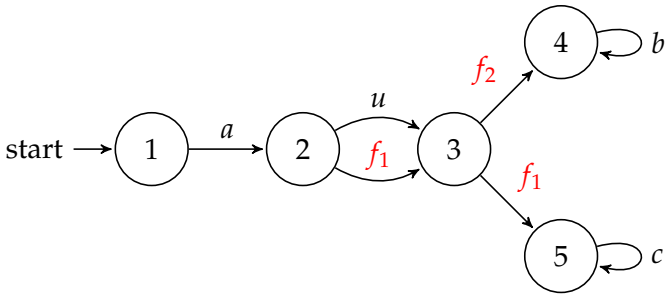


Figure 3.9 – An example of automaton G

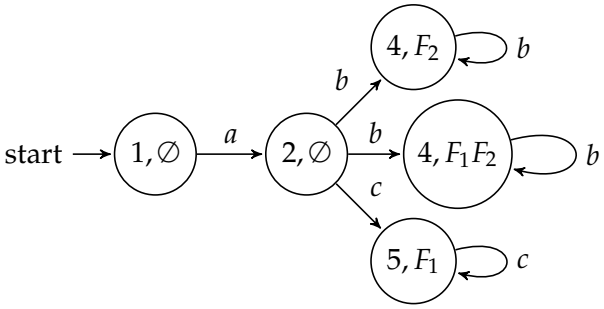


Figure 3.10 – The nondeterministic automaton G_o of G in Figure 3.9

Example 16 *Let us consider the automata G in Figure 3.9. $\Sigma_o = \{a, b, c\}$ and $\Sigma_u = \{u, f_1, f_2\}$. $\Sigma_f = \Sigma^1_f \cup \Sigma^2_f$, where $\Sigma^1_f = \{f_1\}$ and $\Sigma^2_f = \{f_2\}$. The nondeterministic automaton of G is shown in Figure 3.10. The twin-plant of G is shown in Figure 3.11. The system is not diagnosable regarding the fault class Σ^1_f, because there exists a self-loop at the state $\{(4, F_2), (4, F_1F_2)\}$. However, the system is diagnosable regarding Σ^2_f.*

The main advantage of the twin-plant approach is the reduction of complexity. The complexity of twin-plant approach is $|X|^4 \times |\Sigma_o| \times |\Pi_f|$, where $|X|$ is the number of states in the given automaton, $|\Sigma_o|$ is the number of observable events and $|\Pi_f|$ is the number of fault classes.

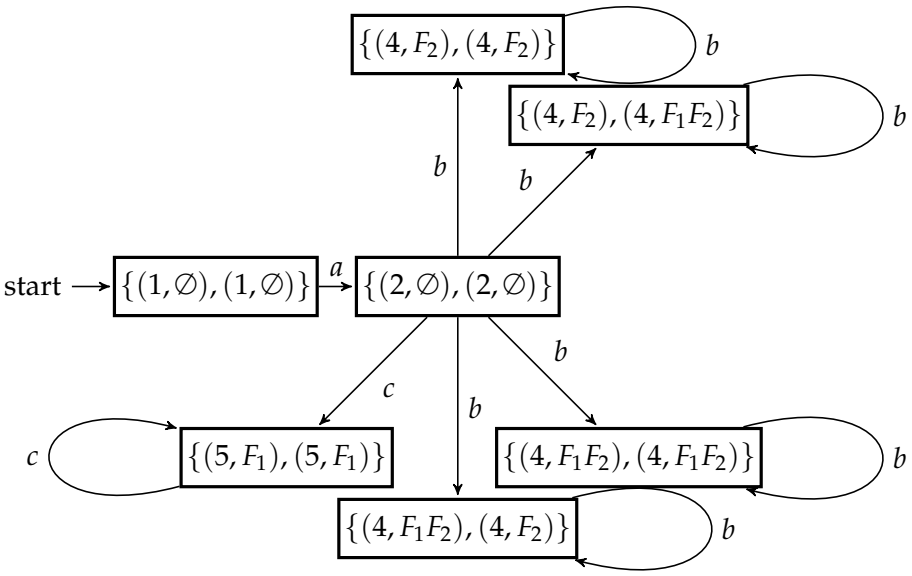


Figure 3.11 – The twin-plant of G in Figure 3.9

3.1.1.3 Verifier approach

In [YL02], the verifier approach was proposed. The diagnosability of a system is analyzed based on the construction of a nondeterministic automaton called a verifier. This approach was improved in [Mor+11; QK06]. In this section, we recall the verifier approach in [Mor+11], which has a better performance than the others.

In [Mor+11], this approach is proposed for codiagnosability analysis, but it can also be applied for monolithic diagnosability by considering that the system contains only one site which is itself.

The algorithm in [Mor+11] for local diagnosability of a module is given as follows:

For a given automata $G = (X, \Sigma, \delta, x_0)$. The set of fault events is $\Sigma_f \subset \Sigma$ and the set of normal events is defined as $\Sigma_N = \Sigma \backslash \Sigma_f$.

1. Step 1: Compute the automaton G_N that models the normal behavior of G:
   - Step 1.1: Build automaton A_N that contains a single state N with a self-loop labeled by all events in Σ_N;
   - Step 1.2: Build the normal automaton $G_N = G \times A_N = (X_N, \Sigma, \delta_N, \Gamma_N, x_{0,N})$;
   - Step 1.3: Define function $R : \Sigma \to \Sigma_R$ as:

$$R(e) := \begin{cases} e & \text{if } e \in \Sigma_o \\ e_R & \text{if } e \in \Sigma_u \backslash \Sigma_f \end{cases}$$

(The function R is used to rename the labels of events in $\Sigma_u \backslash \Sigma_f$. e is called the original event of e_R if $R(e) = e_R$.)

- Step 1.4: Redefine the event set of G_N, $G_N = G \times A_N = (X_N, \Sigma_{Rz}, \delta_N, \Gamma_N, x_{0,N})$. $\Sigma_{Rz} = \{e_{Rz} | e \in \Sigma,\ e_{Rz} = R(e)\}$ and for all $e \in \Sigma$ and $x_N \in X_N$, $\delta(x_N, R(e)) = \delta(x_N, e)$.

2. Step 2: Compute the automaton G_F that models the faulty behavior of G:
   - Step 2.1: Build automaton $A_l = (X_l, \Sigma_f, \delta_l, x_{0,l})$, where $X_l = \{N, F\}$, $x_{0,l} = N$, $\delta_l(N, f) = F$ and $\delta_l(F, f) = F$ for all $f \in \Sigma_{fz}$;
   - Step 2.2: Compute $G_l = G||A_l$ and mark all the states of G_l whose second coordinate is F;
   - Step 2.3: Compute the faulty automaton $G_F = CoAc(G_l)$.
3. Step 3: Compute the verifier automaton $G_V = G_N||G_F = (X_V, \Sigma_V, \delta_V, x_{0,V})$, where $\Sigma_V = \Sigma_{Rz} \cup \Sigma_F$. For a state $x_V \in X_V$, $x_V = (x_N, x_F)$, where $x_F = (x, x_l)$, $x \in X$ and $x_l \in X_l$.
4. Step 4: Verify if there exists a cycle

$$cl := (x_V^k, e_k, x_V^{k+1}, \cdots, x_V^h, e_h, x_V^k)(h \geq k > 0)$$

where for all $j \in \{1, \cdots, h\}$, $x_V^{k+j} = \delta_V(x_V^{k+j-1}, e_j)$ and $x_V^k = \delta_V(x_V^h, e_h)$, such that: (a) for each state x_V^r ($r \in \{k, k+1, \cdots, h\}$) in the cycle, $x_l^r = F$; (b) $\exists e_p \in \Sigma$, where $p \in \{k, k+1, \cdots, h\}$. If such a cycle exists, the module G is not locally diagnosable; otherwise, the module is locally diagnosable.

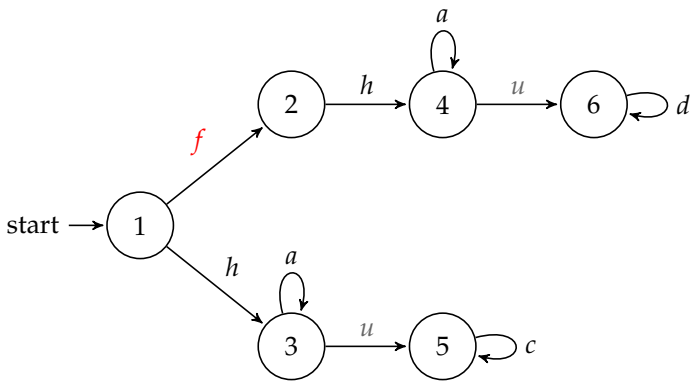


Figure 3.12 – An example of automaton G

Example 17 *Let us consider the automaton $G = (X, \Sigma, \delta, x_0)$ in Figure 3.12. $\Sigma_o = \{a, c, d, h\}$, $\Sigma_u = \{u, f\}$ and $\Sigma_f = \{f\}$. To analyze the diagnosability of G, we follow the above algorithm.*

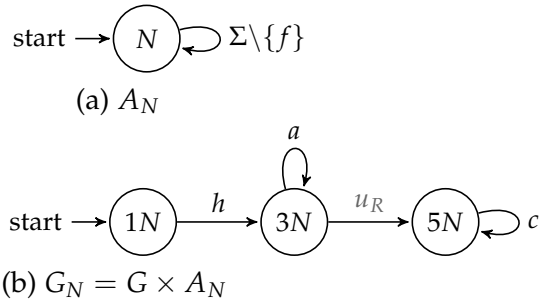


Figure 3.13 – The automata A_N and G_N of module G

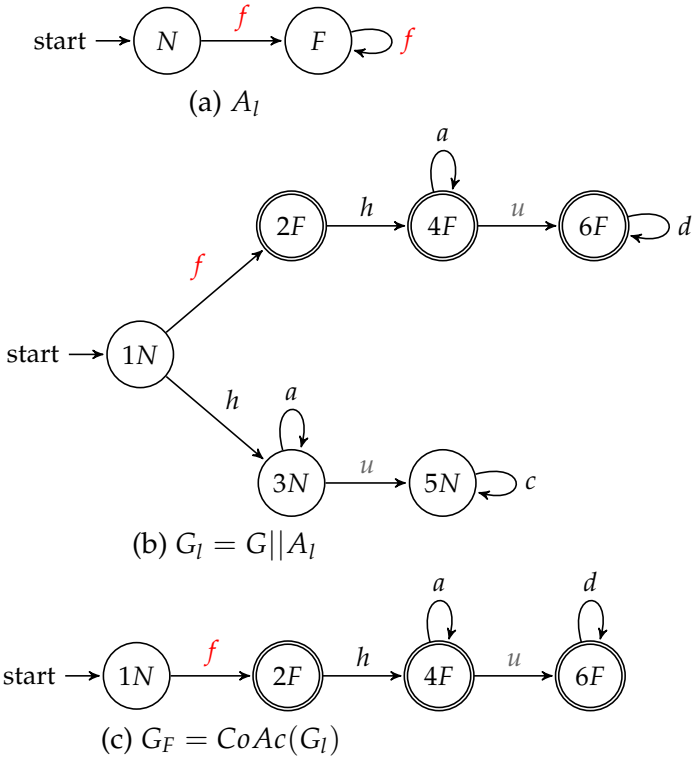


Figure 3.14 – The automata A_l, G_l and G_F of module G

1. *The automaton A_N is shown in Figure 3.13(a). Then, the normal behavior of G is modeled by $G_N = G \times A_N$ shown in Figure 3.13(b). The regular unobservable event u in G is renamed by R function as u_R;*

2. *The automaton A_l is shown in Figure 3.14(a). Then, the automaton $G_l = G||A_l$ shown in Figure 3.14(b). All the faulty states ($\{2F\}$, $\{4F\}$ and $\{6F\}$) are marked. The faulty behavior of G is modeled by $G_F = CoAc(G_l)$ shown in Figure 3.14(c).*

3. *The verifier is built by $G_V = G_N||G_F$ shown in Figure 3.15.*

4. *There exists a cycle at state $\{3N4F\}$ with a self-loop labeled by a. Therefore, G is not diagnosable.*

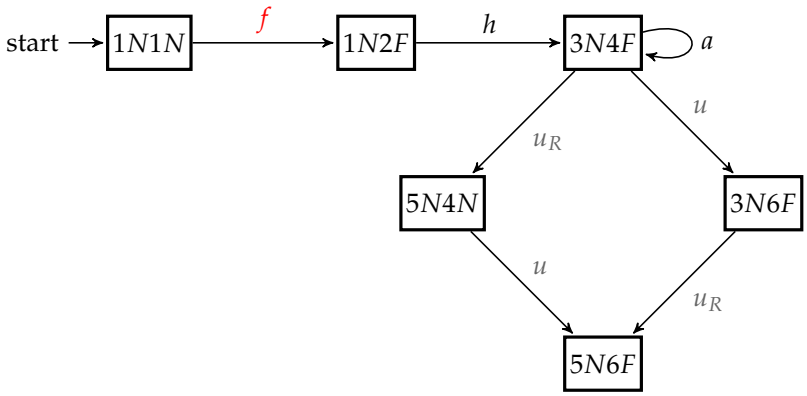


Figure 3.15 – The verifier $G_V = G_N||G_F$ of module G_1

For the verifier approach in [Mor+11], the complexity is $|X|^2 \times (|\Sigma_o| - |\Sigma_f^i|) \times |\Pi_f|$, where $|\Sigma_f^i|$ is the number of faulty events of the fault class i.

The complexity of the twin-plant/verifier approach is polynomial. However, the problem of these approaches is the combinatorial explosion. Generally, the twin-plant/verifier is a large scaled structure because they all build the parallel composition of two models and each one of the model has the same size with the given system in worst case.

3.1.1.4 Other automata-based approaches

In [Bou+15], a new variant of diagnoser was proposed. The variant diagnoser is constructed without constructing any intermediate model (i.e., generator). The efficiency of the verification of the necessary and sufficient condition is improved. For the Sampath's approach in [Sam+95], if there exists an F-uncertain cycle in the diagnoser, it needs to verify the existence of the corresponding normal cycle and faulty cycle in the generator. However, it is proved in [Bou+15], if there exists an F-uncertain cycle in the diagnoser, the normal cycle always exists. Therefore, it only needs to verify the existence of faulty cycle and the authors verify it by using the model of the system. In [Sch10], an abstraction was proposed to reduce the size of the initial automaton model. The author proposed to project the model in a subset of the initial set of events. It is demonstrated that if the projection is a Loop-preserving Observer, it guarantees that the abstracted model preserves the language-diagnosability property of the initial model. This study is proposed in the language theory framework. It does not propose a practical method to help designers to define the abstraction event set. In [PC02], the model checking techniques were applied to diagnosability analysis. A twin-plant is built by the parallel composition of the given automaton and a copy of itself. The system is not diagnosable if there exist two same observable sequences in the given automaton and its copy, such that one sequence contains a fault and the other one does not contain any fault. In [BG15], the diagnosability issue is reformulated as a model-checking problem. The diagnosability property is expressed by using CTL formula while considering extended definitions of diagnosability. The K-diagnosability

is also analyzed in the framework of model-checking. In [Gra09], the author provided a symbolic-based approach for diagnosability analysis. By using this technique, this approach avoids exploring the whole state space of the given model. This approach applied also the decentralized approach for diagnosability. In [Cab+15a], the authors provided an approach by building a PN diagnoser to detect the fault of an automaton model. The computational complexity of building this PN diagnoser is $\mathcal{O}(|\Pi_f| \times |X| \times |\Sigma|)$, where Π_f is the partition of fault classes, X is the set of states and Σ is the set of events. The complexity is linear on the number of fault classes, the number of states and the number of events of the system.

3.1.2 PN-based approaches

More recently, Petri nets (PNs) are used for diagnosability analysis and on-line fault diagnosis, which provide an expressive and compact representation of DES models. Researchers use PNs in order to tackle the combinatorial explosion.

The classic assumptions for diagnosability analysis using LPN are as following:

1. The LPN does not deadlock after firing any fault transition;
2. No cycle of unobservable transitions exists;
3. Faults are permanent, i.e., when a fault occurs the system remains infinitely faulty;
4. The same observable label may be associated with different transitions;
5. The structure of LPN and the initial marking M_0 are well known.

It is worth noticing that the assumptions are varied for different approaches. Precisely, for the approaches mentioned in the following sections:

- The approach in [Wen+05] assumes that the LPN model is *safe* and *live*;
- The approach in [Cab+14; Liu+14] assumes that the LPN model does not deadlock after firing any fault transition and LPN model is *bounded*;
- The approach in [Cab+12] assumes that the LPN model does not deadlock after firing any fault transition and LPN model can be *bounded* or *unbounded*;

The definition of diagnosability of a system modeled by an LPN is given as follows

Definition 8 *Given a live LPN* $= (N, M_0, \Sigma, \mathcal{L})$*, LPN is diagnosable w.r.t fault class* T_f^i *if there are not two sequences* σ_1 *and* σ_2*, which satisfy the following conditions:*

1. $\forall t_f \in T_f^i, t_f \notin \sigma_1$;

2. $\exists t_f \in T_f^i$ *such that* $t_f \in \sigma_2$ *and* σ_2 *can be arbitrarily long after the occurrence of* t_f*;*

3. $\mathcal{L}(\sigma_1) = \mathcal{L}(\sigma_2)$.

In other words, for a diagnosable LPN system, two sequences of transitions with the same observation must not be found, such that: one contains a fault transition and can be arbitrarily long after its occurrence; the other one does not contain a fault transition.

3.1.2.1 Diagosability analysis by checking T-invariants

In [WJ05; Wen+05], an approach was proposed for diagnosability analysis of LPN model. It is assumed that the LPN model is *safe* and *live*. This approach is proposed by checking the T-invariants of the LPN model. The T-invariant is an important property of LPN.

The definition of T-invariant is given in Section 2.3.2.

Definition 9 *A T-invariant* $\overrightarrow{\Omega}_{min}$ *is a minimal T-invariant, if there is no other T-invariant* $\overrightarrow{\Omega}$ *such that* $\overrightarrow{\Omega}(t) \leq \overrightarrow{\Omega}_{min}(t)$ *for all* $t \in T$.

The complexity for computing minimal T-invariants is polynomial [DA05]. $\mathcal{I}_F$ is defined as the set of minimal T-invariants that contain at least one fault transition and $\mathcal{I}_N$ the set of minimal T-invariants that do not contain any fault transition.

For a minimal T-invariant $\overrightarrow{\Omega}_{min}$ (valid as well as for a firing vector), $\Sigma_L(\overrightarrow{\Omega}_{min})$ is the multiset of observable labels of $\overrightarrow{\Omega}_{min}$. For a given $\overrightarrow{\Omega}_{min}$, $\Sigma_L(\overrightarrow{\Omega}_{min})$ can be obtained by the following steps:

1. Initialize $\Sigma_L(\overrightarrow{\Omega}_{min}) \leftarrow \emptyset$;

2. $\forall \exists t_i \in T_o$, $\overrightarrow{\Omega}_{min}(t_i) > 0$:

   a) $\Sigma_L(\overrightarrow{\Omega}_{min}) \leftarrow \Sigma_L(\overrightarrow{\Omega}_{min}) \cup \Sigma_{t_i}$, where Σ_{t_i} consists of $\overrightarrow{\Omega}_{min}(t_i)$ number of the label $\mathcal{L}(t_i)$.

For example, if $\exists t_1,\ t_2,\ s.t. \overrightarrow{\Omega}_{min}(t_1) = 1,\ \overrightarrow{\Omega}_{min}(t_2) = 2$ and $\mathcal{L}(t_1) = \mathcal{L}(t_2) = a$, there will be three '$a$' in $\Sigma_L(\overrightarrow{\Omega}_{min})$.

$\mathcal{S}(\overrightarrow{\Omega}_{min})$ is defined as the set of imply traces of the minimal T-invariant $\overrightarrow{\Omega}_{min}$, which is the set of all the possible firing sequences constructed by the labels in $\Sigma_L(\overrightarrow{\Omega}_{min})$.

Example 18 *Let us consider the LPN model in Figure 3.16.* $T_o = \{t_1, t_2, t_3, t_4, t_6, t_7, t_8\}$, $T_{reg} = \{\varepsilon_9\}$ *and* $T_f = \{f_5\}$. *The labels of transitions are shown in Figure 3.16. This LPN is* safe *and* live. *There exist four minimal T-invariants:* $\overrightarrow{\Omega}_{min,1} = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]^\tau$, $\overrightarrow{\Omega}_{min,2} = [0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1]^\tau$, $\overrightarrow{\Omega}_{min,3} = [0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 1]^\tau$, *and* $\overrightarrow{\Omega}_{min,4} = [0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1]^\tau$. $\Sigma_L(\overrightarrow{\Omega}_{min,1}) =$

$\{d\}$, $\Sigma_L(\overrightarrow{\Omega}_{min,2}) = \{b,c\}$, $\Sigma_L(\overrightarrow{\Omega}_{min,3}) = \{c\}$, *and* $\Sigma_L(\overrightarrow{\Omega}_{min,4}) = \{a,a,b\}$. $\mathcal{S}(\overrightarrow{\Omega}_{min,1}) = \{d\}$, $\mathcal{S}(\overrightarrow{\Omega}_{min,2}) = \{bc,\ cb\}$, $\mathcal{S}(\overrightarrow{\Omega}_{min,3}) = \{c\}$, *and* $\mathcal{S}(\overrightarrow{\Omega}_{min,4}) = \{aab,\ aba,\ bba\}$.

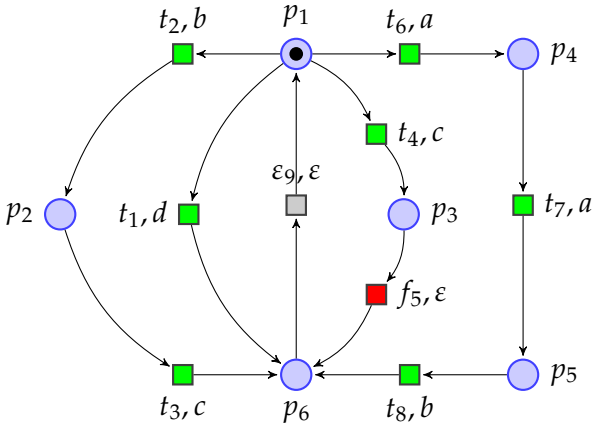


Figure 3.16 – An example of LPN model

A sufficient condition for diagnosability is proposed in [Wen+05]. A *safe* and *live* LPN is diagnosable, if there do not exist two minimal T-invariants, $\overrightarrow{\Omega}_{min,i}$, $\overrightarrow{\Omega}_{min,j}$ s.t. $\Sigma_L(\overrightarrow{\Omega}_{min,i}) = \Sigma_L(\overrightarrow{\Omega}_{min,j})$, i.e., it is not possible to find two arbitrarily long sequences of events ω_i, ω_j s.t. $\omega_i = \omega_j$, where $\omega_i \in \mathcal{S}(\overrightarrow{\Omega}_{min,i})$ and $\omega_j \in \mathcal{S}(\overrightarrow{\Omega}_{min,j})$. If two such minimal T-invariants do not exist, it implies that two sequences with same observation cannot be found, such that: one contains a fault and can be arbitrarily long after the fault; the other one does not contain a fault. Therefore, the system is diagnosable.

Example 19 *Let us consider again the LPN model in Figure 3.16. The system is diagnosable because there do not exist two minimal T-invariants,* $\overrightarrow{\Omega}_{min,i}$, $\overrightarrow{\Omega}_{min,j}$ *s.t.* $\Sigma_L(\overrightarrow{\Omega}_{min,i}) = \Sigma_L(\overrightarrow{\Omega}_{min,j})$.

The approach in [WJ05; Wen+05] has some drawbacks. The assumption that the LPN is *safe* and *live*, is strong. Hence, the scope of use of this approach is restricted. The author proposed only a sufficient condition for diagnosability. If two such minimal T-invariants are found, it needs to use other approaches to check the diagnosability of the LPN model. Moreover, there are some situations that are not taken into account by this sufficient condition. This sufficient condition for the diagnosability of a *safe* and *live* LPN will be supplemented in the following section.

3.1.2.2 Diagosability analysis using Minimal explanations

In [Cab+14; JB10], the notion of minimal explanation is used for diagnosability analysis. This notion reduces the impact of regular unobservable transitions in order to compact the state space. This section recalls the MBRG/BRD (Modified Basis Reachability Graph/ Basis Reachability Diagnoser) approach in [Cab+14].

Definition 10 *The set of explanations of an observable transition t at a marking M is defined by*

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

The corresponding e-vectors (explanation vectors) is the set:

$$Y(M,t) = \{\pi(\sigma) |\ \sigma \in \Sigma(M,t)\}$$

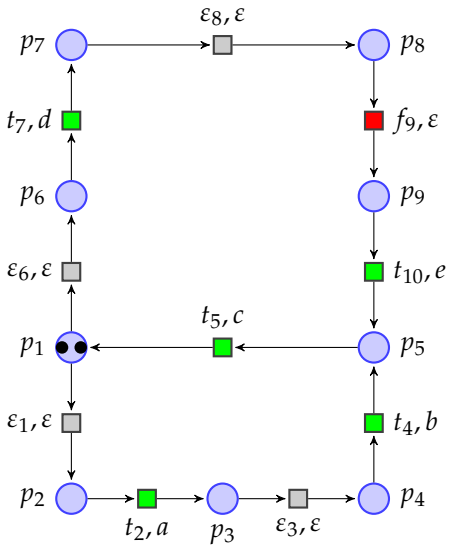


Figure 3.17 – An example of LPN

Example 20 *Let us consider the LPN in Figure 3.17. $T_o = \{t_2, t_4, t_5, t_7, t_{10}\}$, $T_{reg} = \{\varepsilon_1, \varepsilon_3, \varepsilon_6, \varepsilon_8\}$ and $T_f = \{f_9\}$. The labels of transitions are shown in Figure 3.17. The initial marking is $M_0 = [2\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^\tau$. Considering the transition t_2. After firing $\sigma_{u,1} = \varepsilon_1$ or respectively $\sigma_{u,2} = \varepsilon_6\varepsilon_1$, the transition t_2 is enabled. Therefore, $\sigma_{u,1}$ and $\sigma_{u,2}$ are explanations of t_2 at M_0. The explanation vectors of t_2 are $\overrightarrow{e_1} = [1\ 0\ 0\ 0\ 0]^\tau$ and $\overrightarrow{e_2} = [1\ 0\ 1\ 0\ 0]^\tau$. Here, for the e-vector, we keep only the elements of the unobservable transitions. For example, $\overrightarrow{e_1} = [1\ 0\ 0\ 0\ 0]^\tau$ means $\overrightarrow{e_1}(\varepsilon_1) = 1$, $\overrightarrow{e_1}(\varepsilon_3) = 0$, $\overrightarrow{e_1}(\varepsilon_6) = 0$, $\overrightarrow{e_1}(\varepsilon_8) = 0$ and $\overrightarrow{e_1}(f_9) = 0$, instead of $\overrightarrow{e_1} = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$.*

Definition 11 *The set of minimal explanations of an observable transition t at a marking M is defined by*

$$\Sigma_{\min}(M,t) = \{\sigma \in \Sigma(M,t) | \nexists \sigma' \in \Sigma(M,t) : \pi(\sigma') < \pi(\sigma)\}.$$

The corresponding set of minimal e-vectors of t at M is

$$Y_{\min}(M,t) = \{\pi(\sigma) | \sigma \in \Sigma_{min}(M,t)\}$$

Example 21 *As it was analyzed in Example 20, $\sigma_{u,1} = \varepsilon_1$ and $\sigma_{u,2} = \varepsilon_6\varepsilon_1$ are two explanations of t_2 at M_0. However, only $\sigma_{u,1}$ is a minimal explanation of t_1 but $\sigma_{u,2}$ is not one, because $\pi(\sigma_{u,2}) > \pi(\sigma_{u,1})$. The minimal e-vector is $\overrightarrow{e_1} = [1\ 0\ 0\ 0\ 0]^\tau$.*

Definition 12 *Let T_l be the set of transitions that are labeled by the observable event l. The set of minimal explanations of l at M is defined by*

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

The corresponding set of minimal e-vectors of l at M is

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

Example 22 *According to Example 21, since t_1 is the only transition labeled by a, the minimal explanation of a at M_0 is $\sigma_{u,1}$ and the minimal e-vector of a at M_0 is $\overrightarrow{e_1}$.*

Definition 13 *[Cab+14] Given an LPN $= (N, M_0, \Sigma, \mathcal{L})$ with labeling function $\mathcal{L}\colon T \to \Sigma$, where $N = (P, T, Pre, Post)$ and $T = T_o \cup T_u$. Let $\sigma \in L(N, M_0)$ be a firable sequence and $\omega = \mathcal{L}(P_{o,t}(\sigma))$ the corresponding observed word. The set of the* justifications *of ω is defined as follows.*

$$\hat{\mathcal{J}}(\omega) = \{\sigma_u \in T_u^* |\ [\exists \sigma \in \mathcal{FS}(\omega):\ \sigma_u = P_{u,t}(\sigma)]$$
$$\wedge\ [\nexists \sigma' \in \mathcal{FS}(\omega):\ \sigma'_u = P_{u,t}(\sigma') \wedge \pi(\sigma'_u) < \pi(\sigma_u)]\}$$

($\mathcal{FS}(\omega)$ is defined on page 22.)

Moreover, the set of the j-vectors *corresponding to $\hat{\mathcal{J}}(\omega)$ is defined as follows.*

$$\hat{Y}_{\min}(M_0,\omega) = \{\overrightarrow{y_u} \in \mathbb{N}^{|T_u|} |\ \exists \sigma_u \in \hat{\mathcal{J}}(\omega):\ \pi(\sigma_u) = \overrightarrow{y_u}\}$$

Definition 14 *Given an LPN and a firing sequence of transitions σ. Let $\omega \in \Sigma_o^*$ be a given observation, where $\omega = \mathcal{L}(\sigma_o)$ and $\sigma_o = P_{o,t}(\sigma)$. Let $\sigma_u \in \hat{\mathcal{J}}(\omega)$ be one of its minimal justification. Given a marking $M_b = M_0 + C_u \cdot \overrightarrow{y_u} + C_o \cdot \overrightarrow{y_o}$, where $\overrightarrow{y_u} = \pi(\sigma_u)$, $\overrightarrow{y_o} = \pi(\sigma_o)$, and C_u (resp. C_o) is the restriction of the incidence matrix C, which refers to T_u (resp. T_o). The marking reached by firing σ_o interleaved with σ_u is called* basis marking *with its j-vector $\overrightarrow{y_u}$.*

Example 23 *Let us consider again the LPN in Figure 3.17. Assuming that $\omega = ab$. The sequence of transition The set of justifications is $\hat{\mathcal{J}}(\omega) = \{\varepsilon_1\varepsilon_3\}$ and the set of j-vectors is $\hat{Y}_{\min}(M_0,\omega) = \{\overrightarrow{j_1}\}$, where $\overrightarrow{j_1} = [1\ 1\ 0\ 0\ 0]^\tau$. Moreover, this j-vector leads to a basis marking $M_4 = [1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]^\tau$.*

In [Cab+09b], the structure of basis reachability graph (BRG) was introduced as fault diagnosis approach by using minimal explanations. However, the structure BRG is not sufficient for diagnosability analysis, thus MBRG is then developed [Cab+09a]. In an MBRG, all faulty transitions are considered as observable transitions, i.e. minimal explanations are restricted to regular unobservable transitions. Therefore, in general, there are more markings in MBRG than in BRG. Each node of MBRG contains two elements (M, x), where M is defined as basis marking that are computed assuming that all faulty transitions are

observable. x is a row vector in $\{0,\ 1\}^r$, where r is the number of fault classes, and $x(i) = 1$ if the constraint set $\mathcal{T}(M)$ in (3.1) as follows is feasible w.r.t the T_f^i, $x(i) = 0$ otherwise.

$$\mathcal{T}(M) = \begin{cases} M + C_u \cdot \overrightarrow{z} \geq \overrightarrow{0}, \\ \sum_{t_f \in T_f^i} \overrightarrow{z}(t_f) > 0, \\ \overrightarrow{z} \in \mathbb{N}^{|T_u|} \end{cases} \tag{3.1}$$

According to MBRG, BRD is constructed to work in addition to MBRG for diagnosability analysis. A state in a node of BRD is a triple (M, x, h). M is a basis marking, x is the row vector as it was presented above and h is the row vector of fault tags. $h(i) = F$ means, at this marking, a fault in class T_f^i has occurred.

For each node in BRD, a diagnosis label Δ_i is associated to each observation ω and each fault class T_f^i.

- $\Delta(\omega,\ T_f^i) = 0$, if all the sequences with observation ω do not contain any fault transition in T_f^i;
- $\Delta(\omega,\ T_f^i) = 1$ if there exist $\sigma \in \mathcal{L}^{-1}(\omega)$ and $t_f \in T_f^i$ such that $t_f \in \sigma$, but $\forall(\sigma_o, \sigma_u) \in \hat{\mathcal{J}}\ (\omega)$ and $\forall t_f \in T_f^i$ it holds that $t_f \notin \sigma_u$;
- $\Delta(\omega,\ T_f^i) = 2$ if there exist two justifications (σ_o, σ_u), (σ_o', σ_u') such that $(\exists t_f \in T_f^i$ s.t $t_f \in \sigma_u) \wedge (\forall t_f \in T_f^i,\ t_f \notin \sigma_u')$;
- $\Delta(\omega,\ T_f^i) = 3$ if $\forall \sigma \in \mathcal{L}^{-1}(\omega),\ \exists t_f \in T_f^i$ s.t $t_f \in \sigma$.

From the initial node (M_0, x_0, h_0) (h_0 is set as $h_0 = N^r$, where r is the number of fault classes), the following nodes are built by using the MBRG. The main idea is the state estimation which is similar to the construction of a Sampath's diagnoser. All the indicators are calculated as they were presented above.

The label Δ_i is used to find indeterminate cycles and verify the necessary and sufficient condition for diagnosability. A cycle in BRD is called an uncertain cycle w.r.t. a fault class T_f^i if it includes staes with $\Delta = 1$, or $\Delta = 2$, or $\Delta = 1$ and $\Delta = 2$. If such a cycle exists, it needs to verify if this uncertain cycle is an indeterminate cycle.

Example 24 *Let us consider the LPN in Figure 3.17. The MBRG corresponding to the LPN is shown in Figure 3.18. The markings and e-vectors are listed in Table 3.1. Here, for e-vectors, only the elements of the regular unobservable transitions are kept. For example,* $\overrightarrow{e_1} = [1\ 0\ 0\ 0]^T$ *means* $\overrightarrow{e_1}(\varepsilon_1) = 1$, $\overrightarrow{e_1}(\varepsilon_3) = 0$, $\overrightarrow{e_1}(\varepsilon_6) = 0$ *and* $\overrightarrow{e_1}(\varepsilon_8) = 0$, *instead of* $\overrightarrow{e_1} = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T$. *The BRD corresponding to the MBRG is shown in Figure 3.19. It can be concluded that the system is not diagnosable because there exists an indeterminate cycle and the states in this cycle are marked by shadow zone.*

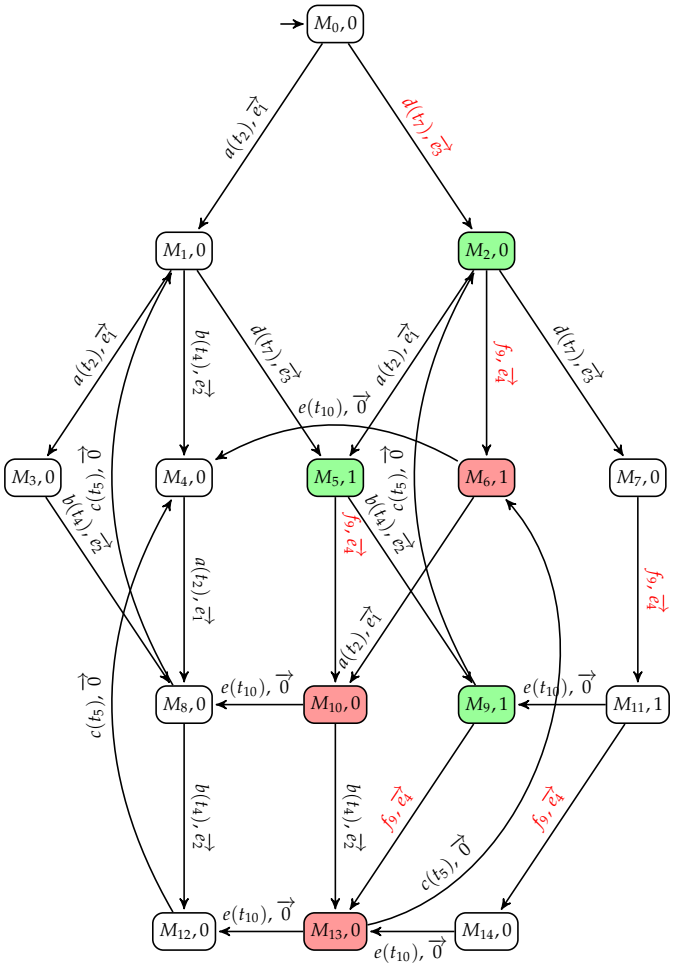


Figure 3.18 – MBRG of the LPN in Figure 3.17

In MBRG, we can notice that the arcs between markings are labeled either by observable transitions with their corresponding e-vectors (e.g. $\left(a\left(t_2\right), \overrightarrow{e_1}\right)$ from M_0) or by unobservable fault transitions with their corresponding e-vectors (e.g. $\left(f_9, \overrightarrow{e_4}\right)$ from M_2). Besides, only M_0, M_1, M_2, M_3, M_4, M_5, M_7, M_8, M_9 and M_{12} are basis markings, while M_6, M_{10}, M_{11}, M_{13}, and M_{14} are reached from basis markings by firing fault transitions. They are auxiliary markings for diagnosability analysis and they do not appear in BRD.

By using minimal explanations, MBRG gives a more compact representation of reachability space for diagnosability analysis, especially for an LPN model with many unobservable regular transitions. Note that, in the worst case, if there is no unobservable regular transition in the LPN model, the number of markings in the MBRG will be equal to the number of consistent markings. Therefore, the computational complexity in the worst case of this approach is equal to that of diagnoser approach. This approach can save much memory so that it reduces the computational cost of diagnosability analysis.

Table 3.1 – Markings and e-vectors in MBRG and BRD

j	M_j	j	M_j
0	$[2\,0\,0\,0\,0\,0\,0\,0\,0]^\tau$	11	$[0\,0\,0\,0\,0\,0\,1\,0\,1]^\tau$
1	$[1\,0\,1\,0\,0\,0\,0\,0\,0]^\tau$	12	$[0\,0\,0\,0\,2\,0\,0\,0\,0]^\tau$
2	$[1\,1\,0\,0\,0\,0\,1\,0\,0]^\tau$	13	$[0\,0\,0\,0\,1\,0\,0\,0\,1]^\tau$
3	$[0\,0\,2\,0\,0\,0\,0\,0\,0]^\tau$	14	$[0\,0\,0\,0\,0\,0\,0\,0\,2]^\tau$
4	$[1\,0\,0\,0\,1\,0\,0\,0\,0]^\tau$		
5	$[0\,0\,1\,0\,0\,0\,1\,0\,0]^\tau$	j	$\overrightarrow{e_j}$
6	$[0\,0\,0\,0\,0\,0\,2\,0\,0]^\tau$	1	$[1\,0\,0\,0]$
7	$[1\,0\,0\,0\,0\,0\,0\,0\,1]^\tau$	2	$[0\,1\,0\,0]$
8	$[0\,0\,1\,0\,1\,0\,0\,0\,0]^\tau$	3	$[0\,0\,1\,0]$
9	$[0\,0\,0\,0\,1\,0\,1\,0\,0]^\tau$	4	$[0\,0\,0\,1]$
10	$[0\,0\,1\,0\,0\,0\,0\,0\,1]^\tau$		

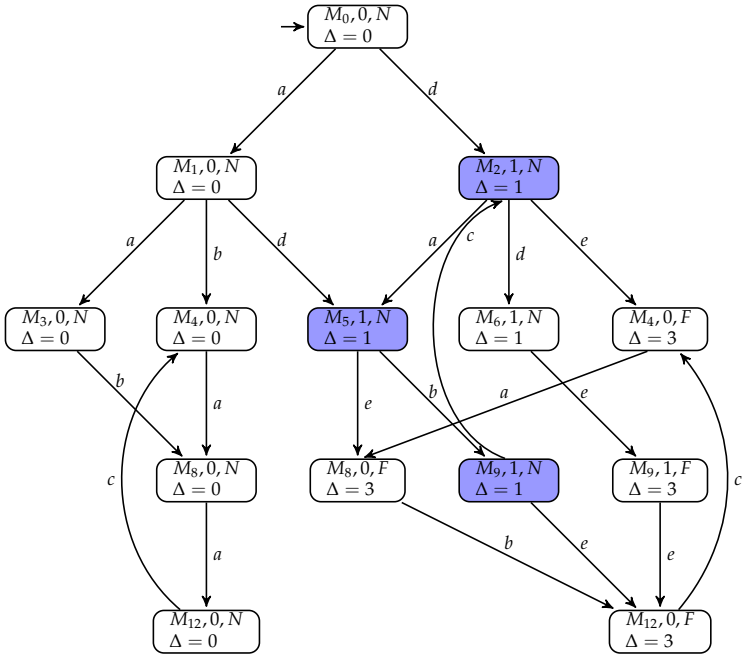


Figure 3.19 – BRD of the LPN in Figure 3.17

3.1.2.3 On-the-fly diagnosability analysis

The on-the-fly model-checking technique ([Bha+95; Fer+92; SE05]) is widely used for the verification of system properties in order to master the combinatorial explosion. The state space of a given system is built "on the fly" and the states are made and explored using depth-first search. The exploration of the state space is immediately stopped when a specified condition is satisfied. In this dissertation, the on-the-fly technique is used for diagnosability analysis.

For majority of the diagnosability analysis approaches based on the exploration of the state space ([Cab+12; Cab+14; Jia+01; Sam+95; Ush+98; YL02]), the entire state space

needs to be built *a priori*. After building the state space, the entire model for diagnosis (such as the diagnoser in [Sam+95]; the BRD in [Cab+14]) is built; in [Jia+01] ([YL02]), the twin-plant (respectively the verifier) is built; in [JB10], the ROF-automaton and F-verifier are built; in [Cab+12], the verifier net and its reachability graph are built.) Afterwards, the state space and the model for diagnosis are explored entirely to verify the specified condition for diagnosability.

In [Liu+14], the on-the-fly diagnosability analysis was proposed to deal with diagnosability, K-diagnosability, the minimal K to ensure diagnosability and online diagnosis using one formalism for a *bounded* LPN model. With the help of some stopping conditions, it has been shown that a part of state space could suffice for diagnosability analysis and on-line diagnosis, particularly when the system is non-diagnosable. The on-the-fly approach intends to avoid building the whole state space for checking diagnosability in order to reduce the computational cost. The state space and the model for diagnosis are built on-the-fly and in parallel with some stop conditions, which stop building certain of their branches. Along with their on-the-fly construction, the verification of diagnosability is executed. When the condition of undiagnosability is satisfied, the result is immediately given, that the system is not diagnosable.

Let us recall some basic notations and definitions from [Liu+14].

An *FM^i-graph* is considered as a directed non-deterministic graph relative to the fault class T_f^i. Without loss of generality, in this section, the diagnosis issue is discussed for a single class of faults. For the simplicity of representation, the superscript i may be omitted w.r.t T_f^i. Each node indicates a given fault marking (FM) and each arc indicates an observable event. An FM^i-graph can be treated as an ϵ-reduced observer automaton [CL07] with fault tag. For a bounded LPN, the number of states of the complete FM^i-graph w.r.t T_f^i is finite.

Definition 15 *An FM upon a sequence $\sigma \in T^*$ and a fault class T_f^i, is a vector $FM^i \in \mathbb{N}^{|P|+1}$:*

$$FM^i = \begin{bmatrix} mark(FM^i) \\ fault(FM^i) \end{bmatrix}$$

Where $mark(FM^i)$ represents a marking and $fault(FM^i)$ is a fault tag relative to the fault class T_f^i. $M_0\ [\ \sigma > mark(FM^i)$ and $fault(FM^i) = 1$ if $\exists t_f \in T_f^i, t_f \in \sigma$, otherwise, $fault(FM^i) = 0$.

Given two FMs FM and FM', it is denoted that $FM\ [\ \sigma > FM'$ iff $mark(FM)\ [\ \sigma > mark(FM')$; and $fault(FM') = fault(FM)$ if $\forall j, \sigma^j \notin T_f^i$, otherwise, $fault(FM') = 1$.

From the definition of FMs, an FM consists of a marking and a binary tag indicating the occurrence of fault. $\mathcal{Q}^i$ is the set of FMs w.r.t T_f^i. Then, for a given PN (N, M_0), the number of FMs is at most twice of the number of markings, i.e., $|\mathcal{Q}^i| \leq 2|R(N, M_0)|$ ($R(N, M_0)$ is defined on page 20). For the purpose of diagnosis, FM-graph is constructed

as the structure of state space and is developed to record the FMs that are reachable just after an observable event.

Definition 16 *The FM-graph relative to fault class T_f^i and called FM^i-graph is a 4-tuple $(\mathcal{N}, \Sigma_o, \delta, FM_0)$, where:*

- *$\mathcal{N} \subseteq \mathcal{Q}^i$ (because it is possible that only a part of FM^i-graph is built) is a set of FM^i nodes (FMs);*
- *Σ_o is a finite set of observable events;*
- *$\delta\colon \mathcal{Q}^i \times \Sigma_o \to 2^{\mathcal{Q}^i}$ is the transition function of FM: given $FM_1^i \in \mathcal{Q}^i$ and $e \in \Sigma_o$, $\delta(FM_1^i, e) = \{FM_2^i \mid \exists\, \sigma \in T^* \text{ s.t. } \mathcal{L}(\sigma) = e, FM_1\ [\ \sigma > FM_2\}$. The algorithm of δ function is illustrated in [Liu14];*
- *$FM_0 = [M_0^\tau, 0\,]^\tau$ is the initial node.*

Let the FM power set be FM-set, which is denoted as $\mathcal{X} = 2^{\mathcal{Q}}$ and the initial FM-set $x_0 = \{FM_0\}$.

Definition 17 *The FM-set transition mapping $\lambda\colon \mathcal{X} \times \Sigma_o \to \mathcal{X}$ is defined as follows: given an FM-set $x \in \mathcal{X}$ and an observable event $e \in \Sigma_o$, $\lambda(x, e) = \{FM' \mid \exists\, FM \in x, u \in T_u^*,\ t \in T_o,\ s.t.\ \mathcal{L}(ut) = e,\ FM\ [\ ut > FM'\}$.*

As a state of "Diagnoser" in [Sam+95], an FM-set can be associated with a tag which indicates the possibility of fault occurrence.

Definition 18 *The tagging function $tag\colon \mathcal{X} \to \{N, F, U\}$ is defined as follows:*

$$tag(x) = \begin{cases} N & \text{if } \forall\ FM \in x,\ fault(FM) = 0 \\ F & \text{if } \forall\ FM \in x,\ fault(FM) = 1 \\ U & \text{otherwise} \end{cases}$$

An FM-set x is also said to be normal (resp. F-certain, F-uncertain) if $tag(x) = N$ (resp. F, U). For FM-set x' reachable from x, if $tag(x) \in \{N, U\}$, it is possible that $tag(x') \in \{N, F, U\}$; whereas if $tag(x) = F$, then $tag(x') = F$, as faults are assumed to be permanent and, therefore, the F-certain tag is propagated to all the successive FM-sets.

An *FM-set tree* is a tree-like structure. The root node is the initial FM-set $x_0 = \{FM_0\}$. The subsequent nodes are the FM-sets reachable from the previous node by using the idea of state estimation. An FM-set is like a state of diagnoser automaton of [Sam+95]. Consequently, the condition of undiagnosability can be assimilated, as in the case of the

diagnoser approach, to the existence of indeterminate cycles in the FM-tree. A cycle is said to be indeterminate if all the states reached by the transition of the sequence in this cycle are F-uncertain and matches with two cycles in FM-graph: one cycle contains a fault, whereas the other one does not contain a fault. However, the diagnoser automaton must be built *a priori*, and all the diagnoser states are entirely enumerated. The FM-graph and FM-set are built on the fly and the conditions to stop the investigation of a branch of FM-set tree are as follows:

1. An F-certain FM-set is generated;
2. A new normal FM-set is equal to an existing one;
3. A new F-uncertain FM-set is equal to an existing one (then checking the existence of indeterminate cycle is necessary).

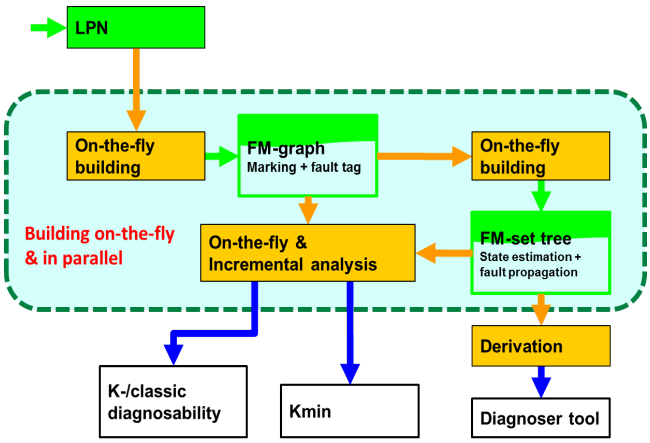


Figure 3.20 – Principle of on-the-fly diagnosability analysis

The principle of on-the-fly approach is shown in Figure 3.20. This approach is based on a depth-first search. The algorithm is developed in [Liu+14], which is proved to cover all the cases while building the FM-set tree on the fly. The main idea of the algorithm is as follow:

1. For a given FM-set $\mathcal{Z}$ (the initial FM-set is $\{FM_0\}$), $\mathcal{Z}_{con}$ is computed, which is the set of FMs that are obtained by firing all the possible unobservable transitions from the FMs in $\mathcal{Z}$;
2. For a given observable event e (e is given randomly because the prorities in the investigation of branches are not defined. Moreover, if e has already fired, another event will be selected), $\mathcal{Y}$ is computed, which is the next FM-set after firing e. The corresponding nodes in FM-graph and FM-set tree are built;
3. a) if $\mathcal{Y}$ is F-certain, return $\mathcal{Z}$ and go to (1);

b) if $\mathcal{Y}$ is normal and equal to an existing one, let $\mathcal{Z}$ be equal to $\mathcal{Y}$ and go to (2);

c) if $\mathcal{Y}$ is F-uncertain and equal to an existing one, check the existence of indeterminate cycle. If there exists an indeterminate cycle, return "the system is non-diagnosable". If there does not exist an indeterminate cycle, let $\mathcal{Z}$ be equal to $\mathcal{Y}$ and go to (1);

4. If the whole FM-graph and FM-set tree are built and there does not exist an indeterminate cycle, return "the system is diagnosable"

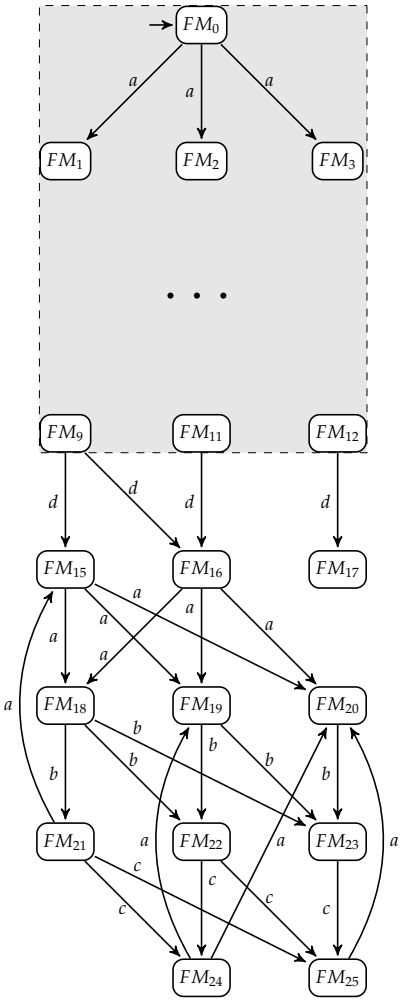


Figure 3.21 – FM-graph of the LPN in Figure 3.17

Example 25 *For the LPN in Figure 3.17 (page 40), the on-the-fly approach is used for checking diagnosability. For the priorities of investigating the branch of FM-set tree, the transition labeled a is chosen* a priori *before the transition labeled b, then c, d and e. The FM-graph (Figure 3.21) and FM-set tree (Figure 3.22) are constructed on the fly in parallel. In Figure 3.22, the tag of each FM-set is indicated beside. The FMs are shown in Table 3.2. It is worth noticing that only "observable" marking are represented in these models. Markings reached by unobservable transitions are not integrated but they are exploited to compute the next FM-set that will be reached after an observable*

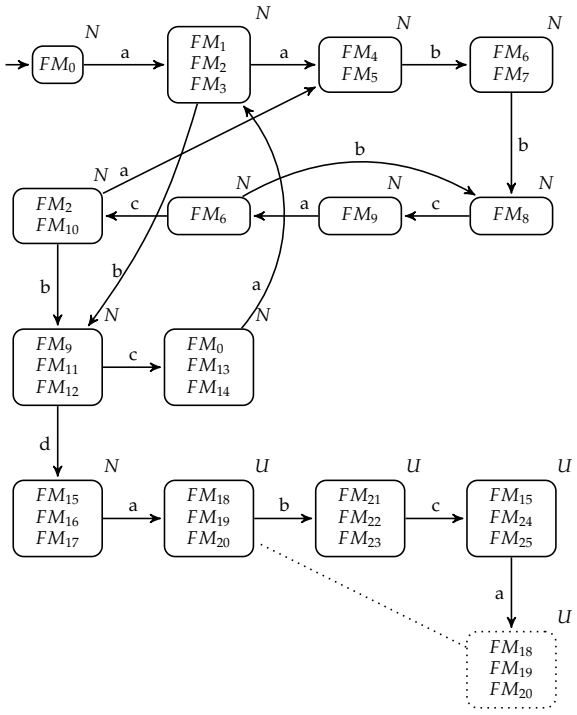


Figure 3.22 – FM-set tree of the LPN in Figure 3.17

Table 3.2 – Fault markings in Figure 3.21 and Figure 3.22

j	FM_j	j	FM_j
0	$[2\,0\,0\,0\,0\,0\,0\,0\,0 \mid 0]^\tau$	13	$[1\,1\,0\,0\,0\,0\,0\,0\,0 \mid 0]^\tau$
1	$[0\,0\,1\,0\,0\,1\,0\,0\,0 \mid 0]^\tau$	14	$[1\,0\,0\,0\,0\,1\,0\,0\,0 \mid 0]^\tau$
2	$[1\,0\,1\,0\,0\,0\,0\,0\,0 \mid 0]^\tau$	15	$[1\,0\,0\,0\,0\,0\,1\,0\,0 \mid 0]^\tau$
3	$[0\,1\,1\,0\,0\,0\,0\,0\,0 \mid 0]^\tau$	16	$[0\,1\,0\,0\,0\,0\,1\,0\,0 \mid 0]^\tau$
4	$[0\,0\,1\,1\,0\,0\,0\,0\,0 \mid 0]^\tau$	17	$[0\,0\,0\,0\,0\,1\,1\,0\,0 \mid 0]^\tau$
5	$[0\,0\,2\,0\,0\,0\,0\,0\,0 \mid 0]^\tau$	18	$[0\,0\,1\,0\,0\,0\,1\,0\,0 \mid 0]^\tau$
6	$[0\,0\,1\,0\,1\,0\,0\,0\,0 \mid 0]^\tau$	19	$[0\,0\,1\,0\,0\,0\,0\,1\,0 \mid 0]^\tau$
7	$[0\,0\,0\,1\,1\,0\,0\,0\,0 \mid 0]^\tau$	20	$[0\,0\,1\,0\,0\,0\,0\,0\,1 \mid 1]^\tau$
8	$[0\,0\,0\,0\,2\,0\,0\,0\,0 \mid 0]^\tau$	21	$[0\,0\,0\,0\,1\,0\,1\,0\,0 \mid 0]^\tau$
9	$[1\,0\,0\,0\,1\,0\,0\,0\,0 \mid 0]^\tau$	22	$[0\,0\,0\,0\,1\,0\,0\,1\,0 \mid 0]^\tau$
10	$[1\,0\,0\,1\,0\,0\,0\,0\,0 \mid 0]^\tau$	23	$[0\,0\,0\,0\,1\,0\,0\,0\,1 \mid 1]^\tau$
11	$[0\,1\,0\,0\,1\,0\,0\,0\,0 \mid 0]^\tau$	24	$[1\,0\,0\,0\,0\,0\,0\,1\,0 \mid 0]^\tau$
12	$[0\,0\,0\,0\,1\,1\,0\,0\,0 \mid 0]^\tau$	25	$[1\,0\,0\,0\,0\,0\,0\,0\,1 \mid 1]^\tau$

event occurrence. The construction of FM-set tree is stopped because a cycle is detected: a new F-uncertain FM-set is equal to an existing one (the FM-set which contains FM_{18}, FM_{19} and FM_{20}). With the help of FM-graph, it is identified that the cycle detected is indeterminate. Therefore, there is no need to continue the construction of FM-graph and FM-set tree, and it can be concluded that the system is not diagnosable. The numbering of FM in Figure 3.21 and Figure 3.22 corresponds to the order of construction of states by the depth-first analysis algorithm. The depth-first analysis is based on the construction of the FM-set tree. For example, at the initial $FM-set_0$ that contains

FM_0, the next firable event can be a or d. The event a fires because a should be chosen before d. Then, the nodes FM_1, FM_2 and FM_3 are added to FM-graph and in parallel, $FM - set_1$ is built which contains FM_1, FM_2 and FM_3. Then, it starts at $FM - set_1$ and the rest of FM-graph and FM-set tree are built like this way until the indeterminate cycle is found.

From Example 25, it can be observed that the main advantage of this approach is to avoid building the whole FM-graph and FM-set tree. There are some FMs that will be built in the whole state space but not in the on-the-fly construction of FM-graph such as $[0\,0\,0\,0\,0\,0\,2\,0\,0 \mid 0]^\tau$ and $[0\,0\,0\,0\,1\,0\,1\,0\,0 \mid 1]^\tau$. It is obvious that there are less FM-sets in the on-the-fly construction of FM-set tree than those in the whole construction of state space. The FM-graph and FM-set tree are built on the fly in order to make an efficient diagnosability analysis with the help of stopping conditions. In particular, for a non-diagnosable LPN, its diagnosability can be identified immediately after an indeterminate cycle is found, rather than continuing generating other branches. Moreover, if an F-certain node is found, it is unnecessary to continue the construction of this branch because all the subsequent nodes will also be faulty, and will be meaningless for diagnosability analysis. Hence, for the on-the-fly approach, there exist the worst cases that are indicated as follows:

1. The LPN is diagnosable;
2. The LPN is non-diagnosable but the indeterminate cycle is found at the end of the on-the-fly construction of state space.

In these cases, it is necessary to build the entire FM-graph and FM-set tree for diagnosability analysis.

As it is explained in [Bha+95; Fer+92; SE05], the on-the-fly verification technique is able to reduce the combinatorial explosion but it increases sightly the complexity. The reason is that on-the-fly verification technique builds on the fly the state space and analyzes the properties in parallel. Therefore, several times of analysis may be executed while building the state space. Assuming that the complexity of diagnoser approach in [Sam+95] is $\mathcal{C}_{diagnoser}$. In the worst case, the complexity of on-the-fly diagnosability analysis is $\alpha \cdot (n_{F-uncertaincycle} + 1) \cdot \mathcal{C}_{diagnoser}$, where $0 < \alpha < 1$ and $n_{F-uncertaincycle}$ is the number of F-uncertain cycles which are not F-indeterminate cycles.

However, it is worth noticing that the priorities in the investigation of branches is not defined. The efficiency of on-the-fly approach depends extremely on the LPN models. For the Example 25, most of the state space is constructed before finding the indeterminate cycle. The situation would not be different if the static priorities to the different observable events are assigned randomly. For example, deciding to process events in the order b, a, c, d, e would lead to a similar result. Therefore, some heuristics need to be developed in terms of priority between the branches to be investigated, in order to improve the

efficiency of the approach. These heuristics must be based on structural properties of the PN model, so the priorities do not depend on the hazard. Moreover, it can be observed that the number of FMs in FM-graph is bigger than the number of markings in MBRG because of the unobservable transitions, even though only a part of FM-graph is built. Therefore, some techniques can be used to improve the on-the-fly diagnosability analysis, such as minimal explanations and reduction rules.

3.1.2.4 Verifier Net (VN) approach

In [Cab+12], the Verifier Net (VN) approach was proposed for diagnosability analysis of both *bounded* and *unbounded* LPN. The K-diagnosablity is also discussed. The VN (an LPN) is built starting from the initial PN model. The diagnosability analysis is based on the construction of Reachability Graph (RG, for bounded PNs)/ Coverability Graph (CG, for unbounded PNs) of the VN. A sufficient and necessary condition was given for the diagnosability of system.

To analyze an LPN system with several fault classes, it needs to iterate the VN approach by building one VN and its RG/CG for each fault class. Therefore, without loss of generality, in this section, the diagnosability issue is discussed for a single class of faults. For the simplicity of representation, the superscript i will be omitted w.r.t T_f^i.

Definition 19 *Given a Petri net $N = (P, T, Pre, Post)$. $T' \subseteq T$ is a subset of the transitions in T. The T'-*induced subnet *of N is defined as the new Petri net $N' = (P, T', Pre', Post')$, where $Pre', Post'$ are the restrictions of $Pre, Post$. In this case, it is denoted that $N' \prec_{T'} N$. The net N' can be considered as being obtained by removing all transitions in $T \setminus T'$ and all related arcs.*

The LPN system associated with N' is called $T' - induced\ sub - LPN$ and denoted by $LPN' = (N', M_0', \Sigma', \mathcal{L}')$, where $M_0' = M_0$, $\Sigma' = \Sigma$ and the labeling function $\mathcal{L}'$ is defined as $\mathcal{L}$ but restricted to T'.

Example 26 *Given an LPN system $LPN = (N, M_0, \Sigma, \mathcal{L})$, where $N = (P, T, Pre, Post)$. Let $N' = (P', T', Pre', Post')$ be the T'-induced subnet, where $T' = T \setminus T_f = T_o \cup T_{reg}$. P and P' are used to distinguish among places of N and N', and they are disjoint even if they represent the same places. The $T' - induced\ sub - LPN$ of the given LPN is denoted by $LPN' = (N', M_0', \Sigma', \mathcal{L}')$*

Let us consider the LPN model in Figure 3.23. $T_o = \{t_3, t_4, t_5, t_6\}$, $T_{reg} = \{\varepsilon_1\}$ and $T_f = \{f_2\}$. The labels of transitions are shown in Figure 3.23. Its T'-induced sub-LPN is shown in Figure 3.24 by copying the given LPN model and removing the fault transition f_2, ε that does not in T'.

The VN is a LPN system constructed by composing LPN' with LPN with the synchronization on the observable transition labels. The VN is denoted as $\widetilde{LPN} = (\widetilde{N}, \widetilde{M}_o, \widetilde{\Sigma}_o, \widetilde{\mathcal{L}})$.

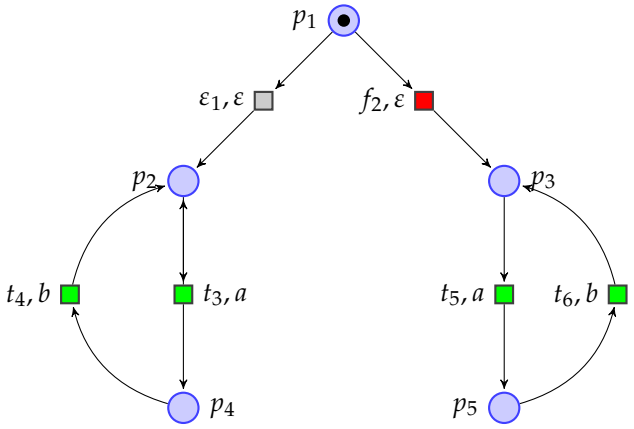


Figure 3.23 – An example of LPN

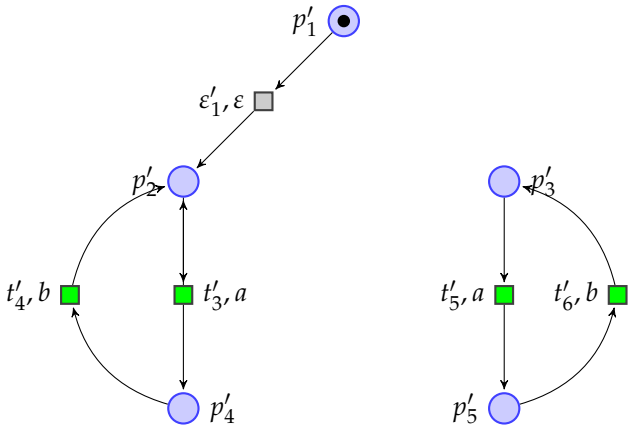


Figure 3.24 – T'-induced sub-LPN of the LPN in Figure 3.23

$\widetilde{N} = (\widetilde{P}, \widetilde{T}, \widetilde{Pre}, \widetilde{Post})$, where $\widetilde{P} = P' \cup P$, $\widetilde{T} = \widetilde{T}_o \cup (T'_{reg} \times \{\lambda\}) \cup (\{\lambda\} \times T_{reg}) \cup (\{\lambda\} \times T_f)$ and $\widetilde{T}_o = \{(t', t) \mid t' \in T'_o, t \in T_o, \mathcal{L}'(t') = \mathcal{L}(t)\}$. $\widetilde{M}_0 = \begin{bmatrix} M'_0 \\ M_0 \end{bmatrix}$, $\widetilde{\Sigma}_o = \{(\Sigma'_o \times \Sigma_o) \cup \{\varepsilon\}\}$ and $\widetilde{\mathcal{L}} : \widetilde{T} \to \widetilde{\Sigma}_o$. The incidence matrix of VN is $\widetilde{C} = \widetilde{Post} - \widetilde{Pre}$.

The algorithm of building the VN is given in [Cab+12]. Each transition of VN is composed either by two transitions (t'_o, t_o) (t'_o and t_o are observable and with the same label l) or by an unobservable transition t with the empty transition λ. The label of (t'_o, t_o) is (l, l). The label of the transition composed by t and λ is ε. The observable transitions of VN are indicated by (t'_o, t_o) where $t'_o \in T'_o$, $t_o \in T_o$ and $\mathcal{L}'(t'_o) = \mathcal{L}(t_o) = l$. The transition (t'_o, t_o) are connected to the places in P' following the column $Pre'(\cdot, t'_o)$ and $Post'(\cdot, t'_o)$; and to the places in P following the column $Pre(\cdot, t_o)$ and $Post(\cdot, t_o)$. The unobservable transitions of VN are indicated by (t', λ) or (λ, t) where $t' \in T'_{reg}$ and $t \in T_u$. For the transitions indicated by (t', λ) in VN, the transitions are connected to the places in P' following the column $Pre'(\cdot, t')$ and $Post'(\cdot, t')$; For the transitions indicated by (λ, t) in VN, the transitions are connected to the places in P following the column $Pre(\cdot, t)$ and

$Post(\cdot, t)$. For a transition $\widetilde{t} = (t', t)$ (one of t' and t can be λ), $\widetilde{Pre}(\cdot, \widetilde{t}) = \begin{bmatrix} Pre'(\cdot, t') \\ Pre(\cdot, t) \end{bmatrix}$ and $\widetilde{Post}(\cdot, \widetilde{t}) = \begin{bmatrix} Post'(\cdot, t') \\ Post(\cdot, t) \end{bmatrix}$.

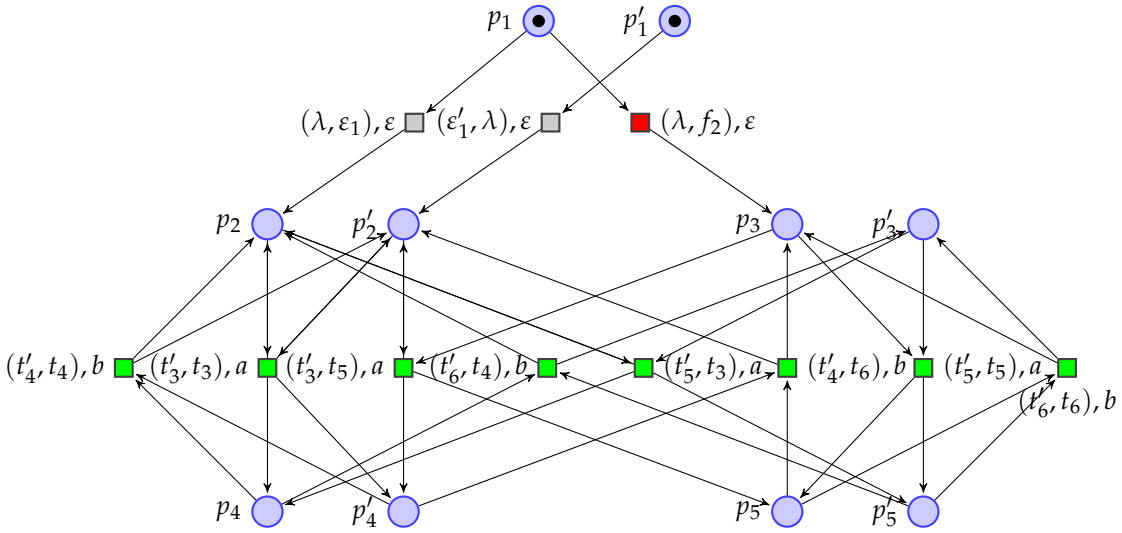


Figure 3.25 – VN of the LPN in Figure 3.23

Example 27 *Let us consider the LPN model in Figure 3.23 and its T'-induced sub-LPN is shown in Figure 3.24. The set of places built in VN is $\widetilde{P} = P' \cup P$. The set of transitions built in VN is*

$$\widetilde{T} = \{(t'_3, t_3), (t'_3, t_5), (t'_5, t_3), (t'_5, t_5), (t'_4, t_4), (t'_4, t_6), (t'_6, t_4), (t'_6, t_4)\} \cup \{(\varepsilon'_1, \lambda)\} \cup \{(\lambda, \varepsilon_1)\} \cup \{(\lambda, f_2)\}$$

$\widetilde{\mathcal{L}}((t'_3, t_3)) = \widetilde{\mathcal{L}}((t'_3, t_5)) = \widetilde{\mathcal{L}}((t'_5, t_3)) = \widetilde{\mathcal{L}}((t'_5, t_5)) = (a, a)$, $\widetilde{\mathcal{L}}((t'_4, t_4)) = \widetilde{\mathcal{L}}((t'_4, t_6)) = \widetilde{\mathcal{L}}((t'_6, t_4)) = \widetilde{\mathcal{L}}((t'_6, t_4)) = (b, b)$ and $\widetilde{\mathcal{L}}((\varepsilon'_1, \lambda) = \widetilde{\mathcal{L}}((\lambda, \varepsilon_1)) = \widetilde{\mathcal{L}}((\lambda, f_2)) = \varepsilon$. 11 transitions are built in the VN. It is worth noticing that there are some constructed transitions that are never enabled. The transitions (t'_5, t_3), (t'_5, t_5), (t'_6, t_4) and (t'_6, t_6) can never be enabled, because there is no token that can enter the places p'_3 and p'_5, since the fault transition is removed in T'-induced subnet.

To analyze the diagnosability of an LPN, the VN is constructed at first. Afterwards, its RG (for bounded LPN)/CG (for unbounded LPN) is built. A sufficient and necessary condition for diagnosability is proposed. Let $F(VN)$ denote the set of faulty nodes in the RG/CG of the VN. A node belongs to $F(VN)$, if it can be reached by firing a sequence that contains a fault transition.

Theorem 2 *[Cab+12] A LPN system $LPN = (N, M_0, \Sigma, \mathcal{L})$ is diagnosable iff there does not exist any cycle associated with a firable repetitive sequence in the VN that is reachable starting from any node in the set $F(VN)$.*

For a bounded LPN, each cycle in the RG of the VN corresponds to a firable repetitive sequence. For an unbounded PN, if a cycle in CG is reachable starting from a node in the set $F(VN)$, it is necessary to check if the cycle is associated with a repetitive sequence, i.e., if $\widetilde{C} \cdot \overrightarrow{y} > \overrightarrow{0}$, where $\overrightarrow{y}$ is the firing vector that contains all the transitions of the cycle.

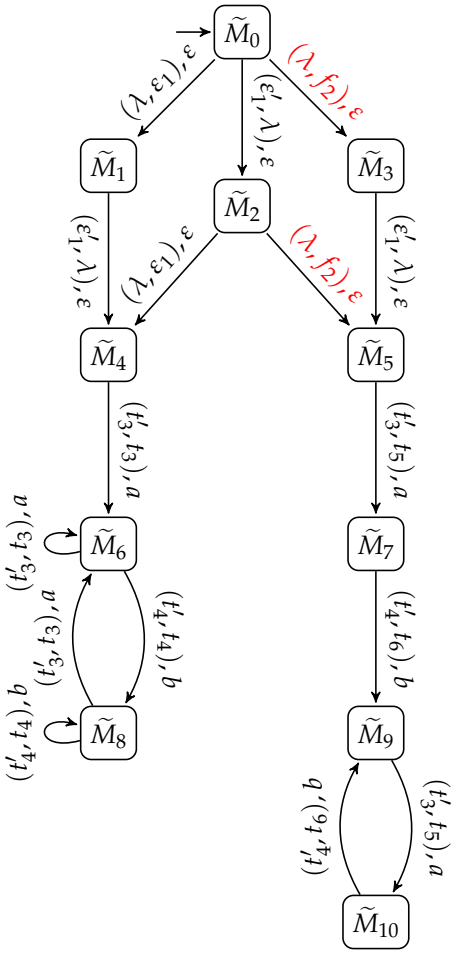


Figure 3.26 – CG of the VN in Figure 3.25

Example 28 *The LPN model in Figure 3.23 is* unbounded, *because after firing the transition ϵ_1, at the obtained marking, the transition t_3 is enabled and can be fired infinite times, which gives infinitse number of token to place p_4. The CG of the VN in Figure 3.25 is shown in Figure 3.25. In the CG, there exists a cycle*

$$\widetilde{M}_9 \xrightarrow{(t'_3,t_5),a} \widetilde{M}_{10} \xrightarrow{(t'_4,t_6),b} \widetilde{M}_9 \cdots$$

Moreover, this cycle corresponds to a firable repetitive sequence, because $\widetilde{C} \cdot \overrightarrow{y} > \overrightarrow{0}$, where $\overrightarrow{y}$ contains $(t'_3, t_5), a$ and $(t'_4, t_6), b$. Therefor, the LPN is not diagnosable, because there exists a cycle in CG associated with a firable repetitive sequence in the VN that is reachable after firing a fault.

The computational complexity for the construction of VN is linear in the number of the places and polynomial in the number of transitions of the initial LPN model. For the

Table 3.3 – Markings in Figure 3.26

j	$\widetilde{M}_j$
0	$[1\,0\,0\,0\,0 \mid 1\,0\,0\,0\,0]^\tau$
1	$[1\,0\,0\,0\,0 \mid 0\,1\,0\,0\,0]^\tau$
2	$[0\,1\,0\,0\,0 \mid 1\,0\,0\,0\,0]^\tau$
3	$[1\,0\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
4	$[0\,1\,0\,0\,0 \mid 0\,1\,0\,0\,0]^\tau$
5	$[0\,1\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
6	$[0\,1\,0\,\omega\,0 \mid 0\,1\,0\,\omega\,0]^\tau$
7	$[0\,1\,0\,1\,0 \mid 0\,0\,1\,0\,0]^\tau$
8	$[0\,\omega\,0\,\omega\,0 \mid 0\,\omega\,0\,\omega\,0]^\tau$
9	$[0\,\omega\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
10	$[0\,\omega\,0\,1\,0 \mid 0\,0\,0\,0\,1]^\tau$

diagnosability analysis of *bounded* PNs, the complexity of the VN approach is linear in the sum of the number of states and transitions of the reachability graph (RG) of the VN. For the diagnosability analysis of *unbounded* PNs, the complexity of the VN approach is an open issue. The limitation of the VN approach is the combinatorial explosion. The size of the VN is larger than the initial LPN model. The size of its RG/CG is increased. There exists combinatorial explosion problem when checking the diagnosability of a complex LPN model.

3.1.2.5 Other PN-based approaches

In [Ush+98], the diagnosability analysis has been extended to the framework of PNs. Based on the assumption that the places are partially observable, a simple $\omega-$diagnoser and an $\omega-$refined diagnoser were introduced for diagnosability analysis of unbounded PNs. In [LD07], the faults occurrence has been exactly estimated with the assumption that places are partially observable. Other approaches are developed with the assumption that the transitions are partitioned into observable and unobservable transitions and the fault transitions are unobservable. In [Lef16], the diagnosability analysis has been addressed for bounded or unbounded PN that are deadlock-free. The necessary and sufficient conditions for diagnosability have been given. This approach is based on the transformation of the coverability graph into an observation graph that encodes all observation sequences of measured markings and events w.r.t. the sensor configuration. The diagnosability is determined by analyzing the paths and circuits in the observation graph. In [JB10], the authors proposed an approach by using minimal explanations: a structure called ROF-automaton is developed, which is smaller than the reachability graph of the PN. Then, the technique of verifier is applied for diagnosability analysis. In [Dot+09], some integer linear programming (ILP) problems are defined for on-line diagnosis: after an observable sequence, the possible occurred fault transitions or regular unobservable transitions of system are provided by an efficient algorithm. In [Bas+12], an approach for checking K-diagnosability

has been developed by using the ILP technique. A necessary and sufficient condition for K-diagnosabilityof bounded PNs is proposed. In [Lef14b], the on-line fault diagnosis has been addressed for DES modeled with partially observable Petri nets. The on-line diagnosis is investigated according to the capture and analysis of observation sequences that include some observable events and the partial measurement of the successive states reached by the system. The on-line diagnosis is defined as ILP problem and a forward-backward algorithm has been proposed, which can provide a diagnosis decision with a reasonable computation effort. The result has been extended in [Lef14a]. In [Mad+10], the unfolding technique was applied for diagnosability analysis: the unfolding of a given LPN is infinite, but if the LPN is bounded, the unfolding will eventually repeat itself. A verifier is built for checking diagnosability, which compares pairs of paths from the initial model with the same observation. In [Gou+14], the discriminability of a system was defined, which is the possibility to detect the exclusive occurrence of a supervision pattern of a particular behavior of interest: the twin-plant approach is adapted to LPN unfolding in order to solve the combinatorial explosion problem.

3.2 Contributions on monolithic diagnosability analysis

3.2.1 Diagnosis and diagnosability analysis using reduction rules

This section proposes 7 reduction rules of LPN in order to simplify *a priori* the initial model before analyzing the diagnosability. Some transitions and places are suppressed. It will be proved that the diagnosability property of the reduced model keeps consistent with that of the initial model.

3.2.1.1 Reduction rules for regular unobservable transitions

The aim of this section is to propose reduction rules to suppress some regular unobservable transitions.

In [Ber86; Ber87; Mur89], some transformation techniques of PN models have been proposed to simplify a PN model to facilitate the analysis of a complex PN system. The PN model is often reduced to a simpler one by using these transformation techniques, while the considered system properties are preserved.

For the event-based diagnosability analysis of a LPN system, the labels of the transitions in T_o generate the language of the LPN model, which is the basis of the diagnosability analysis. The transitions in T_f contain the fault information. However, the transitions in T_{reg} do not contain the necessary information of diagnosability analysis. Besides, the existence of regular unobservable transitions increases the size of the state space of the approaches proposed in [Cab+12; Li+15c; Liu+14; Sam+95]. In this section, some reduction rules are proposed to remove some regular unobservable transitions and some places

before generating the state space of the LPN model [Li+16a]. A theorem will be given to establish that these rules preserve the diagnosability property of the system.

It is worth noticing that the technique of reduction rules is a complement of most approaches for diagnosability analysis. In this section, the diagnoser approach is used to compare the diagnosability analysis of the initial LPN model and that of the reduced LPN model.

In [Ber86; Ber87; Mur89], some reduction rules are proposed, but not all the rules preserve the diagnosability property. In Figure 3.27, five rules are proposed and it has been proved that these rules guarantee the preservation of the *liveness* and *boundedness* of the PN model [Mur89]. The algorithms of these reduction rules though incidence matrix operations have been proposed in [Mm+13] and the computational complexity of these algorithms is polynomial. These rules are used to suppress the regular unobservable transitions. Therefore, some of the reduction rules are modified by indicating the type of transitions. The modified rules do not change the preservation of the *liveness* and *boundedness* of the system:

1. Fusion of Series of Places (FSP) as depicted in Figure 3.27(1). The transition ε_k is regular and unobservable, which has one input arc and one output arc. The input arc of this transition is the only one output arc of its pre-place (p_i). The regular unobservable transition is suppressed and the two places p_i and p_j are merged;

2. Fusion of Series of Transitions (FST) as depicted in Figure 3.27(2)a and Figure 3.27(2)b. ε_h in Figure 3.27(2)a (respectively ε_k in Figure 3.27(2)b) is regular and unobservable. Moreover, the place p_i does not contain any token, because the language of the LPN model will be changed, if the token in p_i is moved to the pre-places of t_k (ε_k) or the post-places of ε_h (t_h). The place p_i has one input arc and one output arc. The output arc of p_i is the only input arc of its post-transition. (Note that in Figure 3.27(2)b, p_i is the only post-place of ε_k, which means the firing of ε_k can only enable the transition t_h so that it does not change the language of the LPN model.) The place p_i is suppressed. The two transitions are merged by suppressing the regular unobservable transition;

3. Fusion of Parallel Places (FPP) as depicted in Figure 3.27(3). Each of the two places has one input arc and one output arc. The pre-transition (respectively the post-transition) of these two places is the same. The two places are merged;

4. Fusion of Parallel Transitions (FPT) as depicted in Figure 3.27(4). ε_k and ε_h are all regular and unobservable. Each of the two transitions has one input arc and one output arc. The pre-place (respectively the post-place) of these two transitions is the same. The two regular unobservable transitions are merged;

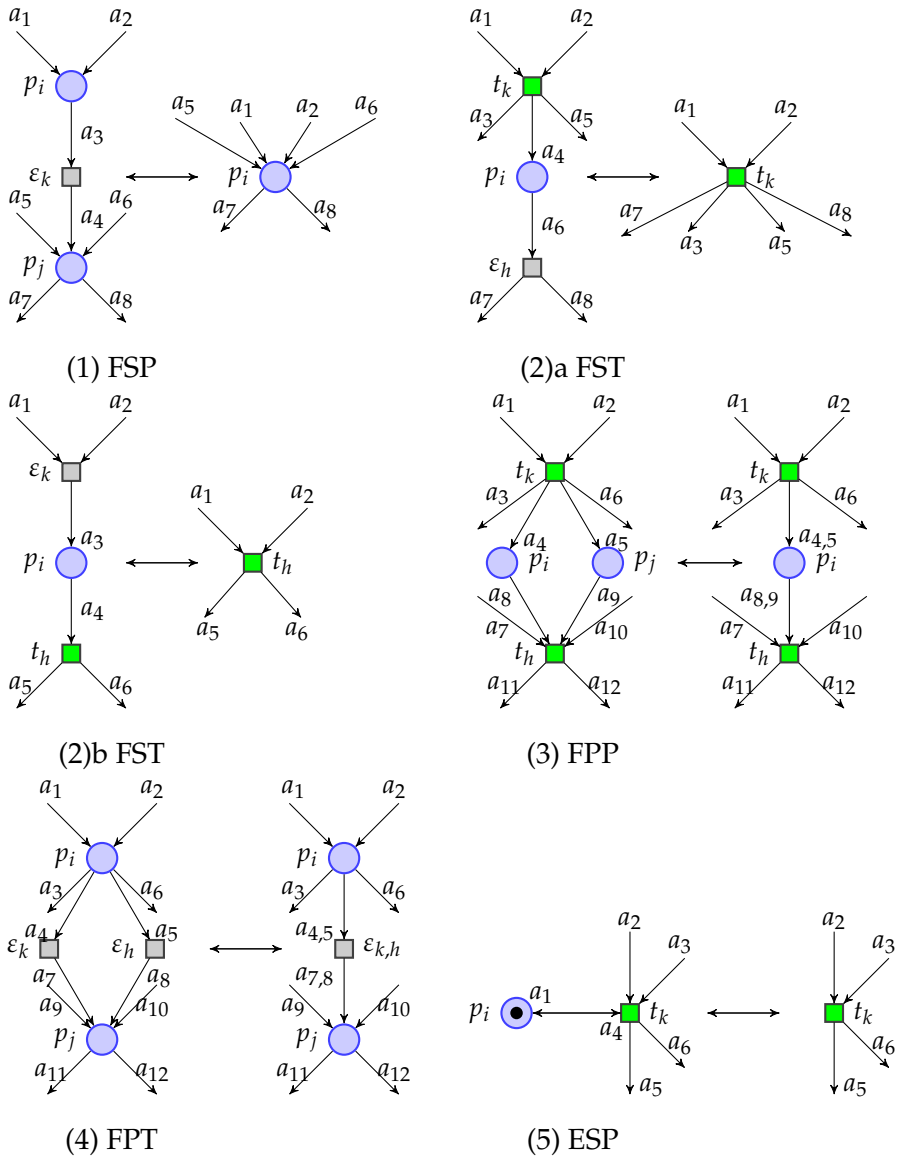


Figure 3.27 – Reduction rules for regular unobservable transitions

5. Elimination of Self-loop Places (ESP) as depicted in Figure 3.27(5), the place p_i contains at least one token and it has one input arc and one output arc. The pre-transition of this place is the same with its post-transition. The place is suppressed;

While using these reduction rules, the initial marking of the reduced LPN model is defined as follows.

Definition 20 *The initial marking of the reduced LPN model $\widehat{M_0}$ is calculated from the initial marking of the initial LPN model M_0 by the following steps:*

– *Initialization of $\widehat{M_0}$: $\widehat{M_0} \leftarrow M_0$. If no rule can be applied, $\widehat{M_0} = M_0$;*

- *If the rule (1) FSP is used, remove the j^{th} line of $\widehat{M_0}$ and $\widehat{M_0}(p_i) = M_0(p_i) + M_0(p_j)$, because the tokens in the two places are all available to be consumed to fire the following transitions;*
- *If the rule (2) FST is used, remove the i^{th} line of $\widehat{M_0}$. The place does not contain any token as presented before;*
- *If the rule (3) FPP is used, remove the j^{th} line of $\widehat{M_0}$ and $\widehat{M_0}(p_i) = Min(M_0(p_i), M_0(p_j))$, where the function $Min(M_0(p_i), M_0(p_j))$ is the minimum of $M_0(p_i)$ and $M_0(p_j)$. For example, without considering the existence of the arc a_7 and a_{10}, if $M_0(p_i) = 2$ and $M_0(p_j) = 3$, the post-transition (t_h) can be fired only twice. After using the rule (3), $\widehat{M_0}(p_i) = Min(M_0(p_i), M_0(p_j)) = 2$ and its post-transition can also be fired twice;*
- *If the rule (5) ESP is used, remove the i^{th} line of $\widehat{M_0}$.*

For all the reduction rules shown in Figure 3.27, the left figure and the right figure of each rule can be treated as two sub-structures of LPN system. The set of input arcs (respectively output arcs) of the initial sub-structure and the reduced sub-structure is the same. It means that using these rules, the removing of the regular unobservable transitions, the places and the arcs has no influence on the firing of the previous transitions and following transitions. Therefore, by using these reduction rules, the language of the PN is not modified [Ber87]. These reduction rules have no influence on the firing of the previous transitions and following transitions of the sub-structures of the PN.

Example 29 *For the reduction rule shown in Figure 3.27(1), the set of input arcs of the two sub-structures is the same: $\{a_1, a_2, a_5, a_6\}$. The set of output arcs of the two sub-structures is also the same: $\{a_7, a_8\}$. The results are the same for the other reduction rules.* □

In the framework of LPN, the Proposition 1 is given.

Proposition 1 *[Li+16a] By using reduction rules (1)-(5), the removing of some regular unobservable transitions, some related places and arcs does not modify the language of the LPN model.*

Proof: According to the proposition in [Ber87], by using reduction rules, the language of the PN model is not modified and the firing of the previous transitions and following transitions is not affected. For the reduction rule (1), given a firing sequence of transitions $\sigma_i\varepsilon_j\sigma_k$, where the unobservable transition ε_j can be suppressed by using reduction rules. The observation of this sequence is $\mathcal{L}(\sigma_i\varepsilon_j\sigma_k) = \omega$. After eliminating the transition ε_j, the sequence becomes $\sigma_i\sigma_k$ and it is still firable. The observation is $\mathcal{L}(\sigma_i\sigma_k) = \mathcal{L}(\sigma_i\varepsilon_j\sigma_k) = \omega$. Therefore, the observation of a firing sequence of transitions does not change after using the reduction rules. The other rules can be proved in a similar way. □

Theorem 3 is given to guarantee the correctness of diagnosability analysis using reduction rules.

Theorem 3 *[Li+16a]By using reduction rules (1)-(5), the diagnosability of the reduced LPN model keeps consistent with the diagnosability of the initial LPN model.*

Proof: By proposition 1, it is proved that by using reduction rules, the language of the LPN model is not modified. According to the definition of diagnosability, if the system is non-diagnosable, there exist two sequences σ_1 and σ_2 with the same observation, such that σ_1 does not contain any fault, but σ_2 contains a fault and can be arbitrarily long after the occurrence of the fault. Assuming the σ_1 (σ_2) contains a regular unobservable transition ε_i that can be removed by using these reduction rules. After removing ε_i, the observation of the reduced sequence $\widehat{\sigma_1}$ ($\widehat{\sigma_2}$) does not change. Therefore, the system is still non-diagnosable, because $\widehat{\sigma_1}$ ($\widehat{\sigma_2}$) and σ_2 (σ_1) satisfy the conditions in definition of diagnosability.

If the system is diagnosable, assuming that the regular unobservable transition ε_j can be removed by using these reduction rules. For any sequence of transitions σ_i that contains ε_j, the observation of the sequence $\widehat{\sigma_i}$ stays the same after removing ε_j. The system is still diagnosable, because it is not possible to find $\widehat{\sigma_i}$ and $\widehat{\sigma_j}$ that satisfy the conditions of definition of diagnosability. □

Theorem 3 proves that, by using the reduction rules (1)-(5), the diagnosability property is preserved. Therefore, before building the state space of the initial LPN system, the reduction rules can be applied to simplify the LPN system in order to facilitate the diagnosability analysis.

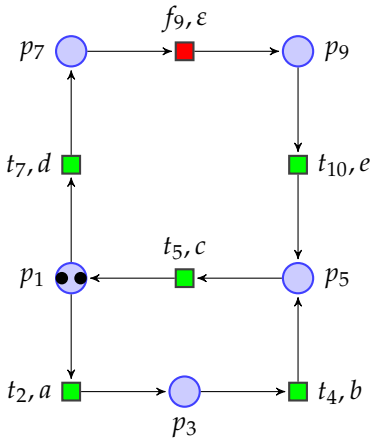


Figure 3.28 – Reduced LPN model of the LPN in Figure 3.17 (page 40)

Example 30 *Considering the LPN model in Figure 3.17 (page 40). By using the reduction rules FST, the regular unobservable transitions ε_1, ε_3, ε_6 and ε_8 are suppressed. Meanwhile, the places p_2,*

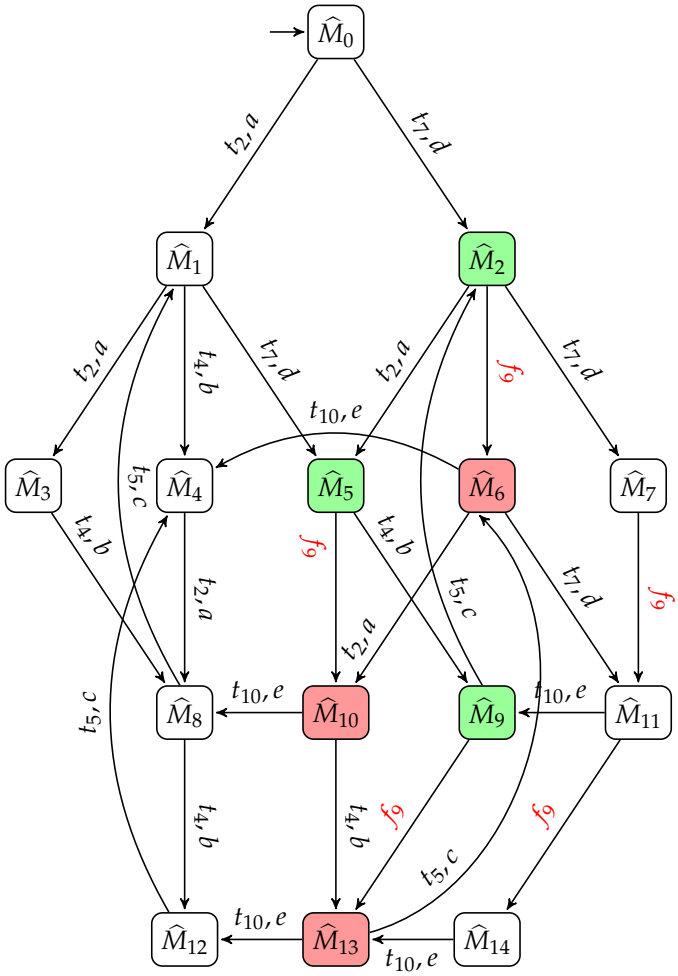


Figure 3.29 – Reachability graph of the reduced LPN shown in Figure 3.28

Table 3.4 – Markings in Figure 3.29 and Figure 3.30

j	$\widehat{M}_j$	j	$\widehat{M}_j$
0	$[2\,0\,0\,0\,0]^\tau$	8	$[0\,1\,1\,0\,0]^\tau$
1	$[1\,1\,0\,0\,0]^\tau$	9	$[0\,0\,1\,1\,0]^\tau$
2	$[1\,0\,0\,1\,0]^\tau$	10	$[0\,1\,0\,0\,1]^\tau$
3	$[0\,2\,0\,0\,0]^\tau$	11	$[0\,0\,0\,1\,1]^\tau$
4	$[1\,0\,1\,0\,0]^\tau$	12	$[0\,0\,2\,0\,0]^\tau$
5	$[0\,1\,0\,1\,0]^\tau$	13	$[0\,0\,1\,0\,1]^\tau$
6	$[1\,0\,0\,0\,1]^\tau$	14	$[0\,0\,0\,0\,2]^\tau$
7	$[0\,0\,0\,2\,0]^\tau$		

p_4, p_6 and p_8 are also suppressed. The reduced LPN model is shown in Figure 3.28. The diagnoser approach is used for the diagnosability analysis of the initial LPN model and the reduced LPN model. For the initial LPN model in Figure 3.17, the reachability graph contains 44 nodes and the diagnoser contains 33 nodes. The system is non-diagnosable, because there exist indeterminate cycles. For the reduced LPN model, the reachability graph (shown in Figure 3.29) contains only

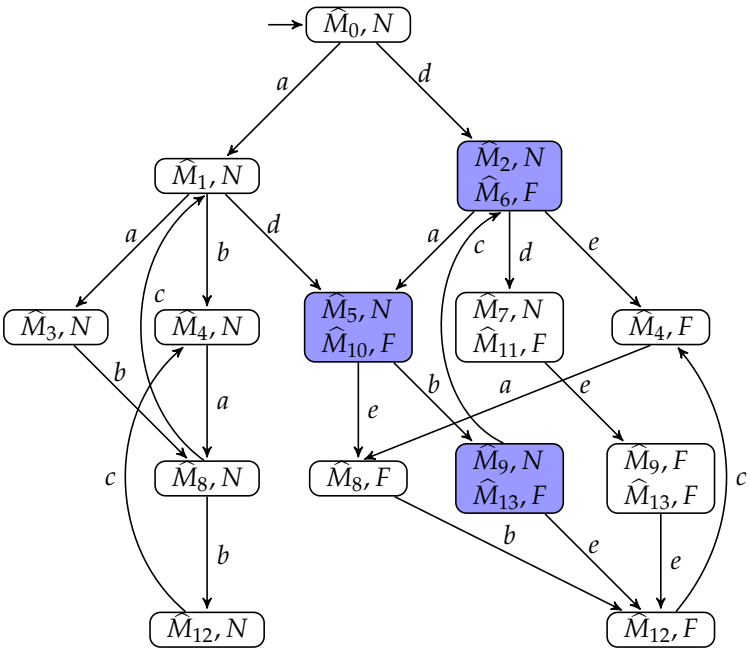


Figure 3.30 – Diagnoser of the reduced LPN shown in Figure 3.28

15 nodes and the diagnoser (shown in Figure 3.30) contains 14 nodes. The markings are shown in Table 3.4. The system is also non-diagnosable and the indeterminate cycle is shown in the shadow zone.

It is worth noticing that not all the regular unobservable transition can be suppressed. A counterexample is proposed in Figure 3.31. The regular unobservable transition ε_1 in Figure 3.31 cannot be suppressed. Therefore, while analyzing the diagnosability of a given LPN, our reduction rules are necessarily applied on the regular unobservable transitions that are possibly reduced.

The reduction rules provide the possibility to facilitate the diagnosability analysis of a complex system. The state space is much smaller than that of the initial LPN model, if many regular unobservable transitions are reduced by using these rules. The reduction rules (1), (2) and (4) suppress some regular unobservable transitions, so the number of states of the state space is reduced. In addition, The reduction rules (1), (2), (3) and (5) suppress some places so that it takes less memory cost to store each marking. Therefore, by using the reduction rules, the memory cost is reduced and the efficiency for diagnosability analysis is improved.

3.2.1.2 Reduction rules for observable transitions

In this section, we intend to suppress a specific kind of observable transitions in order to reduce further the combinatorial explosion. A theorem will be given to prove that the diagnosability of the reduced LPN keeps consistent with the diagnosability of the initial

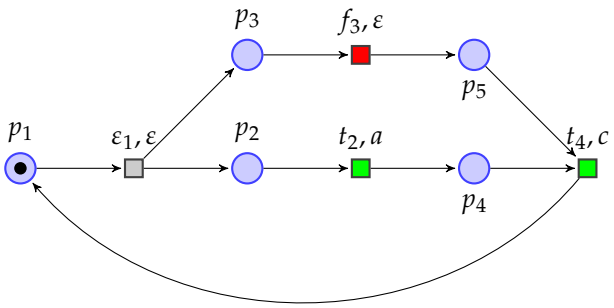


Figure 3.31 – A counter-example for reduction rules of unobservable transitions

LPN model. Afterwards, it will be demonstrated that, by using these rules, the diagnoser of the reduced LPN model is still valid for on-line diagnosis.

The specific kind of observable transitions that can be suppressed is defined as follows.

Definition 21 *A transitions t is called an* exclusively labeled observable transition *(ELOT) if $t \in T_o$ and $\nexists t' \in T_o$ s.t. $\mathcal{L}(t') = \mathcal{L}(t)$.*

In other words, the label of an ELOT is unique. For some diagnosability analysis techniques based on the state estimation (such as the approaches in [Cab+14; Li+15b; Li+15c; Liu+14; Sam+95]), the firing of an ELOT does not cause any ambiguity for the state estimation. For example, a reachability graph of an LPN model (that contains unobservable transitions or observable transitions with the same label) is by default a non-deterministic graph. However, if the transitions of an LPN are all ELOTs, its reachability graph will be a deterministic graph.

The following reduction rules are proposed in Figure 3.32 to suppress this specific kind of observable transitions (ELOTs). As it was presented in Section 3.2.1.1, the modified rules do not change the preservation of the *liveness* and *boundedness* of the LPN model:

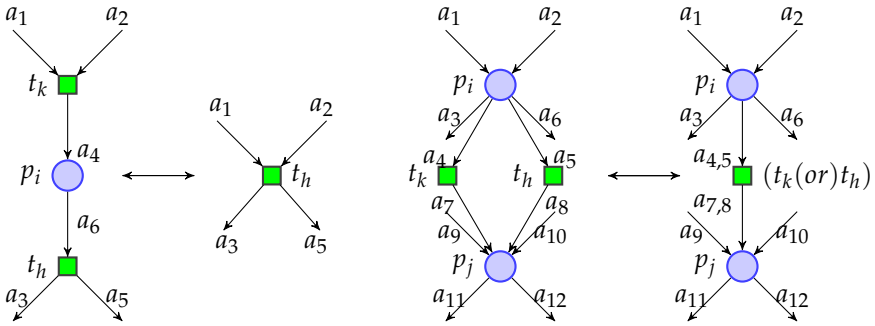
1. Fusion of Series of ELOTs (FSELOT) as depicted in Figure 3.32(6). t_k and t_h are both ELOTs. Moreover, the place p_i does not contain any token. The place p_i has one input arc and one output arc. The input arc of p_i is the only one output arc of its pre-transition (t_k). The output arc of p_i is the only one input arc of its post-transition (t_h). The ELOT t_k and the place p_i are suppressed;

2. Fusion of Parallel ELOTs (FPELOT) as depicted in Figure 3.32(7). t_k and t_h are both ELOTs. Each of the two transitions has one input arc and one output arc. The pre-place (respectively the post-place) of these two transitions is the same. The two ELOTs are merged as one ELOT. The merged ELOT is denoted as $t_k(or)t_h$,

which means if this transition is fired, t_k or t_h is fired. The label of this transition is $\mathcal{L}(t_k)(or)\mathcal{L}(t_h)$;

Remark 1: The condition of using these two rules is that t_k and t_h are ELOTs. After using these rules, the merged transition is also an ELOT. We can continue applying the reduction rules on this ELOT.

Remark 2: The first transition (t_k) contains only one output arc in rule (6) (similar to the rule in Figure 3.27(2)b). It means that the firing of transition t_k can only enable the following transition t_h.



(6) FSELOT (7) FPELOT

Figure 3.32 – Reduction rules for ELOTs

Theorem 4 *By using reduction rules (6) and (7), the diagnosability of the reduced LPN model keeps consistent with the diagnosability of the initial LPN model.*

Proof: First of all, as presented in the proof of Proposition 1, it remains true that, by using the reduction rules (6) and (7), the firing of the previous transitions and following transitions is not affected. Afterwards, we prove respectively the preservation of the diagnosability for the rules (6) and (7).

For rule (6) in Figure 3.32(6), it is worth noticing that, in the initial LPN model, the firing of the transition t_k can only enable its following transition t_h. Rule (6) can be used only if the intermediary place p_i does not contain any token. It means that when the transition t_h is fired, it is certain that t_k has been fired. Moreover, the place p_i is not the pre-place of any transition except of t_h. So if the transition t_h is enabled, it can be fired after no matter which sequence of transitions.

Let us define the projection operator $P_{t_k} : T \rightarrow T \backslash \{t_k\}$.

If the initial system is non-diagnosable, there exist two sequences σ_1 and σ_2 with the same observation, such that σ_1 does not contain any fault, but σ_2 contains a fault and can be arbitrarily long after the occurrence of the fault. Two cases must be studied:

1. $t_k, t_h \notin \sigma_1$ (respectively σ_2): in this case, the language of σ_1 and σ_2 does not change. The reduced LPN model is still non-diagnosable.

2. σ_1 (respectively σ_2)$= \sigma_i t_k \sigma_j t_h \sigma_m$ ($t_h \notin \sigma_j$, but it is possible that $\sigma_j = \lambda$): in this case, because t_k and t_h are ELOTs, it is certain that σ_2 (respectively σ_1)$= \sigma'_i t_k \sigma'_j t_h \sigma'_m$ ($t_h \notin \sigma'_j$). Because the place p_i is the only post-place of t_k and the only pre-place of t_h, the transition t_h can be fired after no matter which sequence of transitions. Therefore, there exist two sequences of transitions $\sigma'_1 = \sigma_i t_k t_h \sigma_j \sigma_m$ and $\sigma'_2 = \sigma'_i t_k t_h \sigma'_j \sigma'_m$ that satisfy the condition for a non-diagnosable system. By using the rule (6), in the reduced LPN, there exist two sequences $\widetilde{\sigma'_1} = P_{t_k}(\sigma'_1)$ and $\widetilde{\sigma'_2} = P_{t_k}(\sigma'_2)$ that satisfy the condition for a non-diagnosable system. Therefore, the reduced LPN system is non-diagnosable.

If the initial system is diagnosable, there does not exist two sequences σ_1 and σ_2 with the same observation, such that σ_1 does not contain any fault, but σ_2 contains a fault and can be arbitrarily long after the occurrence of the fault. By using the rule (6), there does not exist such two sequences by studying the three cases above. Therefore, by using the rule (6), the diagnosability property of the reduced model does not change.

For rule (7) in Figure 3.32(7), t_k and t_h are equivalent. It means that if there exists a sequence of transitions $\sigma = \sigma_i t_k \sigma_j$, the sequence of transitions $\sigma = \sigma_i t_h \sigma_j$ exists (according to structural properties of PN). Moreover, t_k and t_h are ELOTs, so if $\mathcal{L}(t_k)$ (respectively $\mathcal{L}(t_h)$) is observed, t_k (respectively t_h) is certainly fired. If the system is non-diagnosable, assuming that the two sequences of transitions σ_1 and σ_2 contain t_k, that satisfy the condition of non-diagnosable system. By replacing t_k by t_h, the two corresponding sequences σ'_1 and σ'_2 satisfy still the condition for a non-diagnosable system. Therefore, by using the rule (7), there exist two sequences $\widetilde{\sigma_1}$ and $\widetilde{\sigma_2}$ corresponding to σ_1 and σ_2 by replacing t_k by $(t_k(or)t_h)$. In a similar way, if the initial LPN model is diagnosable, the reduced LPN model is also diagnosable by using rule (7). Therefore, by using the rule (7), the diagnosability property of the reduced model does not change.

Overall, By using the reduction rules (6) and (7), the diagnosability of the reduced LPN model keeps consistent with the diagnosability of the initial LPN model. □

The Theorem 4 proves that the reduction rules (6) and (7) do not change the diagnosability property of the system. Therefore, they can be applied to simplify further the initial LPN model so as to reduce the combinatorial explosion problem.

Example 31 *Considering the reduced LPN model in Figure 3.28. The transitions t_2, t_4, t_5, t_7 and t_{10} are all ELOTs. By using the reduction rule (6) FSELOT, the ELOT t_2 is suppressed. The reduced LPN model is called a "further reduced LPN" hereafter (shown in Figure 3.33). To analyze the diagnosability of the further reduced LPN, the reachability graph (shown in Figure 3.34) contains 10 nodes and the diagnoser (shown in Figure 3.35 (solid part)) contains 10 nodes. The markings*

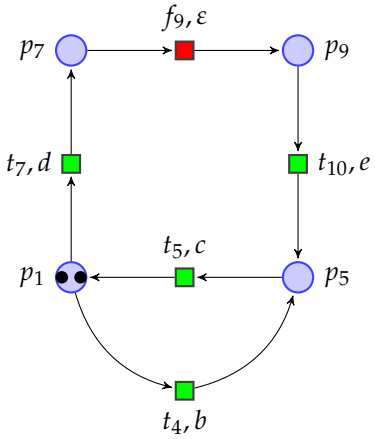


Figure 3.33 – Further reduced LPN model

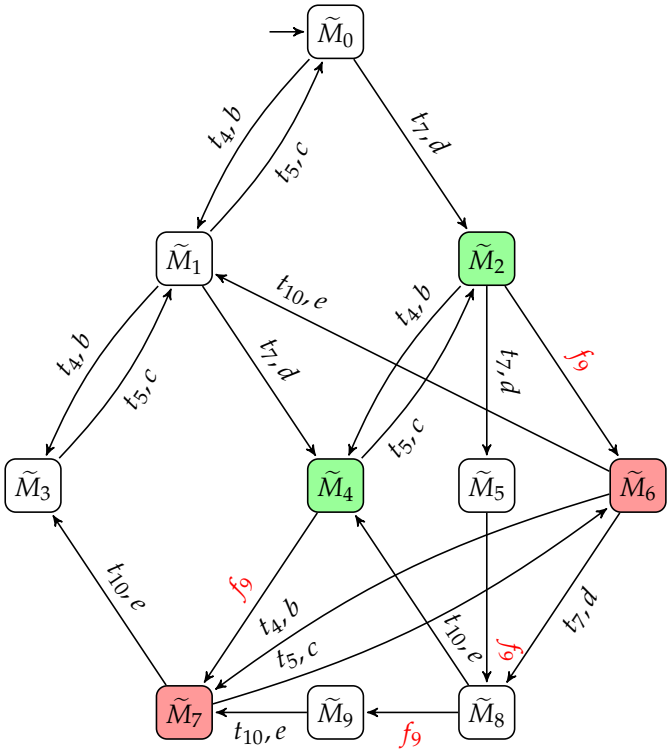


Figure 3.34 – Reachability graph of the further reduced LPN shown in Figure 3.33

are shown in Table 3.5. The system remains non-diagnosable and the indeterminate cycle is shown in the shadow zone of Figure 3.35.

By using the reduction rules (6) and (7), the diagnosability property is preserved. The number of the nodes in the reachability graph in Figure 3.34 (respectively the diagnoser in Figure 3.35) is further reduced comparing to the reachability graph in Figure 3.29 (respectively the diagnoser in Figure 3.30). Therefore, the combinatorial explosion problem is reduced for diagnosability analysis.

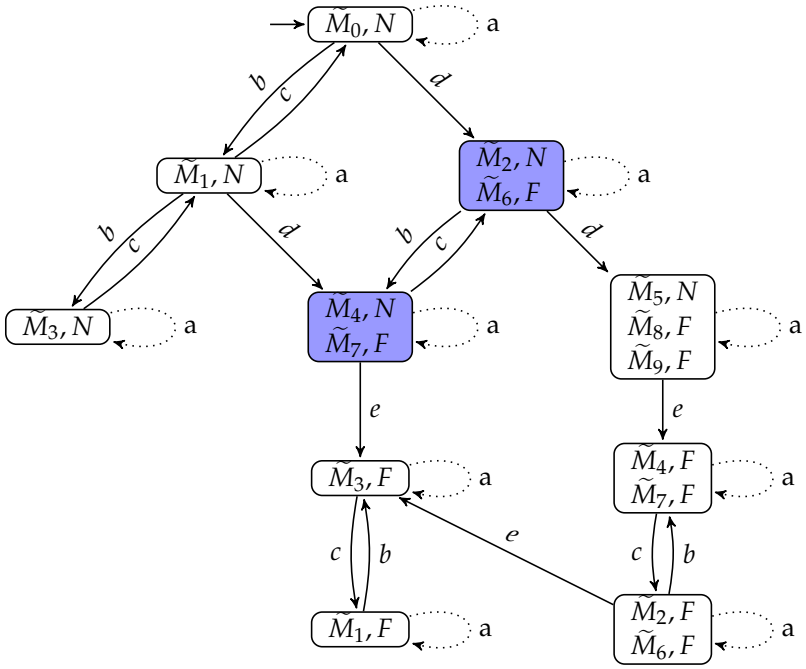


Figure 3.35 – Diagnoser of the further reduced LPN shown in Figure 3.33

Table 3.5 – Markings in Figure 3.34 and Figure 3.35

j	$\widetilde{M}_j$	j	$\widetilde{M}_j$
0	$[2\,0\,0\,0]^\tau$	5	$[0\,0\,2\,0]^\tau$
1	$[1\,1\,0\,0]^\tau$	6	$[1\,0\,0\,1]^\tau$
2	$[1\,0\,1\,0]^\tau$	7	$[0\,1\,0\,1]^\tau$
3	$[0\,2\,0\,0]^\tau$	8	$[0\,0\,1\,1]^\tau$
4	$[0\,1\,1\,0]^\tau$	9	$[0\,0\,0\,2]^\tau$

3.2.1.3 Impact of the reduction rules on the on-line diagnosis

By using the rules (1)-(5), the language of the LPN model does not change. Therefore, the on-line diagnosis by using the diagnoser of the reduced model remains valid for on-line diagnosis.

By using the rules (6) and (7), some ELOTs with their observable labels are suppressed. Since some observable labels are removed, the language of the LPN model changes. However, in this part, the diagnoser of the reduced LPN model remains valid for on-line diagnosis.

Proposition 2 *By using the reduction rules (6) and (7), the diagnoser of the further reduced LPN remains valid for on-line diagnosis.*

Proof: We prove at first that the result for on-line diagnosis is correct by using the rule (6). For the initial LPN model, when the event $\mathcal{L}(t_k)$ (t_k is the first ELOT in Figure 3.32(6))

is observed, the only consequence is that the ELOT t_h is enabled (i.e. the event $\mathcal{L}(t_h)$ can be observed after that). Assuming that, after observing the event $\mathcal{L}(t_k)$ by using the initial LPN model, the diagnoser of initial LPN model changes its state from $\mathcal{D}$ to $\mathcal{D}'$. We can deduce that the labels of the state $\mathcal{D}$ and $\mathcal{D}'$ are the same (the 3 labels of the state in the diagnoser are "Normal", "F-certain" and "F-uncertain" as it is defined in [Sam+95]). If the transition t_k is suppressed by using the rule (6), when $\mathcal{L}(t_k)$ is observed, the state of the system does not change. Therefore, by using the reduction rules (6), the result of on-line diagnosis is correct.

For the rule (7), when $\mathcal{L}(t_k)$ (respectively $\mathcal{L}(t_h)$) is observed, we can consider that $\mathcal{L}(t_k)(or)\mathcal{L}(t_h)$ is observed. The result of on-line diagnosis is correct.

Overall, it is proved that the diagnoser of the further reduced LPN by using the reduction rules (6) and (7) is valid for on-line diagnosis.□

Example 32 *Considering the diagnoser in Figure 3.30, the labels of the states on the both sides of event "a" are the same. (e.g. $(\widehat{M}_0, N) \xrightarrow{a} (\widehat{M}_0, N)$; $(\widehat{M}_4, F) \xrightarrow{a} (\widehat{M}_8, F)$; $(\widehat{M}_2, N|\widehat{M}_6, F) \xrightarrow{a} (\widehat{M}_5, N|\widehat{M}_{10}, F)$).*

Assuming the ELOT t_k (Figure 3.32(6)) is suppressed by using the rule (6). By using the diagnoser of the further reduced LPN, when $\mathcal{L}(t_k)$ is observed, the state of the dignoser does not change. To make the diagnoser deterministic, we can add a self-loop transition labeled by $\mathcal{L}(t_k)$ to every state. For example, since the ELOT (t_2, a) of the LPN in Figure 3.28 is suppressed, a self-loop transition labeled by a (dashed part in Figure 3.35) is added to every state of the diagnoser in Figure 3.35.

Example 33 *Assuming that from the initial state of the system, the observed sequence of events is adb.*

By using the diagnoser in Figure 3.30, while adb is observed, the state of diagnoser changes as: $(\widehat{M}_0, N) \xrightarrow{a} (\widehat{M}_1, N) \xrightarrow{d} (\widehat{M}_5, N|\widehat{M}_{10}, F) \xrightarrow{b} (\widehat{M}_9, N|\widehat{M}_{13}, F)$. The diagnosis result is $Normal \xrightarrow{a} Normal \xrightarrow{d} F-uncertain \xrightarrow{b} F-uncertain$.

By using the diagnoser in Figure 3.35, after observing a, the state of the diagnoser does not change (dashed self-loops in Figure 3.35). Therefore, while adb is observed, the state of diagnoser changes as: $(\widetilde{M}_0, N) \xrightarrow{a} (\widetilde{M}_0, N) \xrightarrow{d} (\widetilde{M}_2, N|\widetilde{M}_6, F) \xrightarrow{b} (\widetilde{M}_4, N|\widetilde{M}_7, F)$. The diagnostic result is $Normal \xrightarrow{a} Normal \xrightarrow{d} F-uncertain \xrightarrow{b} F-uncertain$, which keeps consistent with the previous result.

Remark 3: By using the rule (6), the first ELOT t_k in Figure 3.32(6) is suppressed because for on-line diagnosis, when the label $\mathcal{L}(t_h)$ is observed, it can be deduced that the label $\mathcal{L}(t_k)$ has been observed before. However, conversely, it is not true.

Remark 4: Practically, if the rule (6) can be used to reduce the LPN model, the sensor, that tests the label of the first ELOT (t_k in Figure 3.32(6)), is redundant. From diagnostic viewpoint, this sensor can be removed to reduce the cost for constructing the system.

By using the reduction rules (6) and (7), the size of the diagnoser [Sam+95] for on-line diagnosis is smaller.

It is worth noticing that by reducing the initial LPN model, this technique is complementary to most of diagnosability analysis techniques for LPN systems. It does not reduce the complexity of diagnosability methods, but from a practical point of view, it makes them more efficient by allowing them to work on reduced state spaces. For example, the Verifier Net (VN) in [Cab+12] of the reduced model can be much smaller, so as to the reachability graph (or coverability graph) of the VN. For the on-the-fly and incremental technique in [Liu+14], the nodes generated by these regular unobservable transitions or ELOTs can be reduced while executing the state estimation. For the approach using ILP in [Bas+12], less possibilities of the unobservable sequence of transitions will be produced. Moreover, if some ELOTs are suppressed, it is sufficient to check the sub-language of the initial LPN model, so that the efficiency can be improved. Even for the approach using minimal explanations [Cab+09a; JB10; Li+15b], by using the rules (1)-(5) the efficiency of the algorithm may be improved, because the regular unobservable transitions are normally considered and computed several times while calculating the minimal explanations of observable transitions or fault transitions.

3.2.2 Sufficient condition of diagnosability for *safe* and *live* LPN

This section supplements the sufficient condition for diagnosability of a *safe* and *live* LPN given in [Wen+05]. According to [Wen+05], a *safe* and *live* LPN is diagnosable, if there do not exist two minimal T-invariants such that their sets of observable labels are the same (Section 3.1.2.1 page 38).

There is a defect of this sufficient condition, because the fault information is not included for the diagnosability analysis. There may exist two minimal T-invariants $\overrightarrow{\Omega}_{min,1}$ and $\overrightarrow{\Omega}_{min,2}$ such that: (1) $\Sigma_L(\overrightarrow{\Omega}_{min,1}) = \Sigma_L(\overrightarrow{\Omega}_{min,2})$; (2) $\overrightarrow{\Omega}_{min,1}, \overrightarrow{\Omega}_{min,2} \in \mathcal{I}_N$ or $\overrightarrow{\Omega}_{min,1}, \overrightarrow{\Omega}_{min,2} \in \mathcal{I}_F$. Such a couple of minimal T-invariants is useless for diagnosability analysis. Moreover, there are some situations that are not taken into account in this sufficient condition. A counter-example is presented in Figure 3.36.

For the LPN in Figure 3.36, there exist four minimal T-invariants: $\overrightarrow{\Omega}_{min,1} = [1\,0\,0\,0\,0\,0\,0\,0\,1]^\tau$, $\overrightarrow{\Omega}_{min,2} = [0\,1\,1\,0\,0\,0\,0\,0\,1]^\tau$, $\overrightarrow{\Omega}_{min,3} = [0\,0\,0\,1\,1\,0\,0\,0\,1]^\tau$, and $\overrightarrow{\Omega}_{min,4} = [0\,0\,0\,0\,0\,1\,1\,1\,1]^\tau$. $\Sigma_L(\overrightarrow{\Omega}_{min,1}) = \{a\}$, $\Sigma_L(\overrightarrow{\Omega}_{min,2}) = \{b, c\}$, $\Sigma_L(\overrightarrow{\Omega}_{min,3}) = \{c\}$, and $\Sigma_L(\overrightarrow{\Omega}_{min,4}) = \{a, a, b\}$. According to the sufficient condition in [Wen+05], the system is diagnosable because there do not exist two minimal T-invariants $\overrightarrow{\Omega}_{min,i}$ and $\overrightarrow{\Omega}_{min,j}$ s.t. $\Sigma_L(\overrightarrow{\Omega}_{min,i}) = \Sigma_L(\overrightarrow{\Omega}_{min,j})$.

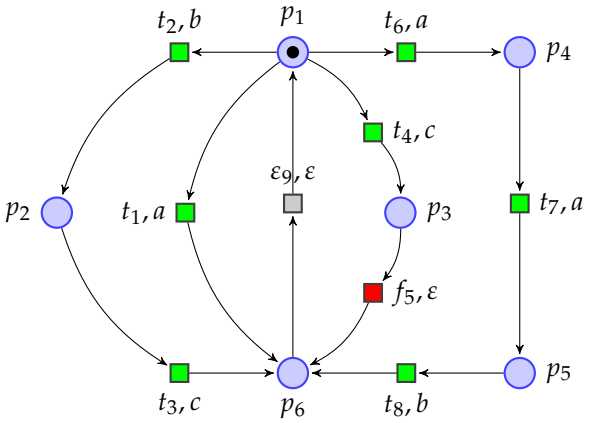


Figure 3.36 – A counter-example of the sufficient condition in [Wen+05]

However, the LPN in Figure 3.36 is non-diagnosable: there exist two sequences of transitions: $\sigma_1 = (t_1\varepsilon_9 t_1\varepsilon_9 t_2 t_3\varepsilon_9)^*$ and $\sigma_2 = (t_6 t_7 t_8\varepsilon_9 t_4 f_5\varepsilon_9)^*$ s.t. σ_1 and σ_2 have the same observation $((aabc)^*)$ and can be arbitrarily long. Moreover, σ_2 contains a fault transition, but σ_1 does not. According to the definition of diagnosability, the LPN is undiagnosable.

Therefore, the sufficient condition for the diagnosability of a *safe* and *live* LPN needs to be supplemented.

Proposition 3 *A* safe *and* live *LPN is diagnosable, if there do not exist two T-invariants* $\overrightarrow{\Omega}_i$ *and* $\overrightarrow{\Omega}_j$*, which can be the linear combination of minimal T-invariants, such that:*

1. $\Sigma_L(\overrightarrow{\Omega}_i) = \Sigma_L(\overrightarrow{\Omega}_j)$;
2. $\overrightarrow{\Omega}_i$ *contains a fault transition but* $\overrightarrow{\Omega}_j$ *does not.*

In Figure 3.36, there exist two T-invariants: $\overrightarrow{\Omega_1} = 2 \cdot \overrightarrow{\Omega}_{min,1} + \overrightarrow{\Omega}_{min,2}$ and $\overrightarrow{\Omega_2} = \overrightarrow{\Omega}_{min,3} + \overrightarrow{\Omega}_{min,4}$ s.t. $\Sigma_L(\overrightarrow{\Omega}_1) = \Sigma_L(\overrightarrow{\Omega}_2)$. $\overrightarrow{\Omega}_2$ contains a fault transition but $\overrightarrow{\Omega}_1$ does not. Thus, it cannot be said that the system is diagnosable and it is necessary to build the state space in order to analyze the diagnosability of this LPN.

Linear programing technique can be used to analyze the diagnosability by using the sufficient conditions.

Let us recall some notions in Section 3.1.2.1. $\mathcal{I}_N$ is the set of minimal T-invariants that does not contain any fault transition. $\mathcal{I}_F$ is the set of minimal T-invariants that contains a fault transition. The minimal T-invariant in $\mathcal{I}_N$ is denoted as $\overrightarrow{\Omega}^N_{min,i}$ and the minimal T-invariant in $\mathcal{I}_F$ is denoted as $\overrightarrow{\Omega}^F_{min,i}$. $\overrightarrow{V}(\overrightarrow{\Omega}_{min}) \in \mathbb{N}^{|\Sigma_o|}$ is called a label vector of the minimal T-invariant $\overrightarrow{\Omega}_{min}$). Assuming $|\Sigma_o| = m$, $\Sigma_o = \{l_1, \cdots, l_m\}$. The label vector of $\overrightarrow{\Omega}_{min}$ is

$$\overrightarrow{V}(\overrightarrow{\Omega}_{min}) = \begin{matrix} l_1 \\ l_2 \\ \cdots \\ l_m \end{matrix} \begin{bmatrix} \overrightarrow{V}(\overrightarrow{\Omega}_{min})^1 \\ \overrightarrow{V}(\overrightarrow{\Omega}_{min})^2 \\ \cdots \\ \overrightarrow{V}(\overrightarrow{\Omega}_{min})^m \end{bmatrix};$$

where for $j \in \{1, \cdots, m\}$, $\overrightarrow{V}(\overrightarrow{\Omega}_{min})^j = \sum_h \overrightarrow{\Omega}^h_{min}$, and $\forall h$, $\mathcal{L}(t_h) = l_j$. ($\overrightarrow{V}(\overrightarrow{\Omega}_{min})^j$ is the $j-$th component of $\overrightarrow{V}(\overrightarrow{\Omega}_{min})$ and $\overrightarrow{\Omega}^h_{min}$ is the $h-$th component of $\overrightarrow{\Omega}_{min}$.)

Example 34 *In Figure 3.36, there exists a minimal T-invariant* $\overrightarrow{\Omega}_{min,4} = [0\,0\,0\,0\,0\,1\,1\,1\,1]^\tau$ *and* $\Sigma_L(\overrightarrow{\Omega}_{min,4}) = \{a, a, b\}$. *The label vector of* $\overrightarrow{\Omega}_{min,4}$ *is*

$$\overrightarrow{V}(\overrightarrow{\Omega}_{min,4}) = \begin{matrix} a \\ b \\ c \end{matrix} \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$

The sufficient condition of diagnosability can be translated into the following constraint:

$$\begin{cases} \sum_i (a_i \cdot \overrightarrow{V}(\overrightarrow{\Omega}^N_{min,i})) = \sum_i (a'_i \cdot \overrightarrow{V}(\overrightarrow{\Omega}^N_{min,i})) + \sum_r (b_r \cdot \overrightarrow{V}(\overrightarrow{\Omega}^F_{min,r})) \\ a_i \cdot a'_i = 0 \\ \sum_i a_i \geq 1 \\ \sum_r b_r \geq 1 \\ a_i, a'_i, b_r \in \mathbb{N} \end{cases} \tag{3.2}$$

where a_i, a'_i and b_r are integer variables.

The first inequality represent two T-invariants (two linear combinations of minimal T-invariants), such that: $\Sigma_L(\overrightarrow{\Omega}_i) = \Sigma_L(\overrightarrow{\Omega}_j)$; $\overrightarrow{\Omega}_i$ contains a fault transition but $\overrightarrow{\Omega}_j$ does not.

We can simplify the constraints as follows:

$$\begin{cases} \sum_i (d_i \cdot \overrightarrow{V}(\overrightarrow{\Omega}^N_{min,i})) = \sum_r (b_r \cdot \overrightarrow{V}(\overrightarrow{\Omega}^F_{min,r})) \\ \sum_r b_r \geq 1 \\ if\ d_i < 0 \ \Rightarrow \ a_i = 0; a'_i = d_i \\ if\ d_i > 0 \ \Rightarrow \ a_i = d_i; a'_i = 0 \\ b_r \in \mathbb{N},\ d_i \in \mathbb{Z} \end{cases} \tag{3.3}$$

Proposition 4 *A* safe *and* live *LPN is diagnosable, if the Constraint 3.3 has no solution.*

Proof: The proposition can be proved by using directly Proposition 3

It is worth noticing that the Proposition 3 and Proposition 4 are just sufficient conditions for the diagnosability of an LPN. If there exist two T-invariants as was presented in Proposition 3 or the Constraint 3.3 has a solution, in this case, the only way to analyze the diagnosability of the system is to construct the state space in order to verify the existence of an indeterminate cycle.

3.2.3 On-the-fly diagnosability analysis using minimal explanations

In this section, a new approach is proposed to improve the on-the-fly diagnosability analysis by using minimal explanations.

In the literature review, the MBRG/BRD approach [Cab+14] and the on-the-fly approach [Liu+14] are mentioned to solve the combinatorial explosion problem for diagnosability analysis. They tackle the combinatorial explosion from different points of view. The MBRG/BRD approach gives a compact manner to build the state space by using minimal explanations. The on-the-fly approach generally builds, on the fly, the FM-graph and FM-set tree in parallel instead of building the whole state space. Both approaches are proposed to avoid building the whole state space of the system *a priori*. Compared with the diagnoser approach [Sam+95], they are both capable to save memory and reduce combinatorial explosion for diagnosability analysis.

In this section, the basis fault marking (BFM) is introduced, which consists of a marking and a binary tag indicating the occurrence of a fault. Two structures are developed, namely basis fault marking graph (BFG) and basis fault marking set tree (BFST). The BFG is similar to a reachability graph and the BFST works as a diagnoser. However, they are built on the fly and in parallel with stop conditions. Every node in BFG is a BFM and every node in BFST is named basis fault marking set (BFS) that is a set of BFMs obtained by firing a sequence of events, whose last event is regular and observable.

The unobservable fault transitions are technically treated, while using minimal explanations:

1. All the fault transitions w.r.t T_f^i are processed as "observable" transitions, while building the BFG, i.e. minimal explanations, as well as justifications, are restricted to regular unobservable transitions. When a cycle is found in BFST, it is necessary to verify the existence of two corresponding cycles in BFG with the same observations: one contains a fault but the other one does not. In order to keep all the fault propagation information to verify the indeterminate cycle with the help of BFG, the BFMs obtained by firing the fault transitions are built in BFG. The same idea is used in [Cab+14] while constructing the MBRG to make up for the insufficiency of BRG in [Cab+10b] for diagnosability analysis.

2. Since it is important to know the possibility to fire a fault transition from a given state while constructing the BFG by using minimal explanations, all the branches labeled by a fault event are exhaustively investigated before the construction of the branches labeled by a regular observable event. Moreover, it is also necessary to get the F-uncertain BFS.

According to the above illustration, the notions used in this section are defined as follows:

Let us consider $T = T_o^i \dot{\cup} T_u^i$ where $T_o^i = T_o \dot{\cup} T_f^i$ and $T_u^i = T_u \setminus T_f^i$. C_u^i (resp. C_o^i) denotes the restriction of the incidence matrix C, which refers to T_u^i (resp. T_o^i). Moreover, let $P_{u,t}^i \colon T^* \to T_u^{i*}$ be the projection which removes the transitions in T_o^i from a sequence $\sigma \in T^*$ and $P_{o,t}^i \colon T^* \to T_o^{i*}$ be the projection which removes the transitions in T_u^i from $\sigma \in T^*$. The transition labeling function w.r.t the fault class T_f^i is $\mathcal{L}^i \colon T \to \Sigma$, where $\Sigma = \Sigma_o^i \dot{\cup} \{\varepsilon\}$. Then, $\forall u \in T_u^i$, $\mathcal{L}^i(u) = \varepsilon$ and $\forall t_f^i \in T_f^i$, $\mathcal{L}^i(t_f^i) = f^i$, $f^i \in \Sigma_o^i$. The labeling function can be extended to $\mathcal{L}^i \colon T^* \to \Sigma^*$. It is important to notice that minimal explanations are restricted to all the transitions in T_u^i which represent the set of regular unobservable transitions.

The composition of a BFM is the same with an FM in [Liu+14].

Definition 22 *A BFM is a vector $BFM^i \in \mathbb{N}^{|P|+1}$ upon a fault class T_f^i:*

$$BFM^i = \begin{bmatrix} mark(BFM^i) \\ fault(BFM^i) \end{bmatrix}$$

.

The initial BFM is BFM_0^i, where $mark(BFM_0^i) = M_0$ and $fault(BFM^i) = 0$. The set of BFMs is denoted as $\mathcal{Q}^{b,i}$. A BFM BFM^i belongs to $\mathcal{Q}^{b,i}$, iff:

1. $\exists \sigma \in T^*$, $\omega \in \Sigma_o^{i*}$, *s.t.*, $\mathcal{L}^i(\sigma) = \omega, \sigma_u^i = P_{u,t}^i(\sigma)$, $\sigma_u^i \in \hat{\mathcal{J}}(\omega)$ *(defined in Definition 13 (page 41)):* $mark(BFM_0^i)\ [\ \sigma > mark(BFM^i)$;

2. $fault(BFM^i) = 1$, *if* $\exists t_f \in T_f^i, t_f \in \sigma$; *otherwise,* $fault(BFM^i) = 0$.

Given two BFMs BFM_1^i, $BFM_2^i \in \mathcal{Q}^{b,i}$, it is denoted that $BFM_1^i\ [\ \sigma > BFM_2^i$ iff $\exists \sigma \in T^*$, $\omega \in \Sigma_o^{i*}$, s.t., $\mathcal{L}^i(\sigma) = \omega, \sigma_u^i = P_{u,t}^i(\sigma)$, $\sigma_u^i \in \hat{\mathcal{J}}(\omega)$: $mark(BFM_1^i)\ [\ \sigma > mark(BFM_2^i)$; and $fault(BFM_2^i) = fault(BFM_1^i)$ if $\forall j, \sigma^j \notin T_f^i$, otherwise, $fault(BFM_2^i) = 1$.

It is worth noticing that the set of BFMs is a subset of the set of fault markings (FMs) defined in [Liu+14]: an FM belongs to the set of FMs but not to the set of BFMs, if it is reached by firing $\sigma' \in T^*$, where the last transition of σ' belongs to T_u^i. The set of BFMs gives a more compact state space of the system.

A BFG is considered as a directed non-deterministic graph relative to the fault class T_f^i. Each node indicates a given BFM and each arc indicates a regular observable transition or a fault transition with its label and its minimal e-vector (Definition 11 page 40).

Definition 23 *The BFG relative to a fault class T_f^i and called BFG^i is a tuple $(\mathcal{N}^{b,i}, BFM_0^i, \Sigma_o^i, \eta)$, where:*

- $\mathcal{N}^{b,i} \subseteq \mathcal{Q}^{b,i}$ *is a set of BFM nodes (because the BFG is built on the fly with stop conditions, $\mathcal{N}^{b,i}$ is a subset of $\mathcal{Q}^{b,i}$);*
- $BFM_0^i = [M_0^\tau, 0\,]^\tau$ *is the initial node;*
- Σ_o^i *is a finite set of observable events and fault events in T_f^i;*
- $\eta : \mathcal{N}^{b,i} \times \Sigma_o^{i*} \to \mathcal{N}^{b,i}$ *is the transition function of BFM: given $BFM_1^i \in \mathcal{Q}^{b,i}$ and $\omega \in \Sigma_o^{i*}$, $\eta(BFM_1^i, \omega) = \{BFM_2^i \mid \exists\ \sigma \in T^*$ s.t. $\mathcal{L}^i(\sigma) = \omega,\ BFM_1^i\ [\ \sigma > BFM_2^i,\ \sigma_u^i = P_{u,t}^i(\sigma),\ \sigma_u^i \in \hat{\mathcal{J}}(\omega)\}$.*

All the BFMs in BFG^i can be reached by firing a sequence of transitions σv where $v \in T_o^i$.

The η function in Definition 23 is given in Algorithm 1 (the Algorithm 1 illustrates the η function and the construction of the nodes in BFG). From the initial BFM, all the branches labeled by a fault event with its minimal explanation are exhaustively investigated *a priori* (line 10-22 of Algorithm 1). All the obtained nodes in the set $\mathcal{G}$ are built in BFG (line 19-20 of Algorithm 1). Afterwards, from all these nodes in $\mathcal{G}$, the next nodes are computed by firing all the possible transitions labeled by a selected observable event with their minimal explanations (line 24-36 of Algorithm 1). All these obtained nodes are in set $\mathcal{F}$. The sets $\mathcal{G}$ and $\mathcal{F}$ store separately the BFM nodes of the two steps for the purpose of the following construction of the BFS nodes in BFST.

Algorithm 1 Algorithm for η function in Definition 23

Input: a BFM, a regular observable event e and T_f^i;
Output: $[\mathcal{G},\ \mathcal{F}] = \eta(BFM, e, T_f^i)$;
function $\eta(BFM, e, T_f^i)$
    $\mathcal{G} \leftarrow \emptyset$; ▷ $\mathcal{G}$ is the set of BFMs reached from BFM just after the occurrence of a faulty transition in BFG.
    $\mathcal{F} \leftarrow \emptyset$; ▷ $\mathcal{F}$ is the set of BFMs reached from BFM just after the occurrence of e.
    $\mathcal{G}_{temp} \leftarrow \{BFM\}$; ▷ $\mathcal{G}_{temp}$ is the set of BFMs which is a temporary variable.
    $\mathcal{F}_{temp} \leftarrow \emptyset$; ▷ $\mathcal{F}_{temp}$ is the set of BFMs which is a temporary variable.
    $\mathcal{G}_{hist} \leftarrow \emptyset$; ▷ $\mathcal{G}_{hist}$ is the set of BFMs, which is a temporary variable, to store all the BFMs in $\mathcal{G}_{temp}$ that has been calculated.
    $\mathcal{F}_{hist} \leftarrow \emptyset$; ▷ $\mathcal{F}_{hist}$ is the set of BFMs, which is a temporary variable, to store all the BFMs in $\mathcal{F}_{temp}$ that has been calculated.
    while $\mathcal{G}_{temp} \neq \emptyset$ **do**
        Choose $BFM' \in \mathcal{G}_{temp}$;
        $\mathcal{G}_{temp} \leftarrow \mathcal{G}_{temp} / BFM'$;
        $\mathcal{G}_{hist} \leftarrow \mathcal{G}_{hist} \cup BFM'$;
        for all $t_f^i \in T_f^i$ **do**
            for all $\overrightarrow{e_v} \in Y_{min}(mark(BFM'), t_f^i)$ **do**
                $mark(BFM'') \leftarrow mark(BFM') + C_u^i \cdot \overrightarrow{e_v} + C(\cdot, t_f^i)$;
                $fault(BFM'') = 1$;
                $\mathcal{G} \leftarrow \mathcal{G} \cup BFM''$;
                $\mathcal{N}^b \leftarrow \mathcal{N}^b \cup BFM''$ ▷ $\mathcal{N}^b$ (global variable) is the set of the BFG nodes.
                $\mathcal{A}^\eta \leftarrow \mathcal{A}^\eta \cup \{(BFM', (t_f^i, \overrightarrow{e_v}), BFM'')\}$; ▷ $\mathcal{A}^\eta$ (global variable) is the set of the BFG arcs.
                if $BFM'' \notin \mathcal{G}_{hist}$ **then**
                    $\mathcal{G}_{temp} \leftarrow \mathcal{G}_{temp} \cup BFM''$; **end if**; **end for**; **end for**; **end while**;
    $\mathcal{F}_{temp} \leftarrow G$;
    while $\mathcal{F}_{temp} \neq \emptyset$ **do**
        Choose $BFM' \in \mathcal{F}_{temp}$;
        $\mathcal{F}_{temp} \leftarrow \mathcal{F}_{temp} / BFM'$;
        $\mathcal{F}_{hist} \leftarrow \mathcal{F}_{hist} \cup BFM'$;
        for all $t \in T$, s.t. $\mathcal{L}(t) = e$ **do**
            for all $\overrightarrow{e_v'} \in Y_{min}(mark(BFM'), t)$ **do**
                $mark(BFM'') \leftarrow mark(BFM') + C_u^i \cdot \overrightarrow{e_v'} + C(\cdot, t)$;
                $fault(BFM'') = fault(BFM')$;
                $\mathcal{F} \leftarrow \mathcal{F} \cup BFM''$;
                $\mathcal{N}^b \leftarrow \mathcal{N}^b \cup BFM''$; ▷ $\mathcal{N}^b$ (global variable) is the set of the BFG nodes.
                $\mathcal{A}^\eta \leftarrow \mathcal{A}^\eta \cup \{(BFM', (e, \overrightarrow{e_v'}), BFM'')\}$; ▷ $\mathcal{A}^\eta$ (global variable) is the set of the BFG arcs.
                if $BFM'' \notin \mathcal{F}_{hist}$ **then**
                    $\mathcal{F}_{temp} \leftarrow \mathcal{F}_{temp} \cup BFM''$; **end if**; **end for**; **end for**; **end while**;
    return $[\mathcal{G},\ \mathcal{F}]$;

The BFST is constructed based on and in parallel with the BFG. Let the basis fault

marking set (BFS) power set be $\mathcal{X}^{b,i} = 2^{\mathcal{Q}^{b,i}}$ and the initial BFS $x_0^{b,i} = \{BFM_0^i\}$.

Definition 24 *The BFS transition mapping* $\psi\colon \mathcal{X}^{b,i} \times \Sigma_o \to \mathcal{X}^{b,i}$ *is defined as follows: given a BFS* $x^{b,i} \in \mathcal{X}^{b,i}$ *and event* $e \in \Sigma_o$, $\psi(x^{b,i}, e) = \cup_{BFM_1^i \in x^{b,i}} \{BFM_2^i |\ \exists \sigma_u \in T_u^*\ ,\ \exists t \in T_o, s.t.\ \mathcal{L}(\sigma_u t) = e,\ BFM_1^i\ [\ \sigma_u t > BFM_2^i,\ \sigma_u \in \hat{\mathcal{J}}\ (e)\}$.

All the BFSs in $BFST^i$ are reached by firing a sequence of transitions σt where $t \in T_o$.

Definition 25 *[Liu+14] The tagging function* $tag\colon \mathcal{X}^{b,i} \to \{N, F, U\}$ *is defined as follows:*

$$tag(x) = \begin{cases} N & if \forall\ BFM \in x,\ fault(BFM) = 0 \\ F & if \forall\ BFM \in x,\ fault(BFM) = 1 \\ U & otherwise \end{cases}$$

A BFS x *is also said to be normal (resp. F-certain, F-uncertain) if* $tag(x) = N$ *(resp. F, U). For BFS* x' *reachable from* x*, if* $tag(x) \in \{N, U\}$*, it is possible that* $tag(x') \in \{N, F, U\}$*; whereas if* $tag(x) = F$*, then* $tag(x') = F$*, as faults are assumed to be permanent and, therefore, the F-certain tag is propagated to all the successive BFSs.*

The formal definition of BFST is defined as follows:

Definition 26 *The BFST relative to fault class* T_f^i *and called* $BFST^i$ *is a tuple* $(\mathcal{X}^{b,i}, BFS_0^i, \Sigma_o, \psi)$*, where:*

- $\mathcal{X}^{b,i}$ *is a set of BFS nodes;*
- $BFS_0^i = \{BFM_0^i\}$ *is the initial BFS node;*
- Σ_o *is a finite set of observable events;*
- $\psi\colon \mathcal{X}^{b,i} \times \Sigma_o^* \to \mathcal{X}^{b,i}$ *is the transition function of BFS defined in Definition 24.*

The ψ function (defined in Definition 24 and used in Definition 26) is developed in Algorithm 2. The η function is called to compute all the following BFM nodes that are reached from the BFM nodes in the input BFS. It needs to be noticed that only the BFM nodes in the set $\mathcal{F}$ are contained in the output BFS' of the ψ function, because according to Definition 24, only the obtained nodes by firing the possible transitions labeled by a given observable event with their minimal explanations (i.e. the nodes in $\mathcal{F}$), are built in a BFS. The nodes in $\mathcal{G}$ does not belong to a BFS, because they are obtained by firing a fault transition.

Without loss of generality, in this section, the diagnosis issue is discussed for a single class of faults. For the simplicity of representation, the superscript i will be omitted w.r.t T_f^i.

Algorithm 2 Algorithm for ψ function of Definition 24

1: Input: a BFS and a regular observable event e;
2: Output: BFS' which is reached from BFS immediately after e;
3: **function** $\psi(BFS, e)$
4: $BFS' \leftarrow \emptyset$;
5: **for all** $BFM \in BFS$ **do**
6: $[\mathcal{G}, \mathcal{F}] \leftarrow \eta(BFM, e)$;
7: $BFS' \leftarrow BFS' \cup \mathcal{F}$; **end for**;
8: **return** BFS';

The principal idea of our approach is implemented in the Algorithm 4 and Algorithm 3. Algorithm 4 is developed for checking the diagnosability of an LPN model. The diagnosability of the system is given according to the verdict n, which is the output of the DIAG function of Algorithm 3.

The ψ function is called in DIAG function to compute the child BFS from the input BFS. DIAG function is a recursive algorithm, but some conditions are given to stop the investigation of a branch of BFST. The Proposition 5 is given to explain Algorithm 3 and to prove that Algorithm 3 terminates and the diagnosability verdict is correct.

Proposition 5 *For a* bounded *and* live *LPN, DIAG function in Algorithm 3 terminates and its diagnosability verdict is correct.*

Proof: First, it needs to prove that the algorithm terminates well for a *bounded* LPN that does not deadlock after firing any fault transition. As presented above, the investigation of a branch of BFST is stopped, when:

1. An F-certain BFS is generated (the investigation of the branch stops immediately, because all the child nodes of an F-certain BFS are still F-certain) (line 15-17 of Algorithm 3);

2. A new normal BFS is equal to an existing one (line 13-14 of Algorithm 3);

3. A new F-uncertain BFS is equal to an existing one (then checking the existence of indeterminate cycle is necessary (line 19-22 of Algorithm 3)).

While building on-the-fly the BFST, for any branch, one of the tree conditions will be satisfied sooner or later, since the LPN system is *bounded* and does not deadlock after firing any fault transition. Therefore, the algorithm terminates well.

It needs to prove that the algorithm covers all the cases while building on-the-fly the BFST. For a generated BFS node BFS',

1. If BFS' is F-certain, it is not necessary to continue the construction, because all of its child nodes will be F-certain;

2. If BFS' is normal and

   (a) If there is already an existing node BFS'' (in $\mathcal{X}^b$) s.t. $BFS' = BFS''$, the investigation of this branch is stopped, since BFS' generates the same branch as BFS'', which has already been considered (line 13-14 of Algorithm 3);

   (b) Otherwise, the construction of this branch needs to be continued (line 7-12 of Algorithm 3).

3. If BFS' is F-uncertain and

   (a) If there is already an existing node BFS'' (in $\mathcal{X}^b$) s.t. $BFS' = BFS''$, the investigation of this branch is stopped. Meanwhile, if x' is in an indeterminate cycle, LPN is non-diagnosable (line 21-22 of Algorithm 3). According to the Proposition 2, if there exists an indeterminate cycle in the system, a corresponding indeterminate cycle can be found in BFST;

   (b) Otherwise, the construction of this branch needs to be continued by recalling the DIAG function (line 24-26 of Algorithm 3).

As presented above, all the cases are considered.

In brief, the algorithm terminates well and its diagnosability verdict is correct.□

Proposition 6 *[Li+15b] For the same LPN model, if an indeterminate cycle exists in the FM-set tree (which works as the diagnoser of the approach in [Liu+14]) with observation ω w.r.t fault class T_f^i, an indeterminate cycle in BFST with observation ω also exists.*

Proof: If an indeterminate cycle exists in the FM-set tree with observation ω w.r.t fault class T_f^i, then there exist at least one normal cycle and one faulty cycle with the same observation ω in FM-graph. First, for the normal cycle, it is assumed that the sequence of transitions is σ with $\mathcal{L}(\sigma) = \omega$. It is denoted that $\sigma_o^i = P_{o,t}^i(\sigma)$ and $\sigma_u^i = P_{u,t}^i(\sigma)$. One $\sigma_u^{i\prime} \in T_{ui}^*$ can be found, s.t. $\pi(\sigma_u^{i\prime}) \leq \pi(\sigma_u^i)$, $(\sigma_o^i, \sigma_u^{i\prime}) \in \hat{\mathcal{J}}(\omega)$. It is noticed that there exists a sequence of transitions σ' with $\sigma_o^i = P_{o,t}^i(\sigma')$ and $\sigma_u^{i\prime} = P_{u,t}^i(\sigma')$. σ' constructs also a normal cycle in FM-graph. Moreover, it is certain that this cycle exists in BFG. Therefore, a normal cycle is found in BFG. Then, for the faulty cycle, it is assumed that the sequence of transitions is v such that $\mathcal{L}(v) = \omega$. It is denoted that $\rho = \mathcal{L}^i(v)$, $v_o^i = P_{o,t}^i(v)$ and $v_u^i = P_{u,t}^i(v)$. One $v_u^{i\prime} \in T_{ui}^*$ can be found, s.t. $\pi(v_u^{i\prime}) \leq \pi(v_u^i)$, $(v_o^i, v_u^{i\prime}) \in \hat{\mathcal{J}}(\rho)$. It is noticed that there exists a sequence of transitions v' with $v_o^i = P_{o,t}^i(v')$ and $v_u^{i\prime} = P_{u,t}^i(v')$. v' constructs also a faulty cycle in FM-graph. Moreover, it is certain that this cycle exists in BFG. Therefore, a faulty cycle is found in BFG. Thus, if an indeterminate cycle exists in the

Algorithm 3 DIAG(): Checking diagnosability by on-the-fly building of BFG and BFST

1: Input: $\mathcal{X}^b, \mathcal{A}^\psi$, a BFS and n;
2: Output: n' is the diagnosability verdict;
3: **function** DIAG($\mathcal{X}^b, \mathcal{A}^\psi, BFS, n$)
4: $n' \leftarrow 1$;
5: **for all** $e \in \Sigma_o$ **do**
6: $BFS' \leftarrow \psi(BFS, e)$; ▷ BFS' is the child node of BFS;
7: **if** $(BFS' \neq \emptyset) \wedge [tag(BFS') = N]$ **then**
8: **if** $BFS' \notin \mathcal{X}^b$ **then**
9: $\mathcal{X}^b \leftarrow \mathcal{X}^b \cup BFS'$; ▷ $\mathcal{X}^b$ is the set of the BFST nodes.
10: $\mathcal{A}^\psi \leftarrow \mathcal{A}^\psi \cup (BFS, e, BFS')$; ▷ $\mathcal{A}^\psi$ is the set of the BFST arcs.
11: $n' \leftarrow DIAG(\mathcal{X}^b, \mathcal{A}^\psi, BFS', n)$;
12: **if** $n' \neq 1$ **then**
13: **return** n'; **end if**;
14: **else**
15: $\mathcal{A}^\psi \leftarrow \mathcal{A}^\psi \cup (BFS, e, BFS')$; **end if**;
16: **else if** $(BFS' \neq \emptyset) \wedge [tag(BFS') = F]$ **then**
17: $\mathcal{X}^b \leftarrow \mathcal{X}^b \cup BFS'$;
18: $\mathcal{A}^\psi \leftarrow \mathcal{A}^\psi \cup (BFS, e, BFS')$;
19: **else if** $(BFS' \neq \emptyset) \wedge [tag(BFS') = U]$ **then**
20: **if** $(\exists BFS'' \in \mathcal{X}^b)(BFS' = BFS'')$ **then**
21: $\mathcal{A}^\psi \leftarrow \mathcal{A}^\psi \cup (BFS, e, BFS')$;
22: **if** BFS' is in an indeterminate cycle **then**
23: **return** $n' \leftarrow 0$; **end if**; ▷ $n' = 0$ denotes that LPN is undiagnosable due to the indeterminate cycle.
24: **else**
25: $\mathcal{X}^b \leftarrow \mathcal{X}^b \cup BFS'$;
26: $\mathcal{A}^\psi \leftarrow \mathcal{A}^\psi \cup (BFS, e, BFS')$;
27: $n' \leftarrow DIAG(\mathcal{X}^b, \mathcal{A}^\psi, BFS', n)$;
28: **if** $n' \neq 1$ **then**
29: **return** n'; **end if**; **end if**; **end if**; **end for**;
30: **return** n'; ▷ $n' = 1$ denotes that LPN is diagnosable.

FM-set tree with observation ω w.r.t fault class T_f^i, a corresponding indeterminate cycle is found in BFST. □

The Proposition 6 proves that although, by using minimal explanations, the number of states in BFG and BFST becomes smaller than that in FM-graph and FM-set tree [Liu+14], the necessary fault propagation information is kept w.r.t fault class T_f^i for diagnosability analysis.

Example 35 *Let us consider again the LPN model Figure 3.17 (page 40). We define the priorities of investigating branches as $a - b - c - d - e$, which is the same with that of Example 25. The Algorithm 1 and the Algorithm 2 are used to build the nodes and arcs in BFG and BFST. The BFG and BFST are built on-the-fly and in parallel. The numeration of BFMs is based on the order of the*

Algorithm 4 MAIN(): algorithm for checking the diagnosability of LPN

```
1: Input: the LPN system LPN, T_o, T_reg and T_f;
2: Output: Diagnosability of LPN;
3: function MAIN(LPN, T_o, T_u, T_f)
4:     N^b ← {BFM_0};                          ▷ N^b is the set of the BFG nodes.
5:     A^η ← ∅;                                ▷ A^η is the set of the BFG arcs.
6:     BFS_0 ← {BFM_0};
7:     X^b ← {BFS_0};                          ▷ X^b is the set of the BFST nodes.
8:     A^ψ ← ∅;                                ▷ A^ψ is the set of the BFST arcs.
9:                                             ▷ N^b, A^η, X^b and A^ψ are global variables.
10:    n ← DIAG(X^b, A^ψ, BFS_0, 1);           ▷ n indicates the diagnosability of LPN.
11:    if n = 1 then
12:        "LPN is diagnosable";
13:    else if n = 0 then
14:        "LPN is undiagnosable"; end if;
```

construction.

Every arc in BFG is labeled by the firing transition (observable transition or fault transition), its label and its e-vector (e.g. $a(t_2)$, $\overrightarrow{e_1}$). The fault transition f_9 is processed as an observable transition and it has a higher priority than the other observable transitions. For example, after firing the transition $d(t_7)$, $\overrightarrow{e_3}$ at BFM_0, BFM_6 is obtained. From BFM_6, BFM_7 is built a priori by firing the fault transition f_9, $\overrightarrow{e_4}$ before the construction of BFM_8 by firing $a(t_2)$, $\overrightarrow{e_1}$. The construction is stopped when An indeterminate cycle is found (shown in Figure 3.38). Therefore, the system is not diagnosable.

Table 3.6 – Markings and e-vectors in BFG and BFST

j	M_j	j	M_j
0	$[2\,0\,0\,0\,0\,0\,0\,0\,0\,\vert\,0]^\tau$	10	$[0\,0\,0\,0\,1\,0\,1\,0\,0\,\vert\,0]^\tau$
1	$[1\,0\,1\,0\,0\,0\,0\,0\,0\,\vert\,0]^\tau$	11	$[0\,0\,0\,0\,1\,0\,0\,0\,1\,\vert\,1]^\tau$
2	$[0\,0\,2\,0\,0\,0\,0\,0\,0\,\vert\,0]^\tau$		
3	$[0\,0\,1\,0\,1\,0\,0\,0\,0\,\vert\,0]^\tau$	j	$\overrightarrow{e_j}$
4	$[0\,0\,0\,0\,2\,0\,0\,0\,0\,\vert\,0]^\tau$	1	$[1\,0\,0\,0]$
5	$[1\,0\,0\,0\,1\,0\,0\,0\,0\,\vert\,0]^\tau$	2	$[0\,1\,0\,0]$
6	$[1\,0\,0\,0\,0\,0\,1\,0\,0\,\vert\,0]^\tau$	3	$[0\,0\,1\,0]$
7	$[1\,0\,0\,0\,0\,0\,0\,0\,1\,\vert\,1]^\tau$	4	$[0\,0\,0\,1]$
8	$[0\,0\,1\,0\,0\,0\,0\,0\,1\,\vert\,0]^\tau$		
9	$[0\,0\,1\,0\,0\,0\,0\,0\,1\,\vert\,1]^\tau$		

Comparing to Example 25, fewer nodes are built in BFG and BFST (Figure 3.37 and Figure 3.38) than that of FM-graph and FM-set tree (Figure 3.21 and Figure 3.22). By using minimal explanations, the size of FM-graph and FM-set tree is reduced. The BFG and BFST contain fewer states than the corresponding FM-graph and FM-set tree, especially

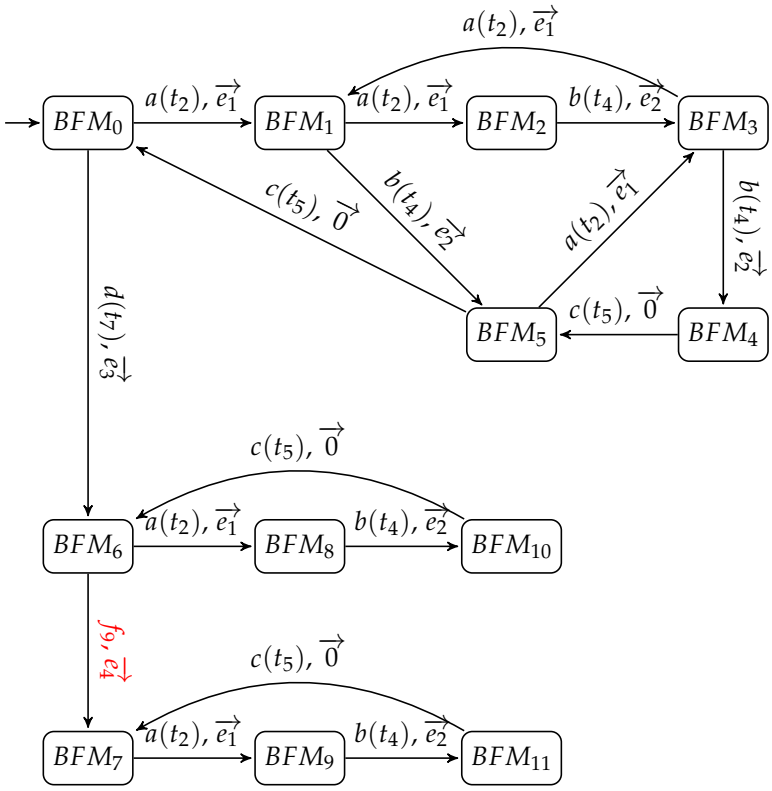


Figure 3.37 – BFG of LPN shown in Figure 3.17

when the LPN contains many regular unobservable transitions. Meanwhile, the necessary information is still kept w.r.t fault class T_f^i for diagnosability analysis. By using minimal explanations, this approach can reduce the combinatorial explosion problem of the on-the-fly diagnosability analysis in [Liu+14], especially when the LPN model contains many regular unobservable transitions. However, its complexity in the worst case is not reduced. The worst cases are as follows

1. There does not exist any regular unobservable transition. Each $F-certain\ BFS$ is found at the end of a branch. The system is diagnosable;

2. There does not exist any regular unobservable transition. Each $F-certain\ BFS$ is found at the end of a branch. The system is not diagnosable but the indeterminate cycle is found at the end of a branch.

In these cases, the whole state space are built, so the complexity of this approach is equal to that of the on-the-fly diagnosability analysis in [Liu+14].

Moreover, the priorities of the investigation of the branches is still not defined. The result of the approach depends on the LPN models because the priorities are defined on the hazard. For example, if we can define the priorities by following the sequence of $d(abc)*$, we can obtain directly the indeterminate cycle in Figure 3.38. Therefore, we

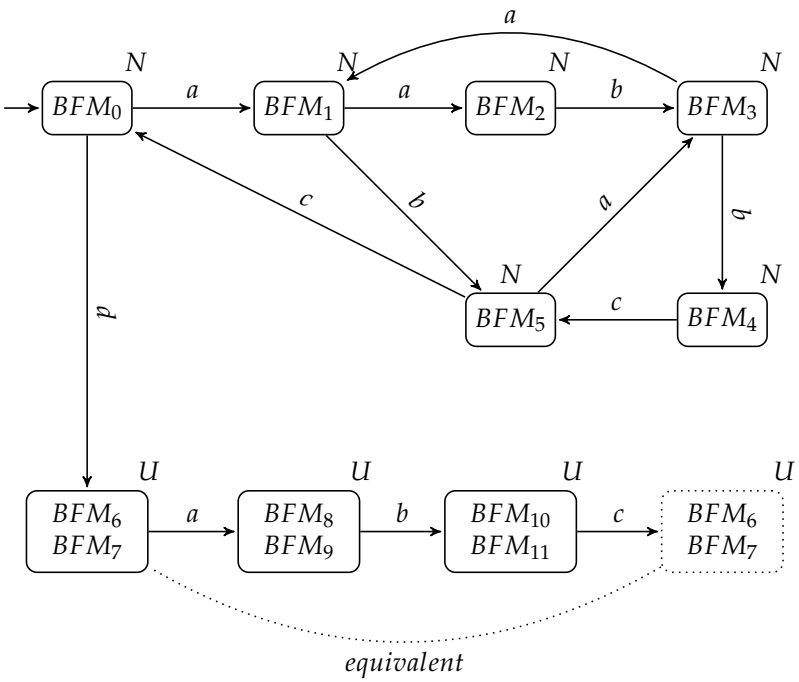


Figure 3.38 – BFST of LPN shown in Figure 3.17

propose an approach to define the priorities in order to find quickly the indeterminate cycle.

3.2.4 On-the-fly diagnosability analysis using T-invariants

In this section, the priorities in the investigation of branches will be defined by using T-invariants. This approach can be applied to the on-the-fly diagnosability analysis in [Liu+14] and the on-the-fly diagnosability analysis using minimal explanations in Section 3.2.3. We will apply this approach to the latter in this section. This approach is developed to orient the investigation of branches, so it does not impact the correctness of the two approaches.

In [Li+15c], we have proposed an approach to define the priorities for the on-the-fly diagnosability analysis of a *bounded* and *live* LPN. In this section, we propose a new approach to remove the strict assumption that the LPN is *live*. It is assumed that, in this section, the LPN is *bounded* and does not deadlock after firing any fault transition.

In this section, only minimal T-invariants (in Definition 9 page 38) are taken into account, because the set of all minimal T-invariants is a basis for all the T-invariants. Moreover, the cycles in reachability graph corresponding to minimal T-invariants are elementary cycles. Therefore, only minimal T-invariants are studied. If there exists a cycle in the reachability graph of a PN, there exists a T-invariant of the PN. The cycle in the reachability graph is really significant for the diagnosability analysis of a LPN system. Therefore, T-invariants of LPNs are likely to be used to give priorities in the investigation

of branches in order to improve the on-the-fly approach and find quickly an existent indeterminate cycle.

To check the indeterminate cycle, it can be observed that all the macro-states in a diagnoser that construct the indeterminate cycle are F-uncertain and the sequence of events in the indeterminate cycle is ω where $\omega \in \mathcal{S}(\overrightarrow{\Omega})$ and $\overrightarrow{\Omega} \in \mathcal{I}_N$ (Recall that $\overrightarrow{\Omega}$ is a minimal T-invariant, $\mathcal{S}(\overrightarrow{\Omega})$ is the set of all the possible firing sequences constructed by the labels in $\Sigma_L(\overrightarrow{\Omega})$ and $\mathcal{I}_N$ is the set of T-invariants that do not contain a fault transition (page 38)).

Definition 27 *The set of* paths *of a transition t from a marking M is defined as follows:*

$$Path(M,t) = \{\sigma \in T^* |\ M[\sigma > M', M' \geq Pre(\cdot,t)\}$$

The corresponding p-vectors (path vectors) is the set:

$$V(M,t) = \{\pi(\sigma) |\ \sigma \in Path(M,t)\}$$

.

A *path* of transition t from a marking M is a sequence of transitions that can be fired, at marking M, in order to enable the transition t. The definition of a *path* is different from the definition of an explanation in Definition 10 (page 40). An explanation is a sequence of unobservable transitions. However, if σ is a *path*, σ belongs to T^*. In other words, an explanation of a transition t from a marking M is also a *path* of transition t from a marking M, but the converse proposition is not true.

Example 36 *Let us consider the LPN in Figure 3.17 (page 40). The initial marking is* $M_0 = [2\,0\,0\,0\,0\,0\,0\,0\,0]^\tau$. *Considering the transition* t_2. *After firing* $\sigma_{u,1} = \varepsilon_1$ *or respectively* $\sigma_{u,2} = \varepsilon_6\varepsilon_1$, *the transition* t_2 *is enabled. Since* $\sigma_{u,1}, \sigma_{u,2} \in T_u^*$, *they are explanations of* t_2 *at* M_0 *as analyzed in Example 20 (page 40). Meanwhile, it can be obtained that* $\sigma_{u,1}, \sigma_{u,2}$ *are also* paths *of* t_2 *at* M_0.

Moreover, after firing $\sigma_1 = \varepsilon_6 t_7 \varepsilon_1$, *the transition* t_2 *is enabled.* σ_1 *is not a explanation of* t_2 *at* M_0, *because* $\sigma_1 \notin T_u^*$. *However,* σ_1 *is a* paths *of* t_2 *at* M_0, *because* $\sigma_1 \in T^*$. *The p-vector of* σ_1 *is* $\overrightarrow{v_1} = [1\,0\,0\,0\,0\,1\,1\,0\,0\,0]^T$.

Definition 28 *The set of* minimal paths *of a transition t from a marking M is defined as follows:*

$$Path_{min}(M,t) = \{\sigma \in Path(M,t) |\ \nexists \sigma' \in Path(M,t) s.t., \pi(\sigma') < \pi(\sigma)\}$$

The corresponding minimal p-vectors is the set:

$$V_{min}(M,t) = \{\pi(\sigma) |\ \sigma \in Path_{min}(M,t)\}$$

.

Example 37 *Let us consider the LPN in Figure 3.17 (page 40). The initial marking is* $M_0 = [2\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^\tau$. *Considering the transition* t_5. *Since at* M_0, *after firing* $\sigma_1 = \varepsilon_1 t_2 \varepsilon_3 t_4$, $\sigma_2 = \varepsilon_6 t_7 \varepsilon_8 f_9 t_{10}$, *or respectively* $\sigma_3 = \varepsilon_6 \varepsilon_1 t_2 \varepsilon_3 t_4$, *the transition* t_5 *are enabled.* σ_1, σ_2 *and* σ_3 *are* paths *of* t_5 *at* M_0. *However, only* σ_1 *and* σ_2 *are* minimal paths, *because* $\pi(\sigma_3) > \pi(\sigma_1)$. *The p-vector of* σ_1 *is* $\overrightarrow{v_1} = [1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$ *and The p-vector of* σ_2 *is* $\overrightarrow{v_2} = [0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1]^T$.

Definition 29 *For a minimal path* σ *of a transition* t *at marking* M, *the firing sequence of events of the minimal path is* $\mathcal{L}(\sigma)$.

Example 38 *For the minimal path* σ_1 *in Example 37, the firing sequence of events of the minimal path is* $\mathcal{L}(\sigma_1) = ab$.

Based on the notion of minimal path, the principle idea of the on-the-fly approach using T-invariants is as follows:

– Step 1: For a given normal BFS (the initial BFS is $BFS_0 = \{BFM_0\}$) and for $BFM \in BFS$, compute a minimal path of a chosen fault transition t_f at $mark(BFM)$. Build the BFG and BFST by following the firing sequence of events ω_1 of the minimal path (Algorithm 5). A minimal path of t_f is computed by Algorithm 8 (MODE=F). The lastly obtained BFS is BFS'. After that, the next BFS is F-uncertain (since t_f is enabled but is not certainly fired, at least one normal BFM and one faulty BFM can be obtained).

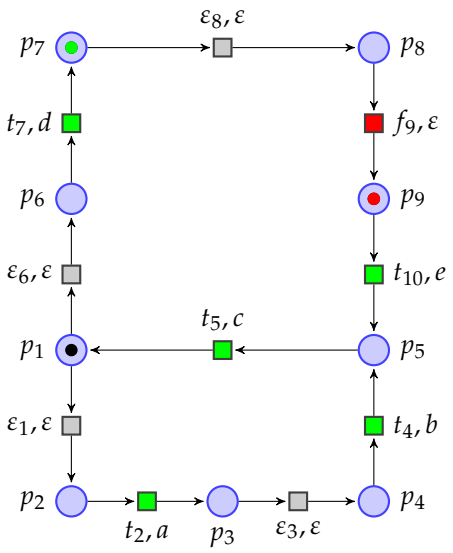


Figure 3.39 – Distribution of tokens after observing label "d" from the initial marking of the LPN in Figure 3.17

Example 39 *Let us consider the LPN in Figure 3.17 (page 40). The initial BFM is* $BFM_0 = [2\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \mid 0]^\tau$. *A minimal path to firing the fault transition is* $\sigma_1 = \varepsilon_6 t_7 \varepsilon_8$, *computed by*

Algorithm 8 (MODE=F). The sequence of events of the minimal path is $\omega_1 = d$. After observing "d" from BFM_0, two BFMs can be obtained: $BFM_1 = [1\,0\,0\,0\,0\,0\,1\,0\,0 \mid 0]^\tau$ by firing the transition t_7 and its minimal explanation ε_6 from BFM_0; and $BFM_2 = [1\,0\,0\,0\,0\,0\,0\,0\,1 \mid 1]^\tau$ by firing the fault transition f_9 and its minimal explanation ε_8 from BFM_1. An ambiguity is generated because the fault transition is enabled and can be fired or not. As it is shown in Figure 3.39, after observing label "d", there is one token in p_1 (black one) but the other token can be in p_7 (green one) or in p_9 (red one).

It is worth noticing that after observing "d", BFM_1 and BFM_2 are all built in the BFG (see Figure 3.41). However, in the BFST (see Figure 3.42), the BFS after observing "d" contains only BFM_1, because BFM_2 is obtained by firing a fault transition (explained in Algorithm 1 and Algorithm 2). The obtained BFS remains normal, but the next BFS can be F-uncertain. For example, the label "a" is observed after "d", two BFMs can be obtained: $BFM_3 = [0\,0\,1\,0\,0\,0\,1\,0\,0 \mid 0]^\tau$ by firing the transition t_2 and its minimal explanation ε_1 from BFM_1; $BFM_4 = [0\,0\,1\,0\,0\,0\,0\,0\,1 \mid 1]^\tau$ by firing the transition t_2 and its minimal explanation ε_1 from BFM_2. Therefore, the obtained BFS that contains BFM_3 and BFM_4, is F-uncertain.

– Step 2: From the obtained BFS', at a BFM in BFS', compute a firing sequence of events ω_2 of a minimal path of a transition labeled by an observable event in a T-invariant $\overrightarrow{\Omega}_N \in \mathcal{I}_N$ (Algorithm 6). Noticing that the tokens that are used to enable the fault transition, will not be used to compute the minimal path (line 9 of Algorithm 8). Build the BFG and BFST by following the firing sequence of events.

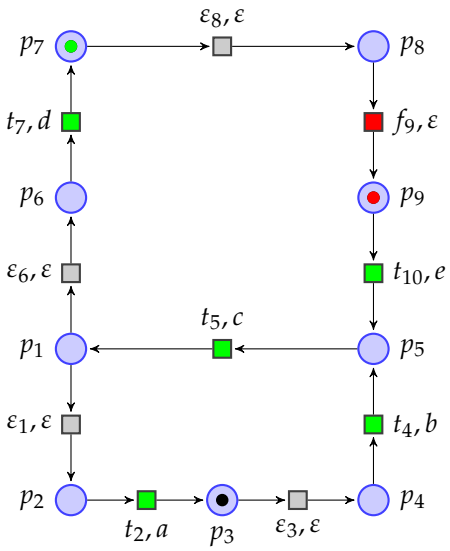


Figure 3.40 – Distribution of tokens after observing sequence "*da*" from the initial marking of the LPN in Figure 3.17

Example 40 *As presented in Example 39, after observing "d", the obtained BFS contains BFM_1. The Step 2 is used to enable an observable transition of a normal minimal T-invariant. The token that*

used to enable the fault transition to generate the ambiguity, will not be used to generate the minimal path of the observable transition. The set of normal T-invariants is $\mathcal{I}_N = \{\overrightarrow{\Omega}_1 = [1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0]^\tau\}$. *If* t_4 *is chosen to be enabled, the minimal path of* t_4 *at* BFM_1 *is* $\sigma_2 = \varepsilon_1 t_2 \varepsilon_3$. *The firing sequence of the minimal path is* $\mathcal{L}(\sigma_2) = a$. *After observing "a" the obtained BFMs and BFS have given in Example 39. The distribution of tokens (shown in Figure 3.40) is that there is one token in* p_3 *(black one) but the other token can be in* p_7 *(green one) or in* p_9 *(red one).*

– Step 3: Afterwards, from the lastly obtained F-uncertain BFS'', the Algorithm 7 is used to find if there exists an indeterminate cycle by firing a sequence ω_3 where $\omega_3 \in \mathcal{S}(\overrightarrow{\Omega}_N)$.

Example 41 *After Step 1 and Step 2, the ambiguity is carried by the token that enables the fault transition. The other token is used to find if an imply trace of the normal T-invariant can be fired or not. From Figure 3.40, the token in* p_3 *(black one) can be used to firing an imply trace of the normal T-invariant* $\overrightarrow{\Omega}_1$, *which is "bca". Therefore, an indeterminate cycle is founded by building the BFG (see Figure 3.41) and BFST (see Figure 3.42) following the sequence bca from the third BFS in BFST (Figure 3.42) that contains* BFM_3 *and* BFM_4.

Algorithm 5 Algorithm for α_1 function: generate the sequence of events to find an F-uncertain BFS

Input: a BFS, a fault trantion t_f and the set of minimal T-invariants without fault transitions $\mathcal{I}_N$;
Output: a sequence of observable events ω_1;
function $\alpha_1(BFS, t_f, \mathcal{I}_N)$
  if $\mathcal{I}_N \neq \emptyset$ **then**
    for all $BFM \in BFS$ **do**
      $\sigma_1 \leftarrow P(BFM, t_f, t_f, F)$; ▷ P is in Algorithm 8, $MODE = F$.
      if $\sigma_1 \neq \emptyset$ **then**
        return $\omega_1 \leftarrow \mathcal{L}(\sigma_1)$; **end if**; **end for**; **end if**;

Algorithm 6 Algorithm for α_2 function: generate a firng sequence of events to enable an observable transition in a T-invariant

Input: a BFS, a fault trantion t_f and the set of minimal T-invariants without fault transitions $\mathcal{I}_N$;
Output: a sequence of events ω_2;
function $\alpha_2(BFS, t_f, \mathcal{I}_N)$
  for all $BFM \in BFS$ **do**
    for all $\overrightarrow{\Omega} \in \mathcal{I}_N$ **do**
      for all $t \in T_o$ s.t $\overrightarrow{\Omega}(t) > 0$ **do**
        $\sigma_2 \leftarrow P(BFM, t_f, t, T)$; ▷ P is in Algorithm 8, $MODE = T$.
        if $\sigma_2 \neq \emptyset$ **then**
          return $\omega_2 \leftarrow \mathcal{L}(\sigma_2)$; **end if**; **end for**; **end for**; **end for**;

Algorithm 7 Algorithm for α_3 function: generate an imply trace of a T-invariant in $\mathcal{I}_N$

1: Input: a BFS and a minimal T-invariant $\overrightarrow{\Omega} \in \mathcal{I}_N$;
2: Output: a sequence of events ω_3;
3: **function** $\alpha_3(BFS, \overrightarrow{\Omega})$
4: **for all** $BFM \in BFS$ **do**
5: $\sigma_3 \leftarrow$FiringSeq$(BFM, \lambda, T_M(\overrightarrow{\Omega}))$; ▷ The function FiringSeq is in Algorithm 9.
6: **if** $\sigma_3 \neq \lambda$ **then**
7: **return** $\omega_3 \leftarrow \mathcal{L}(\sigma_3)$; **end if**; **end for**;

To compute a minimal path to enable a selected transition (a fault transition or an observable transition in a T-invariant), the Algorithm 8 is developed. This algorithm is inspired by the procedure proposed in [MS82] for the computation of minimal P-invariant and the procedure proposed in [GS05] for the computation of minimal explanation. The algorithm is used for two purposes: to compute a minimal path to enable a fault transition ($MODE = F$, called in Algorithm 5); to compute a minimal path to enable an observable transition belonging to a T-invariant ($MODE = T$, called in Algorithm 6). The difference between two modes is the initialization of the vector A and the matrix $\widetilde{C}$ (line 5-7 and 8-10 of Algorithm 8). It is worth noticing that, for $MODE = T$, $A \leftarrow (Mark(BFM) - Pre(\cdot, t_f) - Pre(\cdot, t))^\tau$. It means that the token(s), that is (are) used to enable the chosen fault transition t_f, will not be used to generate the path to enable t. In other words, after firing the sequence of the obtained path, there exists an obtained BFM that can enable t and t_f at the same time.

The Algorithm 9 is called in function P (line 25 of Algorithm 8). This algorithm is applied to get a firing sequence of transitions form a given firing vector.

The Example 42 is given to illustrate the algorithm of the function P. In this example, for the LPN in Figure 3.17 (page 40), the algorithm is applied to find a minimal path from the initial marking BFM_0 to enable the fault transition f_9.

Example 42 *Let us consider the LPN in Figure 3.17 (page 40). Let* $BFS = \{BFM_0 = [2\,0\,0\,0\,0\,0\,0\,0\,0 \mid 0]^\tau\}$ *and* $t_f = f_9$. *Then* $C_{/f_9}$ *is obtained by removing the column of* f_9.

$$\widetilde{C}^\tau = C^\tau_{/f_9} = \left|\begin{array}{c|ccccccccc} & P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 \\ \varepsilon_1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ \varepsilon_3 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ t_5 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ \varepsilon_6 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ t_7 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ \varepsilon_8 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ t_{10} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{array}\right|$$

Algorithm 8 Algorithm for P function

1: Input: a BFM, a fault transition t_f, a transition t to be enabled and a enumerated variable $MODE = \{F, T\}$ to indicate the using mode of the function, where F (short for $FAULT$) indicates that the function is used to find a minimal path to enable a fault transition and T (short for $T - INVARIANT$) indicates that the function is used to find a minimal path to enable a transition in a T-invariant;
2: Output: a firing sequence of transitions σ;
3: **function** $P(Mark(BFM), t_f, t, MODE)$
4: $\mathcal{U} \leftarrow \emptyset$;
5: **if** $MODE = F$ **then**
6: $A \leftarrow (Mark(BFM) - Pre(\cdot, t_f))^\tau$;
7: $\widetilde{C} \leftarrow C_{/t_f}$; ▷ $C_{/t_f}$ is the matrix obtained by removing the column t_f of the incidence matrix C;
8: **else if** $MODE = T$ **then**
9: $A \leftarrow (Mark(BFM) - Pre(\cdot, t_f) - Pre(\cdot, t))^\tau$;
10: $\widetilde{C} \leftarrow C_{/t}$; **end if**; ▷ $C_{/t}$ is the matrix obtained by removing the column t of the incidence matrix C;
11: Let $\Gamma = \left.\begin{array}{c|c} \widetilde{C}^\tau & I_{(n-1)\times(n-1)} \\ \hline A & B \end{array}\right.$
12: where $B := \overrightarrow{0}^\tau_{n-1}$;
13: **while** A has negative integers **do**
14: choose an element $A(i^*, j^*) < 0$;
15: let $\mathcal{H}^+ = \{i | \widetilde{C}^\tau(i, j^*) > 0\}$;
16: **for all** $i \in \mathcal{H}^+$ **do**
17: Add to $[A|B]$ a new row $[A(i^*, \cdot) + \widetilde{C}^\tau(i, \cdot) | B(i^*, \cdot) + \overrightarrow{n}^\tau_i]$,
18: where $\overrightarrow{n}_i$ is the i-th canonical basis vector; **end for**;
19: Remove the row $[A(i^*, \cdot)|B(i^*, \cdot)]$; **end while**;
20: Remove from B any row that covers other rows and any row containing a fault transtion;
21: **for all** $\overrightarrow{k}$ is a row of B **do**
22: $\mathcal{U} \leftarrow \mathcal{U} \cup \overrightarrow{k}^\tau$; **end for**;
23: **for all** $\overrightarrow{y} \in \mathcal{U}$ **do**
24: $\sigma \leftarrow \text{FiringSeq}(BFM, \lambda, T_M(\overrightarrow{y}))$; ▷ The function FiringSeq is in Algorithm 9.
25: **if** $\sigma \neq \emptyset$ **then**
26: **return** σ; **end if**; **end for**;

$Pre(\cdot, f_9) = [0\,0\,0\,0\,0\,0\,0\,1\,0]^\tau$, so $A = (Mark(BFM_0) - Pre(\cdot, f_9))^\tau = [2\,0\,0\,0\,0\,0\,0\,-1\,0]$. $B = [0\,0\,0\,0\,0\,0\,0\,0\,0]$ then $\Gamma = \left.\begin{array}{c|c} \widetilde{C}^\tau & I_{(n-1)\times(n-1)} \\ \hline A & B \end{array}\right.$. There is a negative element of A, namely $A(1,8)$. It can be observed that $\mathcal{H}^+ = \{8\}$ (line 14-15 of Algorithm 8). By using the Algorithm 8 (line 17 to 18), the new row $[2\,0\,0\,0\,0\,0\,-1\,0\,0|\,0\,0\,0\,0\,0\,0\,0\,1\,0]$ is added to Γ and the row $\Gamma(10, \cdot)$ is removed. Repeat the steps above until that there does not exist negative element of the new A (line 13-20 of Algorithm 8). Finally, the last new row that is added to Γ, is $[1\,0\,0\,0\,0\,0\,0\,0\,0|\,0\,0\,0\,0\,0\,1\,1\,1\,0]$. The vector in $\mathcal{U}$ is $[0\,0\,0\,0\,0\,1\,1\,1\,0]$. It means that, in order to enable f_9, it needs to fire ε_6, t_7 and ε_8. By using the Algorithm 9, at BFM_0 only ε_6 is enabled (line

6-7 of Algorithm 9). The reached marking BFM' is calculated. At BFM', t_7 is enabled an so on (repeat line 6-16 of Algorithm 9). The obtained firing sequence of transitions is $\sigma = \varepsilon_6 t_7 \varepsilon_8$ that is a minimal path (without any fault transition) to enable f_9.

Algorithm 9 Algorithm for FiringSeq function: generate a firing sequence of transitions at a BFM from a firing vector

1: Input: a BFM, a sequence of transition σ and $T(\overrightarrow{y})$ the multiset of transitions corresponding to the firing vector $\overrightarrow{y}$;
2: Output: a firing sequence of transition σ';
3: **function** $FiringSeq(BFM, \sigma, T_M(\overrightarrow{y}))$
4: $T_{enabled} \leftarrow \emptyset$;
5: **for all** $t \in T_M(\overrightarrow{y})$ **do** ▷ $T_M(\overrightarrow{y})$ is the multiset of transitions in the firing vector $\overrightarrow{y}$.
6: **if** t is enabled at BFM **then**
7: $T_{enabled} \leftarrow T_{enabled} \cup t$; **end if**; **end for**;
8: **if** $T_{enabled} \neq \emptyset$ **then**
9: **for all** $t \in T_{enabled}$ **do**
10: $mark(BFM') \leftarrow mark(BFM) + C(\cdot, t)$;
11: $\sigma' \leftarrow \sigma t$;
12: $T'_M(\overrightarrow{y}) \leftarrow T_M(\overrightarrow{y})/t$;
13: **if** $T'_M(\overrightarrow{y}) = \emptyset$ **then**
14: **return** σ';
15: **else**
16: FiringSeq$(BFM', \sigma', T'_M(\overrightarrow{y}))$; **end if**; **end for**; **end if**;
17: **return** λ;

Example 43 *For the LPN in Figure 3.17, there are two minimal T-invariants and $\mathcal{I}_N = \{\overrightarrow{\Omega}_1 = [1\,1\,1\,1\,1\,0\;\;0\,0\,0\,0]^\tau\}$. First, the Algorithm 5 is applied to generate a firing sequence of events to enable the fault transition f_9. Using Algorithm 8 ($MODE = F$), it is found out that the firing sequence of transitions $\varepsilon_6 t_7 \varepsilon_8$ needs to be fired in order to enable f_9 (Example 42), so the firing sequence of events of the path is $\mathcal{L}(\varepsilon_6 t_7 \varepsilon_8) = d$. After that, it is possible to obtain an F-uncertain BFS. The Algorithm 1 and the Algorithm 2 are used to build the nodes and arcs in BFG and BFST. Every arc in BFG is labeled by the firing transition, its label and its minimal e-vector (e.g. $d(t_7)$, $\overrightarrow{e_3}$). Afterwards, the Algorithm 6 is applied to generate a firing sequence of events to enable an observable transition in a T-invariant in $\mathcal{I}_N$. For example, if the transition t_4 is chosen to be fired, by using the Algorithm 8 ($MODE = T$), it is found out that the firing sequence of transitions $\varepsilon_1 t_2 \varepsilon_3$ needs to be fired in order to enable t_4. Thus, the next firing sequence of events is $\mathcal{L}(\varepsilon_1 t_2 \varepsilon_3) = a$. However, according to the Algorithm 1, the fault transition f_9 is processed observable and the node in BFG (BFM_2) is built in BFG by firing f_9 before the firing of t_2. After that, the nodes obtained by firing t_2 labeled by "a" (BFM_3 and BFM_4) are built. The obtained BFS ($BFS_2 = \{BFM_3, BFM_4\}$, BFM_2 is not in BFS_2 because it is obtained by firing a fault transition) is F-uncertain. Then, using Algorithm 7, the firing sequence of $\overrightarrow{\Omega}_1$ at BFS_2 is $\varepsilon_3 t_4 t_5 \varepsilon_1 t_2$. Therefore, the firable imply trace of $\overrightarrow{\Omega}_1$ is $\omega = \mathcal{L}(\varepsilon_3 t_4 t_5 \varepsilon_1 t_2) = bca$ is found. The output of Algorithm 7 is "bca". After the*

construction of BFG (Figure 3.41) and BFST (Figure 3.42) by firing the sequence of events "bca", it is found out that there exists an indeterminate cycle, so the system is non-diagnosable.

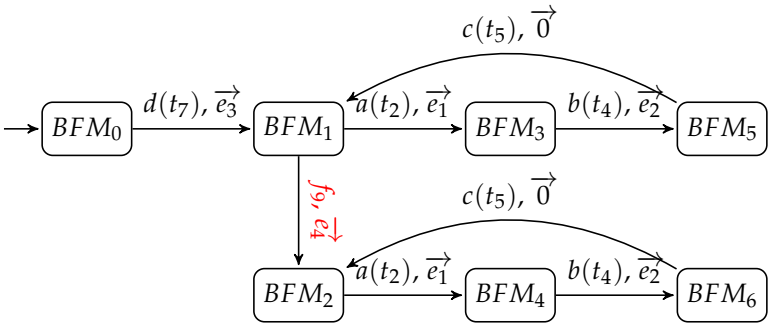


Figure 3.41 – BFG using T-invariants of LPN shown in Figure 3.17

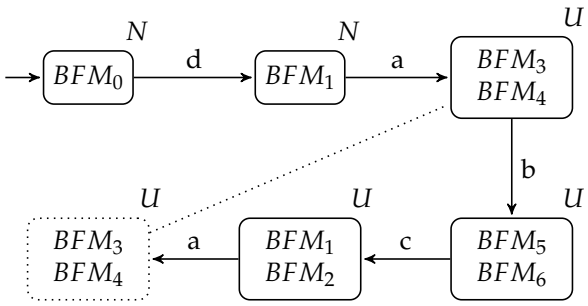


Figure 3.42 – BFST using T-invariants of LPN shown in Figure 3.17

Table 3.7 – BFMs and e-vectors of the BFG and BFST (Figure 3.41 and Figure 3.42)

j	BFM_j	j	$\overrightarrow{e_j}$
0	$[2\,0\,0\,0\,0\,0\,0\,0\,0 \mid 0]^\tau$	1	$[1\,0\,0\,0]$
1	$[1\,0\,0\,0\,0\,0\,1\,0\,0 \mid 0]^\tau$	2	$[0\,1\,0\,0]$
2	$[1\,0\,0\,0\,0\,0\,0\,0\,1 \mid 1]^\tau$	3	$[0\,0\,1\,0]$
3	$[0\,0\,1\,0\,0\,0\,1\,0\,0 \mid 0]^\tau$	4	$[0\,0\,0\,1]$
4	$[0\,0\,1\,0\,0\,0\,0\,0\,1 \mid 1]^\tau$		
5	$[0\,0\,0\,0\,1\,0\,1\,0\,0 \mid 0]^\tau$		
6	$[0\,0\,0\,0\,1\,0\,0\,0\,1 \mid 1]^\tau$		

From the example, the priorities in the investigation of branches are defined by using T-invariants. For a non-diagnosable LPN, an existent indeterminate cycle is found quickly so that fewer nodes are generated before finding the indeterminate cycle. The result of on-the-fly diagnosability analysis does not totally depend on the LPN model. The construction of the BFG and BFST (FM-graph and FM-set tree) is oriented to find quickly the existing indeterminate cycle and the combinatorial explosion problem is reduced. By using T-invariant, only the order of investigating the branches is changed, so the complexity is not reduced.

3.2.5 On-the-fly diagnosability analysis using VN

In this section, a new approach is developed for the on-the-fly diagnosability analysis using VN. It is proposed to reduce the combinatorial explosion of the VN approach. A new algorithm, which is based on the depth-first search, is given for the on-the-fly construction and analysis of the VN and its reachability graph/coverability graph (RG/CG).

In Section 3.1.2.4, we recalled the Verifier Net (VN) approach in [Cab+12], which was proposed for both *bounded* and *unbounded* LPN. For a *bounded* LPN, the complexity of this approach is linear in the sum of the number of states and transitions of the reachability graph (RG) of the VN. The problem is the combinatorial explosion, because of the scale of VN and its RG is usually large. In [Cab+12], the author said that this approach is not favorable for diagnosability analysis of *bounded* LPN because it is less efficient than the MBRG/BRD approach in [Cab+14].

For a given LPN model, the LPN $LPN' = (N', M'_0, \Sigma', \mathcal{L}')$ associated with the T'-induced subnet (Definition 19) of the considered LPN system $LPN = (N, M_0, \Sigma, \mathcal{L})$ is built as presented in Section 3.1.2.4. (In order to save the memory, there is no need to build exactly the PN N'. Only the Pre' and $Post'$ are useful for the following algorithm.)

In order to distinguish from the symbols of VN in Section 3.1.2.4, the partial VN is denoted as $\widehat{LPN} = (\widehat{N}, \widehat{M}_o, \widehat{\Sigma}_o, \widehat{\mathcal{L}})$, where $\widehat{N} = (\widehat{P}, \widehat{T}, \widehat{Pre}, \widehat{Post})$. The set of places is $\widehat{P} = P \cup P'$ and the initial marking is $\widehat{M}_0 = \begin{bmatrix} M'_0 \\ M_0 \end{bmatrix}$. The transitions of the VN $\widehat{T}$ are built on-the-fly.

Proposition 7 *[Li+16b] At a marking $\widehat{M} = \begin{bmatrix} M' \\ M \end{bmatrix}$ of the VN, a transition $\widehat{t}$ is enabled iff one of the following conditions is satisfied*

1. *$\widehat{t} = (\lambda, t_f)$, the transition $t_f \in T_f$ is enabled in N at the marking M;*
2. *$\widehat{t} = (\lambda, t_{reg})$, the transition $t_{reg} \in T_{reg}$ is enabled in N at the marking M;*
3. *$\widehat{t} = (t'_{reg}, \lambda)$, the transition $t'_{reg} \in T'_{reg}$ is enabled in N' at the marking M';*
4. *$\widehat{t} = (t'_o, t_o)$, the transition $t'_o \in T'_o$ is enabled in N' at the marking M' and the transition $t_o \in T_o$ is enabled in N at the marking M. Meanwhile, $\mathcal{L}'(t'_o) = \mathcal{L}(t_o)$;*

Proof: (*If*) Given a transition $\widehat{t} = (t', t)$ (one of t' and t can be λ). According the construction of transitions in VN [Cab+12], $\widehat{Pre}(\cdot, \widehat{t}) = \begin{bmatrix} Pre'(\cdot, t') \\ Pre(\cdot, t) \end{bmatrix}$.

For the conditions (1), $\widehat{t} = (\lambda, t_f)$, $t_f \in T_f$ and t_f is enabled in N at the marking M i.e., $M \geq Pre(\cdot, t_f)$. Therefore, $\begin{bmatrix} M' \\ M \end{bmatrix} \geq \begin{bmatrix} Pre'(\cdot, \lambda) \\ Pre(\cdot, t_f) \end{bmatrix}$ (where $Pre'(\cdot, \lambda) = \overrightarrow{0}$), i.e., $\widehat{M} \geq \widehat{Pre}(\cdot, \widehat{t})$.

The transition $\widehat{t} = (\lambda, t_f)$ is enabled at $\widehat{M}$. The conditions (2) and (3) can be proved in the same way.

For the conditions (4), $\widehat{t} = (t'_o, t_o)$, the transition $t'_o \in T'_o$ is enabled in N' at the marking M' and the transition $t_o \in T_o$ is enabled in N at the marking M, i.e., $M' \geq Pre'(\cdot, t'_0)$ and $M \geq Pre(\cdot, t_0)$. Therefore, $\begin{bmatrix} M' \\ M \end{bmatrix} \geq \begin{bmatrix} Pre'(\cdot, t'_o) \\ Pre(\cdot, t_o) \end{bmatrix}$, i.e., $\widehat{M} \geq \widehat{Pre}(\cdot, \widehat{t})$. The transition $\widehat{t} = (t'_o, t_o)$ is enabled at $\widehat{M}$.

(*Only if*) Assuming that, in VN, a transition $\widehat{t} = (t', t)$ (one of t' and t can be λ) is enabled. $\widehat{M} \geq \widehat{Pre}(\cdot, \widehat{t})$ i.e., $\begin{bmatrix} M' \\ M \end{bmatrix} \geq \begin{bmatrix} Pre'(\cdot, t') \\ Pre(\cdot, t) \end{bmatrix}$. It can be deduced that $M' \geq Pre'(\cdot, t')$ and $M \geq Pre(\cdot, t)$. Therefore, the transition $t' \in T'$ is enabled in N' at the marking M' and the transition $t \in T$ is enabled in N at the marking M. □

The Algorithm 10 is given to calculate the enabled transition at a marking $\widehat{M}$ according to the Proposition 7. This algorithm is called in Algorithm 12 for the on-the-fly construction of VN and its RG/CG.

Algorithm 10 Algorithm for *EnabledT* function: find the enabled transitions at a marking $\widehat{M}$

Input: N, N' and $\widehat{M} = \begin{bmatrix} M' \\ M \end{bmatrix}$;
Output: $(\widehat{T}_f, \widehat{T}_{reg}, \widehat{T}'_{reg}, \widehat{T}_o)$;
function $EnabledT(N, N', \widehat{M})$
    $\widehat{T}_f, \widehat{T}_{reg}, \widehat{T}'_{reg}, \widehat{T}_o \leftarrow \emptyset$;
    $T'_{con,o}, T_{con,o} \leftarrow \emptyset$; ▷ Local variables
    for all $t' \in T'$ **do**
        if t' is enabled at M' **then**
            if $t' \in T'_{reg}$ **then**
                $\widehat{T}'_{reg} \leftarrow \widehat{T}'_{reg} \cup \{(t', \lambda)\}$; **end if**;
            else
                $T'_{con,o} \leftarrow T'_{con,o} \cup \{t'\}$; **end if**; **end if**; **end for**;
    for all $t \in T$ **do**
        if t is enabled at M **then**
            if $t \in T_{reg}$ **then**
                $\widehat{T}_{reg} \leftarrow \widehat{T}_{reg} \cup \{(\lambda, t)\}$; **end if**;
            else if $t \in T_f$ **then**
                $\widehat{T}_f \leftarrow \widehat{T}_f \cup \{(\lambda, t)\}$; **end if**;
            else
                $T_{con,o} \leftarrow T_{con,o} \cup \{t\}$; **end if**; **end if**; **end for**;
    for all $t'_o \in T'_{con,o}$ and $t_o \in T_{con,o}$ s.t. $\mathcal{L}(t'_o) = \mathcal{L}'(t_o)$ **do**
        $\widehat{T}_o \leftarrow \widehat{T}_o \cup \{(t'_o, t_o)\}$; **end for**;
    return $(\widehat{T}_f, \widehat{T}_{reg}, \widehat{T}'_{reg}, \widehat{T}_o)$;

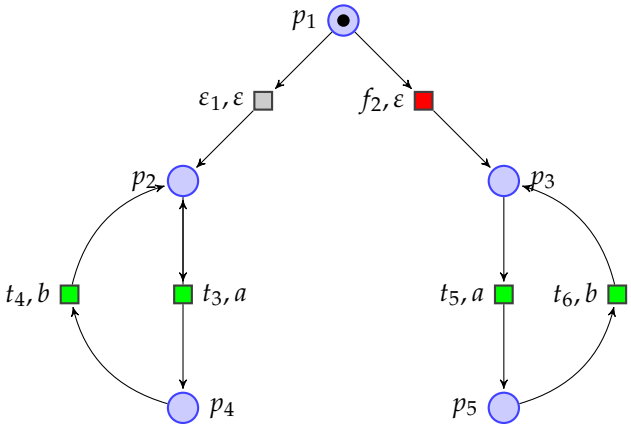


Figure 3.43 – An example of LPN

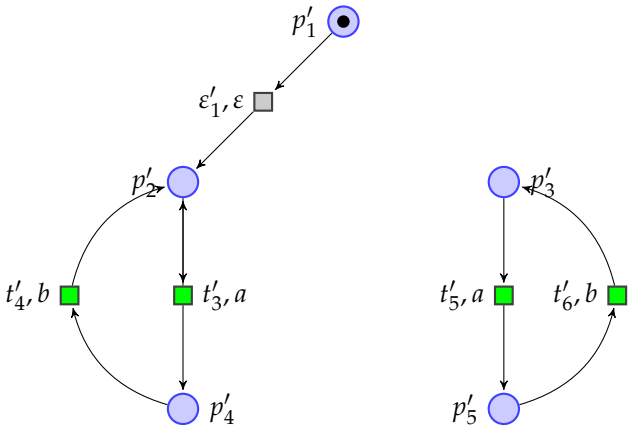


Figure 3.44 – T'-induced sub-LPN of the LPN in Figure 3.43

Example 44 *For the LPN system LPN in Figure 3.43, its T'-induced sub-LPN LPN' is built in Figure 3.44 (We repeat the LPN models in Figure 3.23 and Figure 3.24 for the convenience of readers). The initial marking of LPN is $M_0 = [1\ 0\ 0\ 0\ 0]^\tau$ and the initial marking of LPN' is $M'_0 = [1\ 0\ 0\ 0\ 0]^\tau$. At M_0, ε_1 and f_2 are enabled. At M'_0, only the transition ε'_1 is enabled. According to the Proposition 7, at the initial marking of VN $\widehat{M}_0 = \begin{bmatrix} M'_0 \\ M_0 \end{bmatrix}$, (λ, f_2) (condition (1)), (λ, ε_1) (condition (2)) and $(\varepsilon'_1, \lambda)$ (condition (3)) are enabled. Therefore, the output of the function $EnabledT(N, N', \widehat{M}_0)$ is $(\widehat{T}_f, \widehat{T}_{reg}, \widehat{T}'_{reg}, \widehat{T}_o)$, where $\widehat{T}_f = \{(\lambda, f_2)\}$, $\widehat{T}_{reg} = \{(\lambda, \varepsilon_1)\}$, $\widehat{T}'_{reg} = \{(\varepsilon'_1, \lambda)\}$ and $\widehat{T}_o = \emptyset$.*

If the entire VN is built [Cab+12], the transitions, which are never enabled, are also built such as (t'_5, t_3), (t'_5, t_5), (t'_6, t_4) and (t'_6, t_6). However, since these transitions are never enabled, they will never be generated by the Algorithm 10.

Proposition 8 *[Li+16b] The set of transitions built by using Algorithm 10, Algorithm 11 and Algorithm 12 is a subset of that of VN approach in [Cab+12], i.e. $\widehat{T} \subseteq \widetilde{T}$.*

Proof: If the remove of the fault transitions deadlocks some transitions in T-induced sub-LPN, it is certain that the transitions in $\widetilde{T}$, which are composed by the blocked transitions, will not be enabled by Algorithm 1 and will not be built by Algorithm 3. Hence, these transitions do not belong to $\widehat{T}$. In this case, $\widehat{T} \subset \widetilde{T}$. However, if the remove of the fault transitions does not deadlock any transition, it is deduced that $\widehat{T} = \widetilde{T}$. Overall, $\widehat{T} \subseteq \widetilde{T}$. □

The Algorithm 11 and the Algorithm 12 are developed for the on-the-fly diagnosability analysis of LPN systems.

Algorithm 11 Algorithm for $MainVN$ function

1: Input: LPN;
2: Output: Diagnosability analysis of the LPN;
3: **function** $MainVN(LPN)$
4: $\widehat{\mathcal{N}} \leftarrow \emptyset$; ▷ $\widehat{\mathcal{N}}$ is the set of RG/CG nodes;
5: $\widehat{\mathcal{A}} \leftarrow \emptyset$; ▷ $\widehat{\mathcal{A}}$ is the set of RG/CG arcs;
6: $F(VN) \leftarrow \emptyset$; ▷ $F(VN)$ is the set of fault nodes;
7: ▷ $\widehat{\mathcal{N}}$, $\widehat{\mathcal{A}}$ and $F(VN)$ are global variables;
8: $n \leftarrow 1$; ▷ n is the tag to indicate the diagnosability of the system;
9: build $LPN' =< N', M_0', \Sigma', \mathcal{L}' >$;
10: initialize $\widehat{LPN}$, where $\widehat{P} = P' \cup P$, $\widehat{M}_0 = \begin{bmatrix} M_0' \\ M_0 \end{bmatrix}$ and the incidence matrix $\widehat{C}$ is initially empty.
11: $\widehat{\mathcal{N}} \leftarrow \widehat{\mathcal{N}} \cup \widehat{M}_0$;
12: $(\widehat{LPN'}, \widehat{M'}, n') \leftarrow DIAG(LPN, LPN', \widehat{C}, \widehat{M_0}, n)$;
13: **if** $n' = 1$ **then**
14: assert (The LPN system LPN is diagnosable);
15: **else if** $n' = 0$ **then**
16: assert (The LPN system LPN is undiagnosable); **end if**;

Proposition 9 *For an LPN system, the DiagVN function in Algorithm 12 terminates and its diagnosability verdict is correct.*

Proof: In Algorithm 12, the $DiagVN$ function is presented, step by step, to build the VN and its RG/CG on-the-fly.

First, we prove that the algorithm terminates for both bounded and unbounded LPN. The investigation of a branch of the RG/CG is stopped, when one of the following conditions is satisfied:

1. There exists a deadlock at the new node (Line 8 of Algorithm 12);
2. A new node in RG/CG is equal to a previous one (Line 50-57 of Algorithm 12).

In an unbounded case, the unbounded places are treated in the Line 39-42 of Algorithm 12.

Algorithm 12 Algorithm for *DiagVN* function

1: Input: $(LPN, LPN', \widehat{C}, \widehat{M}, n)$;
2: Output: $(\widehat{LPN'}, \widehat{M}', n')$;
3: **function** $DiagVN(LPN, LPN', \widehat{LPN}, \widehat{M}, n)$
4: ▷ $\widehat{\mathcal{N}}$, $\widehat{\mathcal{A}}$ and $F(VN)$ are global variables defined in Algorithm 2.
5: $(\widehat{T}_{con,f}, \widehat{T}_{con,reg}, \widehat{T}'_{con,reg}, \widehat{T}_{con,o}) \leftarrow EnabledT(LPN, LPN', \widehat{M})$;
6: $\widehat{T}_{con} \leftarrow \widehat{T}_{con,f} \cup \widehat{T}_{con,reg} \cup \widehat{T}'_{con,reg} \cup \widehat{T}_{con,o}$;
7: $n' \leftarrow 1$; ▷ Local variable
8: **if** $\widehat{T}_{con} = \emptyset$ **then** ▷ There is no enabled transition at $\widehat{M}$.
9: **return** $(\widehat{LPN}, \widehat{M}, 1)$;
10: **else**
11: **for all** $\widehat{t} \in \widehat{T}_{con}$ **do**
12: **if** $(\widehat{t} = (\lambda, t_f) \in \widehat{T}_{con,f}) \wedge (\widehat{t} \notin \widehat{T})$ **then**
13: **for all** $p \in P'$ **do**
14: $\widehat{Pre}(p, \widehat{t}) = \widehat{Post}(p, \widehat{t}) = 0$; **end for**;
15: **for all** $p \in P$ **do**
16: $\widehat{Pre}(p, \widehat{t}) = Pre(p, t_f), \widehat{Post}(p, \widehat{t}) = Post(p, t_f)$; **end for**;
17: $\widehat{\mathcal{L}}(\widehat{t}) = \varepsilon$;
18: **else if** $(\widehat{t} = (\lambda, t_{reg}) \in \widehat{T}_{con,reg}) \wedge (\widehat{t} \notin \widehat{T})$ **then**
19: **for all** $p \in P'$ **do**
20: $\widehat{Pre}(p, \widehat{t}) = \widehat{Post}(p, \widehat{t}) = 0$; **end for**;
21: **for all** $p \in P$ **do**
22: $\widehat{Pre}(p, \widehat{t}) = Pre(p, t_{reg}), \widehat{Post}(p, \widehat{t}) = Post(p, t_{reg})$; **end for**;
23: $\widehat{\mathcal{L}}(\widehat{t}) = \varepsilon$;
24: **else if** $(\widehat{t} = (t'_{reg}, \lambda) \in \widehat{T}'_{con,reg}) \wedge (\widehat{t} \notin \widehat{T})$ **then**
25: **for all** $p \in P'$ **do**
26: $\widehat{Pre}(p, \widehat{t}) = Pre'(p, t'_{reg}), \widehat{Post}(p, \widehat{t}) = Post'(p, t'_{reg})$; **end for**;
27: **for all** $p \in P$ **do**
28: $\widehat{Pre}(p, \widehat{t}) = \widehat{Post}(p, \widehat{t}) = 0$; **end for**;
29: $\widehat{\mathcal{L}}(\widehat{t}) = \varepsilon$;
30: **else if** $(\widehat{t} = (t'_o, t_o) \in \widehat{T}_{con,o}) \wedge (\widehat{t} \notin \widehat{T})$ **then**
31: **for all** $p \in P'$ **do**
32: $\widehat{Pre}(p, \widehat{t}) = Pre'(p, t'_o), \widehat{Post}(p, \widehat{t}) = Post'(p, t'_o)$; **end for**;
33: **for all** $p \in P$ **do**
34: $\widehat{Pre}(p, \widehat{t}) = Pre(p, t_o), \widehat{Post}(p, \widehat{t}) = Post(p, t_o)$; **end for**;
35: $\widehat{\mathcal{L}}(\widehat{t}) = (l, l)$; **end if**; ▷ $\mathcal{L}(t_o) = \mathcal{L}'(t'_o) = l$
36: $\widehat{T} \leftarrow \widehat{T} \cup \widehat{t}$;
37: $\widehat{C}' \leftarrow$(add the column $\widehat{Post}(\cdot, \widehat{t}) - \widehat{Pre}(\cdot, \widehat{t})$ to $\widehat{C}$);
38: $\widehat{M}' \leftarrow \widehat{M} + \widehat{C}'(\cdot, \widehat{t})$;
39: Let $\widehat{M}^{\alpha}$ be the first node met on the backward path from $\widehat{M}'$ to the initial marking $\widehat{M}_0$ s.t. $\widehat{M}^{\alpha} < \widehat{M}'$; ▷ As it was presented in [KM69]
40: **if** $\widehat{M}^{\alpha}$ exists **then**
41: **for** $p \in \widehat{P}$ s.t. $\widehat{M}^{\alpha}(p) < \widehat{M}'(p)$ **do**
42: $\widehat{M}'(p) = \omega$; **end for**; **end if**;
▷ Line 39-42 are only applied for an unbounded LPN to treat the unbounded places.
43: **if** $(\forall \widehat{M}^* \in \widehat{\mathcal{N}}, \widehat{M}^* \neq \widehat{M}')$ **then**
44: **if** $(\widehat{M} \in F(VN)) \vee (\widehat{t} \in \widehat{T}_{con,f})$ **then**
45: $F(VN) \leftarrow F(VN) \cup \widehat{M}'$; **end if**;
46: $\widehat{\mathcal{N}} \leftarrow \widehat{\mathcal{N}} \cup \widehat{M}'$;
47: $\widehat{\mathcal{A}} \leftarrow \widehat{\mathcal{A}} \cup (\widehat{M}, \widehat{\mathcal{L}}(\widehat{t}), \widehat{M}')$;
48: $(\widehat{LPN''}, \widehat{M''}, n'') \leftarrow DIAG(LPN, LPN', \widehat{C}', \widehat{M}', n')$;
49: **if** $n'' = 0$ **then return** $(\widehat{LPN''}, \widehat{M''}, n'')$; **end if**;
50: **else if** $(\exists \widehat{M}^* \in \widehat{\mathcal{N}}, \widehat{M}^* = \widehat{M}')$ **then**
51: $\widehat{\mathcal{A}} \leftarrow \widehat{\mathcal{A}} \cup (\widehat{M}, \widehat{\mathcal{L}}(\widehat{t}), \widehat{M}^*)$;
52: **if** $((\widehat{M} \in F(VN)) \vee (\widehat{t} \in \widehat{T}_{con,f})) \wedge (\widehat{M}^* \notin F(VN))$ **then**
53: $F(VN) \leftarrow F(VN) \cup \widehat{M}^*$
54: **for all** $\widehat{M}^o$ can be reached from $\widehat{M}^*$ **do**
55: $F(VN) \leftarrow F(VN) \cup \widehat{M}^o$; **end for**; **end if**;
56: **if** There exists a cycle associated with a firable repetitive sequence from a node in $F(VN)$ **then** ▷ *path_exists* from the library *digraph* (Rushton, 2012).
57: **return** $(\widehat{LPN'}, \widehat{M}', 0)$; **end if**; **end if**; **end for**;
58: **return** $(\widehat{LPN'}, \widehat{M}', 1)$; **end if**; ▷ 1 denotes that *LPN* is diagnosable.

In the on-the-fly construction of the RG/CG, for any branch, these two conditions will be met sooner of later, therefore, the algorithm terminates well.

Second, when an above stop condition is satisfied, the following three cases can occur.

1. A deadlock is found;
2. The cycle is reachable starting from a node in the set $F(VN)$;
3. The cycle is not reachable starting from any node in the set $F(VN)$.

In case (1), if there exists a deadlock, the investigation of this branch is stopped. However, the diagnosability of the LPN system cannot be determined. The construction needs to be continued. The construction restarts from its previous node (backtracking from the recursive call of Line 48, Algorithm 12), and the other branches are investigated (iteration of Line 11, Algorithm 12). In case (2), if the cycle corresponds to a firable repetitive sequence, the result that the LPN system is non-diagnosable, can be obtained immediately according to the Theorem 2 (Line 56, 57 of Algorithm 12). Otherwise, it needs to investigate other branches. In case (3), the construction needs to be continued.

If the LPN system is diagnosable or a cycle is reachable starting from a node in the set $F(VN)$ at the end of the construction, all the *semi-live* and *live* transitions (Line 11-36 of Algorithm 12) of VN are built and the whole RG/CG is built.

Since all the possible cases are considered, it can be sure that Algorithm 12 terminates well and its diagnosability verdict is correct. □

The Proposition 9 guarantees the correctness of this approach. When a cycle corresponding to a firable repetitive sequence is found, and it is reachable starting from any node in the set $F(VN)$, the LPN is determined immediately as non-diagnosable.

Here, it is assumed that the transitions in $\widehat{T}_f$ have higher priorities, because, according to the Theorem 2 (page 53), the diagnosability analysis is based on finding the cycle after firing a fault transition. For the transitions in $\widehat{T}_{reg}$, $\widehat{T}'_{reg}$ and $\widehat{T}_o$, the priorities cannot be defined reasonably. To analyze the example in this paper, the priorities between the branches to be investigated are heuristically defined as follows:

$$\widehat{T}_f > \widehat{T}_{reg} > \widehat{T}'_{reg} > \widehat{T}_o$$

The priorities of the transitions in the same set are defined by numerical order of the transitions. For example, for the transition in $\widehat{T}_o$, $(t'_2, t_2) > (t'_2, t_7) > (t'_7, t_2) > (t'_7, t_7)$.

Example 45 *Let us consider again the LPN $LPN = (N, M_0, \Sigma, \mathcal{L})$ in Figure 3.43. (we study an unbounded LPN system, because a bounded LPN is a special case of unbounded LPN.) The T'-induced sub-LPN $LPN' = (N', M'_0, \Sigma', \mathcal{L}')$ is built in Figure 3.44. The places of VN are built as $\widehat{P} = P \cup P'$ and the initial marking is $\widehat{M}_0 = \begin{bmatrix} M'_0 \\ M_0 \end{bmatrix}$.*

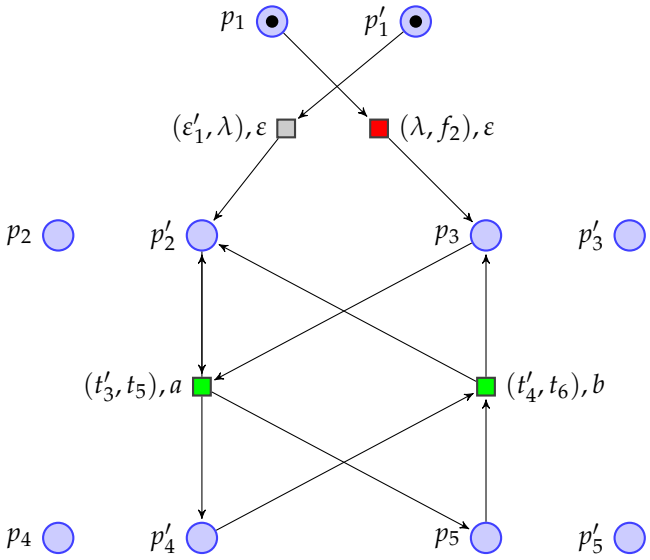


Figure 3.45 – On-the-fly construction of the VN of the LPN in Figure 3.43

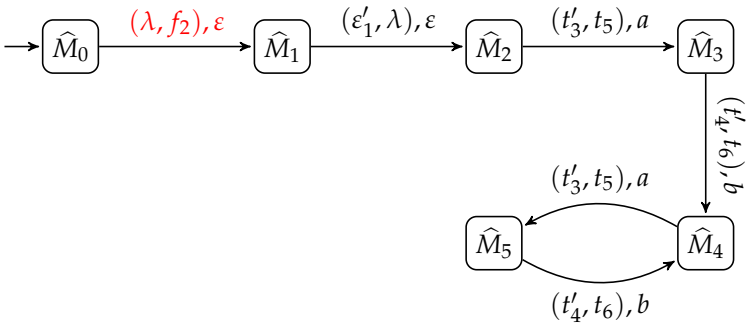


Figure 3.46 – On-the-fly construction of CG

Table 3.8 – Markings in Figure 3.46

j	$\widehat{M}_j$
0	$[1\,0\,0\,0\,0 \mid 1\,0\,0\,0\,0]^\tau$
1	$[1\,0\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
2	$[0\,1\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
3	$[0\,1\,0\,1\,0 \mid 0\,0\,0\,0\,1]^\tau$
4	$[0\,\omega\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$
5	$[0\,\omega\,0\,1\,0 \mid 0\,0\,0\,0\,1]^\tau$

At the initial marking $\widehat{M}_0 = [1\,0\,0\,0\,0 \mid 1\,0\,0\,0\,0]^\tau$, $(\lambda, \varepsilon_1), \varepsilon$, $(\varepsilon'_1, \lambda), \varepsilon$ *and* $(\lambda, f_2), \varepsilon$ *are enabled (Algorithm 10). The transition* $(\lambda, f_2), \varepsilon$ *is built in the VN (Line 12-18, Algorithm 12) and is fired because of its higher priority. The next node* $\widehat{M}_1 = [1\,0\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$ *is generated and built in the CG (Line 50-51, Algorithm 12). The set of fault nodes* $F(VN)$ *is updated (Line 49, Algorithm 12).*

At $\widehat{M}_1$ *(Line 52, Algorithm 12), only* $(\varepsilon'_1, \lambda), \varepsilon$ *is enabled. This transition is built and by firing*

this transition, the next node in CG is $\widehat{M}_2 = [0\,1\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$. *The set of fault nodes* $F(VN)$ *is updated.*

At $\widehat{M}_2$, *only* (t'_3, t_5), *a is enabled according to the Condition (4) in Proposition 7. This transition is built and the next node in CG is* $\widehat{M}_3 = [0\,1\,0\,1\,0 \mid 0\,0\,0\,0\,1]^\tau$. *The set of fault nodes* $F(VN)$ *is updated.*

At $\widehat{M}_3$, *only* (t'_4, t_6), *b is enabled. This transition is built. By firing this transition, the computed marking is* $\widehat{M}'_4 = [0\,2\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$. *Since on the backward path, the marking* $\widehat{M}_2$ *can be found s.t.* $\widehat{M}'_4 > \widehat{M}_2$. *Therefore, the number of the token in place* p'_2 *is set as* ω *(Line 43-46, Algorithm 12). The next node in CG is* $\widehat{M}_4 = [0\,\omega\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$ *and it is put into* $F(VN)$.

At $\widehat{M}_4$, *only* (t'_3, t_5), *a is enabled and* $\widehat{M}_5 = [0\,\omega\,0\,1\,0 \mid 0\,0\,0\,0\,1]^\tau$ *is built in CG. The node is put into* $F(VN)$.

At $\widehat{M}_5$, *only* (t'_4, t_6), *b is enabled. The computed node is* $[0\,\omega\,0\,0\,0 \mid 0\,0\,1\,0\,0]^\tau$, *which is equal to* $\widehat{M}_4$ *(Line 54-61, Algorithm 12). Therefore, a cycle from a node in* $F(VN)$ *is found. Since* $\widehat{C}' \cdot \overleftarrow{y} = \overleftarrow{0}$ *(where* $\widehat{C}'$ *is the updated incidence matrix;* $\overleftarrow{y}$ *is the firing count vector that contains* (t'_3, t_5) *and* (t'_4, t_6), *i.e.* $\overleftarrow{y}((t'_3, t_5)) = \overleftarrow{y}((t'_4, t_6)) = 1$ *and for all other transition* $\widehat{t} \in \widehat{T}$, $\overleftarrow{y}(\widehat{t}) = 0$*), the cycle is associated with a stationary repetitive sequence. Therefore, the construction of VN and its CG is stopped and the result is immediately given, that the unbounded LPN system is non-diagnosable (Line 57, Algorithm 12).*

A comparison of the VN approach in [Cab+12] and our approach is given for the diagnosability analysis of the LPN system in Figure 3.43. For the diagnosability analysis of this LPN system, our approach generates fewer transitions in the on-the-fly construction of VN and fewer nodes in CG. The result that the system is not diagnosable, is immediately given, when a cycle associated with a firable repetitive sequence is found which is reachable starting from a node in the set $F(VN)$.

Table 3.9 – Comparison of the VN approach in [Cab+12] and the on-the-fly diagnosability analysis using VN for the diagnosability analysis of the LPN model in Figure 3.43

Name of approaches	$\lvert\widetilde{P}\rvert/\lvert\widehat{P}\rvert$	$\lvert\widetilde{T}\rvert/\lvert\widehat{T}\rvert$	Number of nodes in CG
VN approach [Cab+12]	10	11	11
the on-the-fly diagnosability analysis using VN	10	4	6

If the remove of the fault transitions deadlocks some transitions in T'-induced sub-LPN, fewer transitions will be built in our approach comparing to the VN approach in [Cab+12], because it is presented in Proposition 8 that $\widehat{T} \subseteq \widetilde{T}$. Even for a diagnosable system, only the *semi-live* and *live* transitions are built in VN.

The on-the-fly diagnosability analysis using VN avoids building, *a priori*, the whole

VN and its RG/CG. The VN structure and its RG/CG are built on-the-fly and in parallel with stop conditions. When a cycle corresponding to a firable repetitive sequence is found which is reachable by firing a fault, there is no need to continue the construction and the LPN system is determined as non-diagnosable.

Afterwards, we analyze the complexity of the on-the-fly diagnosability analysis using VN. We analyze at first the complexity of the VN approach in [Cab+12], because it is not done in [Cab+12], in order to compare the complexity of the VN approach in [Cab+12] and that of our approach.

(1) For bounded LPN

Let us denote $n_1 = |P|$ the number of

the places of the initial LPN model LPN, $n_2 = |T|$ the number of the transitions of LPN and $n_3 = |\mathcal{N}_{RG}|$ the number of the nodes in the RG of LPN.

Theorem 5 *The complexity of VN approach in [Cab+12] is* $\mathcal{O}(n_1 n_2^2 + n_3^2 n_2^2)$.

Proof: The complexity of the VN approach in [Cab+12] contains three parts: (1) construction of VN; (2) construction of the RG of VN; (3) verification of the cycle that is reachable starting from a node in the set $F(VN)$.

(1) The number of the places in VN is $|\widetilde{P}| = |P'| + |P| = 2 \cdot n_1$. The number of the transitions in VN is $|\widetilde{T}| \leqslant |T|^2 = n_2^2$. Besides, the arcs built in VN between the places and transitions is at most $2 \cdot |\widetilde{T}| \cdot |\widetilde{P}| \leqslant 2 \cdot (n_2^2) \cdot (2 \cdot n_1)$. Therefore, the complexity of building the VN is $\mathcal{O}(n_1 n_2^2)$;

(2) $\mathcal{N}_{RG}^{VN}$ denotes the set of the nodes in RG of VN and the $\mathcal{A}_{RG}^{VN}$ denotes the set of the arcs in RG of VN. The number of $\mathcal{N}_{RG}^{VN}$ is at most n_3^2. The arcs in the RG of VN is $|\mathcal{A}_{RG}^{VN}| \leqslant |\mathcal{N}_{RG}^{VN}| \cdot |\widetilde{T}| \leqslant (n_3^2) \cdot (n_2^2)$. Therefore, the complexity of building the RG of the VN is $\mathcal{O}(n_3^2 n_2^2)$.

(3) Deciding if there is a cycle that is reachable starting from a node in the set $F(VN)$ takes $\mathcal{O}(|F(VN)| + |\mathcal{A}_{F(VN)}|)$ [Cor+90], where $\mathcal{A}_{F(VN)}$ is the set of arcs between the nodes in $F(VN)$. In the worst case (all nodes belong to $F(VN)$, $|\mathcal{A}_{F(VN)}| = |\mathcal{A}_{RG}^{VN}|$), it can be decided in $\mathcal{O}(n_3^2 n_2^2)$.

Therefore, the complexity of VN approach in [Cab+12] is $\mathcal{O}(n_1 n_2^2 + n_3^2 n_2^2)$.□

Theorem 6 *The complexity of the diagnosability analysis of LPN using Algorithms 1, 2 and 3 w.r.t a given fault class is* $\mathcal{O}(n_1 n_2^2 n_3^2 + n_3^4 n_2^4)$.

Proof: First, the complexity of Algorithm 1 is $\mathcal{O}(|\widehat{T}|) = \mathcal{O}(n_2^2)$ (Line 26 of Algorithm 10). Algorithm 10 is called in Algorithm 3 but not in any iteration. Algorithm 12 contains one

main iteration that is indicated by Line 11-59 of Algorithm 12 (in the worst case, the number of iterations is $\mathcal{O}(|\widehat{T}|)$). The complexity of building a transition of VN is $\mathcal{O}(|P|)$ (Line 12-39 of Algorithm 12). To verify the condition in Line 43, the complexity is $\mathcal{O}(|\widehat{\mathcal{N}}_{RG}|)$. The verification of a cycle from a node in $F(VN)$ (Line 56 of Algorithm 12) has a complexity as $\mathcal{O}(n_3^2 n_2^2)$. The other steps in the main iteration can be neglected, because of their lower complexity. In the worst case, the Algorithm 12 is call $|\widehat{\mathcal{N}}_{RG}|$ times (Line 48). Therefore, the entire complexity is $\mathcal{O}(|\widehat{\mathcal{N}}_{RG}| \cdot (|\widehat{T}| + |\widehat{T}| \cdot (|P| + |\widehat{\mathcal{N}}_{RG}| + n_3^2 n_2^2))) = \mathcal{O}(n_1 n_2^2 n_3^2 + n_3^4 n_2^4)$.□

(2) For unbounded LPN

In [Cab+12], the authors have explained that the complexity of the VN approach for unbounded LPN cannot be given because the complexity of the construction of the CG is still an open issue. In the worst case, the on-the-fly diagnosability analysis using VN needs to build the whole VN and its CG. Moreover, the on-the-fly diagnosability analysis using VN builds the transitions in VN and the nodes in its CG and analyze at the same time when a cycle in CG is found. Assuming that $|F(VN)|$ is the number of nodes in $F(VN)$, $\mathcal{A}_{F(VN)}$ the number of arc between these nodes and the complexity of the VN approach for unbounded LPN is $\mathcal{C}_{VN}$. Deciding if there is a cycle that is reachable starting from a node in the set $F(VN)$ takes $\mathcal{O}(|F(VN)| + |\mathcal{A}_{F(VN)}|)$. In the worst case (i.e. all the nodes belong to $F(VN)$), $\mathcal{O}(|F(VN)| + |\mathcal{A}_{F(VN)}|) \leq \mathcal{C}_{VN}$. Therefore, the complexity of the on-the-fly diagnosability analysis using VN is $(n_{cycle} + 1) \cdot \mathcal{C}_{VN}$, where n_{cycle} is the number of cycles in CG, which do not correspond to repetitive sequences.

The on-the-fly diagnosability analysis using VN is proposed to reduce the combinatorial explosion problem for the diagnosability analysis of both *bounded* and *unbounded* LPN systems. The VN and its RG/CG are built on-the-fly and in parallel with stop conditions. As soon as the counterexample of diagnosability is found, the construction and the analysis are stopped immediately. The computational complexity of this approach is slightly increased by using the on-the-fly analysis technique [Fer+92; SE05]. This approach achieves a compromise between computation efficiency and combinatorial explosion limitation.

3.3 Synthesis of the contributions (on monolithic diagnosability analysis)

In this chapter, we have reviewed the existing studies on diagnosability analysis. We have recalled the automata-based approaches and the PN-based approaches. The critical problems for diagnosability analysis are combinatorial explosion and computational complexity.

Afterward, we have proposed our contributions of diagnosability analysis using LPN model.

Section 3.2.1 proposed some reduction rules that can be applied to simplify the initial LPN model before analyzing the diagnosability. By using these rules, some regular unobservable transitions, some specific observable transitions (ELOT) and some places can be removed. We have proved that by using these rules, the diagnosability property is preserved. This approach is a strong complement for most of diagnosability analysis techniques.

Section 3.2.2 proposed a new sufficient condition for the diagnosability of a *safe* and *live* LPN. This sufficient condition supplemented the defect of the sufficient condition in [Wen+05]. We have developed a method to check this sufficient condition by using linear programing technique.

Section 3.2.3 proposed the on-the-fly diagnosability analysis using minimal explanations. By using minimal explanations, the on-the-fly diagnosability analysis is improved. The BFG/BFST is a compact version of FM-graph/FM-set tree. Fewer nodes are built for analyzing the diagnosability of an LPN model.

Section 3.2.4 proposed the on-the-fly diagnosability analysis using T-invariant. By using the T-invariants, the priorities of investigating branches is defined. For a non-diagnosable LPN, the existing indeterminate cycle can be found quickly. The combinatorial explosion problem is then reduced.

We compare the results for diagnosability analysis of the LPN model in Figure 3.17 by using different approaches. Table 3.10 shows the comparison between different approaches based on states numbers. The approaches ①-③ use neither minimal explanations nor reduction rules. The approaches ④-⑥ use minimal explanations. The approaches ⑦-⑨ use reduction rules (1)-(5). The approaches ⑩-⑫ use reduction rules (1)-(7). By using the techniques mentioned above, the combinatorial explosion problem is reduced. However, the approaches proposed in Section 3.2.1 - Section 3.2.4 do not reduce the complexity for diagnosability analysis.

Section 3.2.5 proposed the on-the-fly diagnosability analysis using Verifier Nets, which can be used for both *bounded* and *unbounded* LPN model. This approach achieves a compromise between computation efficiency and combinatorial explosion limitation. The computational complexity of this approach is slightly increased by using the on-the-fly analysis technique but it remains polynomial for diagnosability analysis of *bounded* LPN.

We have proposed these approaches in order to give a synthetic solution for diagnosability analysis of different types of LPN models. For a given LPN model, we can apply the reduction rules a priori to simplify the LPN model. The memory cost is lower by using the reduced LPN model. For a *safe* and *live* LPN, we can use the sufficient condition to check the diagnosability analysis. For a *bounded* LPN that does not deadlock after the occurrence of a fault, the on-the-fly diagnosability analysis using minimal explanations and T-invariants can be applied for diagnosability analysis (if the sufficient condition for

Table 3.10 – States numbers comparison

	Name of approaches	State space (Reachability graph)	Model for diagnosis
Without minimal explanations or reduction rules	①Diagnoser approach [Sam+95] (Reachability graph/Diagnoser)	44	33
	②On-the-fly and incremental diagnosis technique [Liu+14] (FM-graph/FM-set tree)	36	14
	③On-the-fly diagnosability analysis using T-invariants [Li+15c] (FM-graph/FM-set tree)	12	5
With minimal explanations	④ MBRG/BRD	15	14
	⑤On-the-fly diagnosability analysis using minimal explanations [Li+15b] (BFG/BFST)	12	10
	⑥On-the-fly diagnosability analysis using T-invariants and minimal explanations [Li+15a] (BFG/BFST)	7	5
With reduction rules (1)-(5)	⑦Diagnoser approach after reduction rules (1)-(5) (Reachability graph/Diagnoser)	15	14
	⑧On-the-fly diagnosability analysis using reduction rules (1)-(5) (FM-graph/FM-set tree)	12	10
	⑨On-the-fly diagnosability analysis using T-invariants after reduction rules (1)-(5) (FM-graph/FM-set tree)	7	5
With reduction rules (1)-(7)	⑩Diagnoser approach after reduction rules (1)-(7) (Reachability graph/Diagnoser)	10	10
	⑪On-the-fly diagnosability analysis using reduction rules (1)-(7) (FM-graph/FM-set tree)	7	6
	⑫On-the-fly diagnosability analysis using T-invariants after reduction rules (1)-(7) (FM-graph/FM-set tree)	5	4

the diagnosability of a *safe* and *live* LPN is not fulfilled, we can use this approach to check the diagnosability); For an *unbounded* LPN, the on-the-fly diagnosability analysis using VN can be applied for diagnosability analysis. We propose these approaches in order to give a solution for industrial use that allows iterating the diagnosability analysis for a system at design stage.

CHAPTER 4

MODULAR DIAGNOSABILITY ANALYSIS USING LPN

Contents

This chapter deals with the modular diagnosability analysis, which was formally defined in [Con+06]. The modular diagnosability is less strict than monolithic diagnosability, but it provides another solution for diagnosis in practice.

We give at first a literature review of the decentralized diagnosis, modular diagnosis and distributed diagnosis which can get the same diagnosis performance with the monolithic diagnosis. Then, we focus on the modular diagnosis, especially the modular

diagnosability analysis. We propose a new approach for modular diagnosability analysis using LPN model. The initial idea is published in [Li+17c].

4.1 Literature review of decentralized fault diagnosis, modular fault diagnosis and distributed fault diagnosis

In the previous chapter, the monolithic (or centralized) diagnosis was discussed. The main advantage of monolithic diagnosis is the conceptual simplicity. However, the main disadvantage is the computational complexity and the combinatorial explosion problem. With the growth of industrial system, plants have become spread throughout a wide physical area. In order to deal with this new configuration, the monolithic approach is no longer suitable and to overcome that, decentralized diagnosis architecture [Deb+00], modular diagnosis architecture [Con+06] and distributed diagnosis architecture [GL07] were proposed. The main objective is similar to the monolithic approach i.e., to achieve the same diagnosis performance without building the whole monolithic diagnoser. We emphasize that these architectures are not mutually exclusive, having each one its own scope of use based on different assumptions and different types of models.

This chapter introduces:

- a literature review of decentralized, modular and distributed fault diagnosis;
- the contribution of the thesis on modular diagnosability using LPN.

4.1.1 Decentralized diagnosis

In [Deb+00], the pioneer work of decentralized diagnosis is proposed in the framework of automata. The main objective is to develop a fault diagnosis method without building the global diagnoser, which causes the combinatorial explosion problem. The architecture of decentralized approaches is shown in Figure 4.1. The system is partitioned into several sites. Each site has a full knowledge of the global model but has only local observation of the system. A local diagnoser is built, for each site, based on the local observation of the whole system. Local diagnosers cannot communicate directly with each other, but they provide their diagnostic decisions to a coordinator, using a communication protocol. The coordinator provides the global final diagnostic decision.

In the framework of [Deb+00], the global model is observed by two sites. Each site has its own observation module and its diagnosis module. The site i, $i \in \{1,2\}$ has its local observation which is a subset of the set of the global observable events Σ_o. Assuming that Σ_{oi} is the set of observable events of site i. Note that Σ_{o1} and Σ_{o2} are not necessarily disjoint, but they need to cover Σ_o, i.e., $\cup_{i=1}^{\{1,2\}}\Sigma_{oi} = \Sigma_o$. The projection $P_i : \Sigma^* \rightarrow (\Sigma_{oi})^*$ is defined on the set of observable events Σ_{oi}. The two sites generate their own diagnosis information and communicate via the coordinator. The way to communicate is defined by

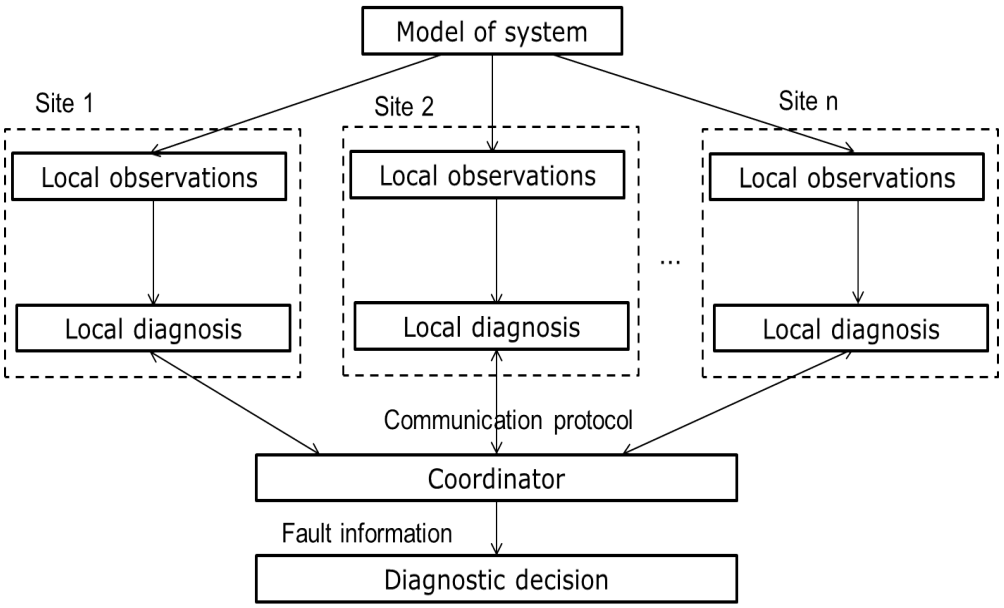


Figure 4.1 – Architecture of decentralized diagnosis approaches [Deb+00]

the protocol which is the key point for decentralized approaches. The protocol defines: the rules of communication between the local sites and the coordinator; the decision rules applied at the coordinator.

The decentralized diagnosis approaches are studied under the following assumptions:

1. The system is live;
2. The system has no cycle of unobservable events w.r.t. either site 1 or site 2;
3. The system is not diagnosable w.r.t. P_i, $i \in \{1,2\}$ and the partition of fault classes Π_f (If the system is diagnosable (as in a monolithic case) w.r.t. one of the projections and the fault partition, the system would still be diagnosable by using the decentralized architecture.);
4. The communication between local sites and the coordinator is reliable and the communicated messages are received in the same order of the sent messages;
5. Each site knows the set of observable events of each site.

Definition 30 *The diagnostic information of the coordinator C is said to be F_i – certain if the coordinator is certain that a fault in class F_i has occurred.*

Similarly, if the coordinator is certain that no fault has occurred, the diagnostic information is said to be *Normal*. Otherwise, it is said to be F_i – *uncertain* w.r.t. the fault class F_i.

Based on this notion, the diagnosability under a given protocol can be defined as follows:

Definition 31 *A prefix-closed and live language L is diagnosable under a protocol w.r.t. a set of projection and a fault partition, if the following condition holds:*

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})(\forall s \in \Psi(\Sigma_{fi}))(\forall t \in L/s)(|t| \geq n_i \Rightarrow C \text{ is } F_i - certain)$$

where $\Psi(\Sigma_{fi})$ *is the set of sequences that end with a fault in* Σ_{fi}.

The diagnosability, under a given protocol, is achieved by the coordinator. By using the given protocol, the coordinator is capable to detect any fault and its class, within a finite delay after the occurrence of the fault. The protocol is the key point of the decentralized approaches, which defines the communication rules and the decision rules. In order to illustrate the functionality of the protocol, let us consider a simple protocol called "naive protocol" as follows:

When an event $e \in \Sigma_i, i \in \{1, 2\}$ is observed, the diagnoser of site i computes its state and communicates with the coordinator. If G_{di}, the diagnoser of site i, knows that the observed event "e" belongs equally to the set of observable events of the other sites and the new state of these sites have not been computed, then the site i sets its flag SB_i to 1 (otherwise, $SB_i = 0$) to declare that the coordinator needs to wait for the communication with the other site before giving its diagnostic decision. The coordinator stores three types of data: D_1, D_2 and SB. D_i contains the last state computed by G_{di}. SB indicates the behaviors of the coordinator. If $SB = 0$, the coordinator computes directly its new state which is the intersection of the current states of the two diagnoser i.e., $C = D_1 \cap D_2$, where C denotes the state given by the coordinator. If $SB = 1$, the coordinator needs to wait the communication of the other site. The new value of SB is computed by $SB = SB \oplus SB_i$. The initial value of D_i is the initial state of the diagnoser of site i and initially, $SB = SB_i = 0$.

Example 46 *Considering the automaton in Figure 4.2. The set of observable events is* $\Sigma_o = \{a, b, c, d, e\}$ *and the set of fault event is* $\Sigma_f = \{f\}$. *Assuming that two sites are developed to diagnose the fault. The set of events observed by site 1 is* $\Sigma_{o1} = \{a, c, d, e\}$ *and that of site 2 is* $\Sigma_{o2} = \{b, d, e\}$. $\Sigma_o = \Sigma_{o1} \cup \Sigma_{o2}$. *Note that the event b cannot be observed by site 1 and a, c cannot be observed by site 2. Based on the local observations, the local diagnosers of the two sites* G_{d1} *and* G_{d2} *are built in Figure 4.3.*

Let us assume that the sequence of observable events 'bacded' is generated by the system. The diagnostic result by using 'naive protocol' is given in Table 4.1. The initial states of site 1 and site 2 are all $1N$. *Therefore, the initial state given by the coordinator is* $D_1 \cap D_2 = \{1N\}$. *The first event "b" is only observed by the site 2. The diagnoser of site 2 updates its state as* $\{3N, 4N\}$. *The coordinator computes directly its state* $D_1 \cap D_2 = \{1N\} \cap \{3N, 4N\} = \emptyset$. *The diagnostic*

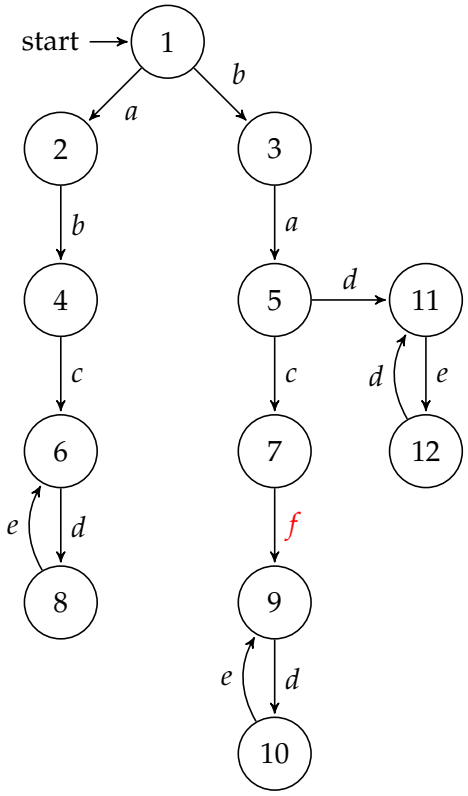
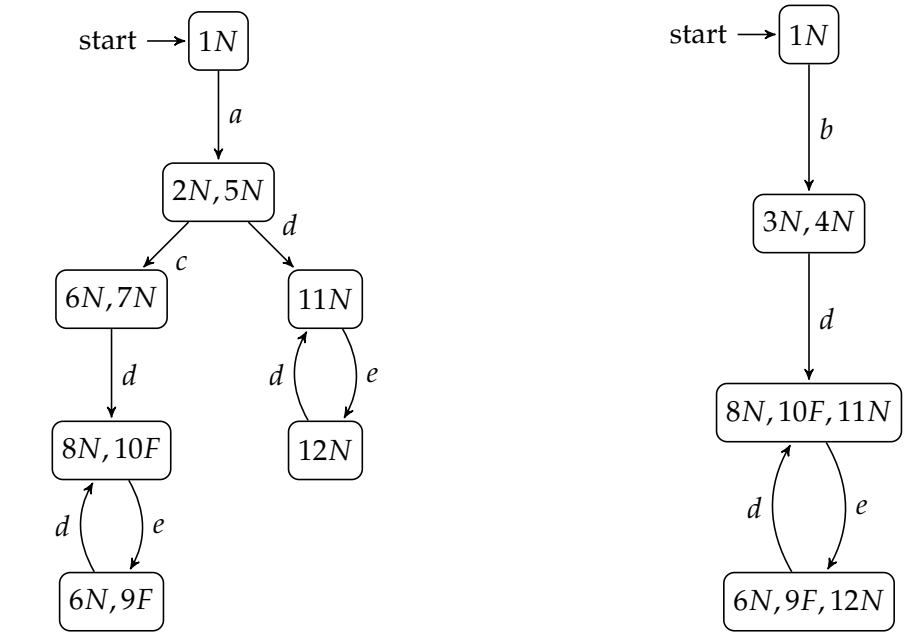


Figure 4.2 – An example of automaton

results are given in the same way after observing "a" and "c", because "a" and "c" are observed only by site 1. The coordinator gives directly the diagnostic result.

When the event "d" is observed, the diagnoser of site 1 updates its state and its flag SB_1 is set as 1 (because the site 1 knows that the event "d" can also be observed by site 2). The flag of the the coordinator is set as $SB = SB \oplus SB_1 = 0 \oplus 1 = 1$, which means that the coordinator needs to wait for communication with site 2 before giving the diagnostic result. After the communication with site 2, the diagnostic result is $D_1 \cap D_2 = \{8N, 10F\} \cap \{8N, 10F, 11N\} = \{8N, 10F\}$. The diagnostic results after the rest of events of the sequence ("e" and "d") are computed in the same way.

In Example 46, the "naive protocol" is used for fault diagnosis in order to explain the functionality of decentralized approach. The main advantage of this protocol is that the communication rules and decision rules are simple, so that the protocol is easy to implement. However, the diagnostic performance is not really accurate. In certain cases, the diagnostic result given by the decision rules is empty as it is shown in Table 4.1. Moreover, if we use a monolithic diagnoser, it can be deduced that the fault f has occurred after observing the sequence "*bacd*". It shows that a decentralized approach using a "naive protocol" does not perform as well as a monolithic approach. In [Deb+00], three protocols are developed for decentralized fault diagnosis. The fundamental issue is the tradeoff



(a) $G_{d1} \backslash \Sigma_{o1} = \{a, c, d, e\}$ (b) $G_{d2} \backslash \Sigma_{o2} = \{b, d, e\}$

Figure 4.3 – Diagnosers of local sites

Table 4.1 – Application of the naive protocol for the example in Figure 4.1

Event	Site	D_1	SB_1	D_2	SB_2	SB	C	Updated SB
(Start)	1,2	$1N$	0	$1N$	0	0	$1N$	0
b	2	$1N$	0	$3N, 4N$	0	0	$\emptyset$	0
a	1	$2N, 5N$	0	$3N, 4N$	0	0	$\emptyset$	0
c	1	$6N, 7N$	0	$3N, 4N$	0	0	$\emptyset$	0
d	1	$8N, 10F$	1	$3N, 4N$	0	0	$\emptyset$	1
	2	$8N, 10F$	0	$8N, 10F, 11N$	1	1	$8N, 10F$	0
e	2	$8N, 10F$	0	$6N, 9F, 12N$	1	0	$\emptyset$	1
	1	$6N, 9F$	1	$6N, 9F, 12N$	0	1	$6N, 9F$	0
d	1	$8N, 10F$	1	$3N, 4N$	0	0	$\emptyset$	1
	2	$8N, 10F$	0	$8N, 10F, 11N$	1	1	$8N, 10F$	0

between performance and complexity: if a protocol performs better than the others, it is certain that this protocol uses more memory and requires more computation power.

Based on the decentralized architecture, some works are proposed. In [Pen00], the decentralized approach is applied to on-line diagnosis of telecommunication networks. A local diagnoser is built for each component of the system and a strategy for coordination is proposed to minimize the on-line computation. Nevertheless, a crucial issue is the efficient update of the states by merging the local diagnostic information of all sites. In [Cab+10a], two protocols are proposed using Petri nets (PN). The notion of minimal

explanations is applied, in order to reduce the complexity for decentralized fault diagnosis. The decentralized diagnosability under the two protocols is also addressed. The advantages and disadvantages of these two protocols are similarly discussed, in terms of the tradeoff between performance and complexity. In [BS02], the authors discussed the decentralized fault diagnosis with costly communication between diagnosers. The approach supposes that the capacity of the communication channel between two diagnosers is limited. Therefore, the fault should be diagnosed by the two diagnosers with a low cost of communication.

To apply the decentralized approaches for on-line diagnosis, the codiagnosability property of the system needs to be ensured. The definition of codiagnosability is given as follows:

Definition 32 *A live and prefix-closed language $L(G)$ is codiagnosable w.r.t. P_i, $i \in I_N = \{1, 2, \cdots, N\}$ and Σ_f, iff:*

$$(\exists n \in \mathbb{N})(\forall s \in \Psi(\Sigma_f))(\forall t \in L(G)/s,\ |t| \geq n) \Rightarrow$$

$$(\exists i \in I_N)(\forall \omega \in P_i^{-1}(P_o(st)) \cap L(G))(\exists f \in \Sigma_f, f \in \omega)$$

A language $L(G)$ is codiagnosable if it is possible to detect in a finite delay any failure occurrence by at least one local diagnoser. Hence, the codiagnosability is stronger than the diagnosability property in the monolithic framework.

In [Mor+11; Mor+16; QK06; Wan+07], different approaches are proposed for codiagnosability analysis of automata based on the verifier approach. The approach in [Mor+11] has the lowest computational complexity. In [Cab+11], the decentralized diagnosability analysis is discussed in the framework of PN, by using a particular net called Modified Verifier Net (MVN): regardless of the protocol, the authors have proposed an approach by detecting the presence of particular strings called "failure ambiguous strings". A failure ambiguous string is a sequence containing some fault transitions w.r.t. a set of sites and a given fault class, such that the word of this sequence can also be explained by a non faulty sequence w.r.t. the given fault class. Assuming that the system is diagnosable in the monolithic framework, if there is no failure ambiguous strings for the considered set of sites, the system is diagnosable in the decentralized framework.

Overall, the decentralized approaches avoid building the monolithic diagnoser which could be too large for a large-scale system. The objective is to achieve the same diagnosis performance with the monolithic diagnoser, but the tradeoff between the performance and the complexity need to be considered to define the protocols used in the coordinator. Another problem of the decentralized approach is that the way of distributing the set of observable events into two local sets is not clearly defined. With different distributions, the performance may be different under the same protocol. From a practical viewpoint, the

communication between the diagnosers and the coordinator takes memory and time costs and there is a need to make sure that the order of the occurrence of events respects the order of their execution by the system. Moreover, for a large-scale system, the architecture needs to be extended to m sites. Hence, the implementation of a protocol for the coordinator is more complicated.

4.1.2 Modular diagnosis

In [Con+06], a modular approach is developed to diagnose the unobservable faults of a large and complex system modeled by parallel composition of automata. The architecture of modular diagnosis approaches is shown in Figure 4.4. Different from the decentralized approaches, the modular approaches are applied on a system physically modeled by a collection of modules. Hence, a local diagnoser is built using the model of its module, but not the global model as it is used in decentralized approach. Moreover, there does not exist a coordinator. If certain diagnosability property (e.g. modular diagnosability in [Con+06]) of system is fulfilled, the on-line diagnosis is achieved using only local diagnosers, i.e., a local diagnoser is capable to diagnose a considered fault of the corresponding module without communication with a coordinator or other diagnosers.

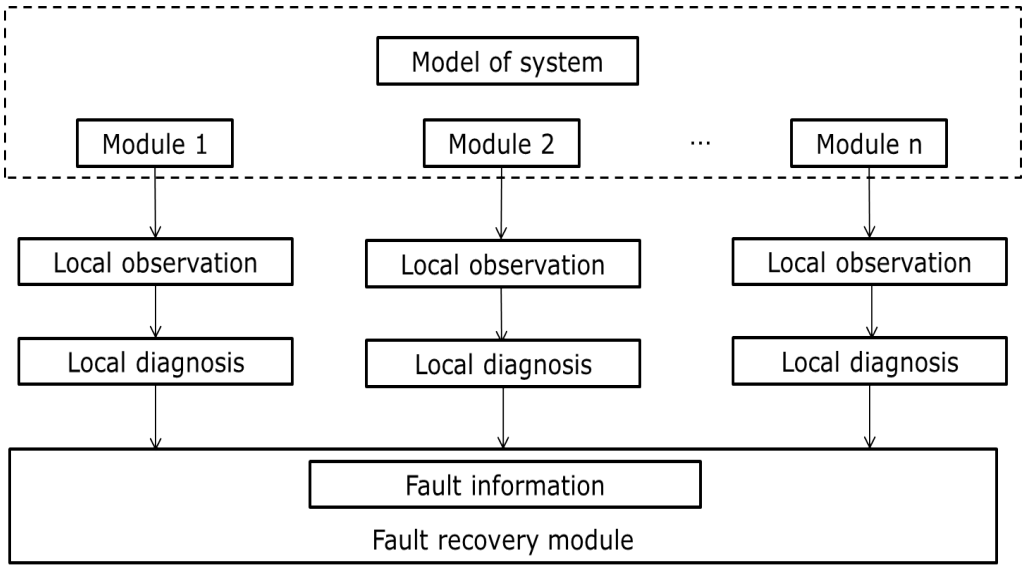


Figure 4.4 – Architecture of Modular diagnosis approaches [Con+06]

The modular diagnosability needs to be ensured so that the modular architecture can be applied for on-line diagnosis. Let us recall some basic notions in order to introduce the definition of modular diagnosability [Con+06].

Let Σ_X and Σ_Y be any sets of events. $P_{\{\Sigma_X,\Sigma_Y\}} : \Sigma_X^* \to \Sigma_Y^*$ denotes the natural projection and $P^{-1}_{\{\Sigma_Y,\Sigma_X\}} : \Sigma_Y^* \to 2^{\Sigma_X^*}$ is the inverse projection operator. $P_{\{\Sigma_X,\Sigma_Y\}}$ is defined as follows:

1. $P_{\{\Sigma_X,\Sigma_Y\}}(\varepsilon) := \varepsilon$
2. $P_{\{\Sigma_X,\Sigma_Y\}}(e) := \begin{cases} e & \text{if } e \in \Sigma_Y \\ \varepsilon & \text{if } e \notin \Sigma_Y \end{cases}$
3. $P_{\{\Sigma_X,\Sigma_Y\}}(se) := P_{\{\Sigma_X,\Sigma_Y\}}(s)P_{\{\Sigma_X,\Sigma_Y\}}(e)$ (for $s \in \Sigma_X^*, e \in \Sigma_X$.)

The inverse projection operator is denoted as $P^{-1}_{\{\Sigma_Y,\Sigma_X\}}(s) = \{t \in \Sigma_X^* | P_{\{\Sigma_X,\Sigma_Y\}}(t) = s\}$.

With respect to particular languages $L \subseteq \Sigma_Y^*$ and $L' \subseteq \Sigma_X^*$, the natural projection is defined as $P^L_{\{\Sigma_X,\Sigma_Y\}}(s) = \{t \in L | P_{\{\Sigma_X,\Sigma_Y\}}(s) = t\}$ and the inverse projection operator $P^{-1,L'}_{\{\Sigma_Y,\Sigma_X\}}(s) = \{t \in L' | P_{\{\Sigma_X,\Sigma_Y\}}(t) = s\}$.

Let $H := \{1, 2, \cdots, m\} \subset \mathbb{N}$ be an index set. Every element $j \in H$ is called a *module* hereafter. In [Con+06], the system to be diagnosed is modeled as a collection of automaton modules $\{G_{i\in H}\}$ and a set of corresponding languages $\{L(G_{j\in H})\}$. Each module $G_{j\in H}$ is called a *local* model with its *local* language. The monolithic model can be obtained by the parallel composition of local models $G_H := \|_{j\in H}\ G_j$ and the monolithic language is defined by $L(G_H) := \|_{j\in H}\ L(G_j)$.

Given $S \subseteq H$. Let $G_S = (X_S, \Sigma_S, \delta_S, x_{0S})$ denote the automaton system with the state space X_S, the set of the event Σ_S, the transition function δ_S and the initial state x_{0S}. When $S = \{j\}$, G_S denotes the individual local module j. When $S = H$, G_S denotes the monolithic model. When $S \subset H, S \neq \emptyset, S \neq \{j\}, j \in H$, G_S denotes the partial system composed by the modules contained in S. $G_S := \|_{z\in S}\ G_z$ is obtained by the parallel composition of the modules G_z with $z \in S$.

Let Σ_{oS} and Σ_{uS} be the set of observable and unobservable events, where $\Sigma_S = \Sigma_{oS} \dot\cup \Sigma_{uS}$. Σ_{fS} is the set of fault events and $\Sigma_{fS} \subset \Sigma_{uS}$.

The event set Σ_S can also be partitioned as $\Sigma_S = \Sigma_{CM_S} \dot\cup \Sigma_{PV_S}$, where $\Sigma_{CM_S} = \Sigma_S \cap [\cup_{z\in H/S}\Sigma_z]$ is the set of common events in G_S and $\Sigma_{PV_S} = \Sigma_S \backslash \Sigma_{CM_S}$ represents the set of private events.

The principle two assumptions are given as follows:

1. $\forall S \subseteq H$, $L(G_S)$ is assumed to be live, i.e., G_S cannot reach a point at which no event can be fired;
2. For each module $j \in H$, the common events of module j are observable, i.e., $\Sigma_{CM_j} \subset \Sigma_{oj}$.

The Assumption 1 is strong. It requires that not only each module j is live, but also each composed module $G_S, S \subseteq H$ is live. It implies also that the occurrence of any fault does not bring the system, or any module, to a deadlock.

The Assumption 2 implies that all the unobservable events including faults are private events.

For a system modeled by a collection of modules, the *Local Diagnosability* is considered as the "monolithic diagnosability" of a local module. The approaches for monolithic diagnosability analysis can be applied for the local diagnosability analysis of each module.

The definition of *Modular Diagnosability* extends the definition of *modular Diagnosability*, which is given as follows:

Definition 33 *(Modular Diagnosability [Con+06]) Let* $H := \{1, 2, \cdots, m\}$, $S \subseteq H$, $G_S := \|_{j \in S} G_j$ *and* $S^- \subseteq S$. *The language* $L(G_S)$ *is modularly diagnosable w.r.t* (Σ_{oj}:$j \in S$) *and* (Σ_{fz}:$z \in S^-$), *if* $\forall z \in S^-$, $\forall f \in \Sigma_{fz}$, $\forall v \in L(G_S)$ *ends with* f, $\exists n \in \mathbb{N}$ *s.t.* $\forall t \in L(G_S)/v, |P_{\{\Sigma_S, \Sigma_{oz}\}}(t)| \geq n \Rightarrow \mathcal{D}(vt) = 1$, *where*

$$\mathcal{D}(vt) := \begin{cases} 1 & \text{if } \omega \in P^{-1, L(G_S)}_{\{\Sigma_{oS}, \Sigma_S\}}[P_{\{\Sigma_S, \Sigma_{oS}\}}(vt)] \Rightarrow f \in \omega \\ 0 & \text{otherwise} \end{cases}$$

In other terms, if a set of modules $S \subseteq H$ is modularly diagnosable w.r.t. the set of faults in each module, the fault in each module is capable to be detected in a finite delay by using only the local observation of each module.

To analyze the modular diagnosability of a system, the definition of F^{M_z}-indeterminate cycle is proposed in [Con+06] as follows.

Definition 34 *Let* $H := \{1, 2, \cdots, m\}$, $S \subseteq H$, $G_S := \|_{z \in S} G_z$. $G'_S = (Q_{o_S}, \Sigma_{oS}, \delta_{o_S}, q_0)$ *is the non-deterministic automaton built from* G_S *by eliminating unobservable events and the diagnoser of* G_S *can be defined as* $G_{d_S} = (Q_{d_S}, \Sigma_{oS}, \delta_{d_S}, q_0)$.

A set of F-uncertain states $q_1, q_2, \cdots, q_n \in Q_{d_S}$ *is said to form an* $F^{M_z}-$*indeterminate cycle if*

1. $q_1, q_2, \cdots, q_n \in Q_{d_S}$ *form a cycle in* G_{d_S} *with* $\delta_{d_S}(q_v, e_v) = q_{v+1}$, $v = 1, \cdots, n-1$, $\delta_{d_S}(q_n, e_n) = q_1$,*where* $e_v \in \Sigma_{oS}$, $v = 1, \cdots, n-1$ *and* $\exists j \in 1, \cdots, n$ *s.t.* $e_j \in \Sigma_{oz}$.
2. $\exists (x^k_v, l^k_v), (y^r_v, \widetilde{l}^r_v) \in q_v$, $v = 1, \cdots, n$, $k = 1, \cdots, m$, *and* $r = 1, \cdots, m'$ *such that:*
   a) $[(F \in l^k_v) \wedge (F \notin \widetilde{l}^r_v)]$, *where* F *represents the label associated with the fault event* $f \in \Sigma_{fz}, z \in S$
   b) *the sequence of states* $\{x^k_v\}, v = 1, \cdots, n$, $k = 1, \cdots, m$, *and* $\{y^r_v\}, v = 1, \cdots, n$, $r = 1, \cdots, m'$, *form cycles in the observer* G'_S *with*
      i. $(x^k_v, e_v, x^k_{v+1}) \in \delta_{o_S}, v = 1, \cdots, n$, $k = 1, \cdots, m$, $(x^k_n, e_n, x^{k+1}_1) \in \delta_{o_S}$, $k = 1, \cdots, m-1$, $(x^m_n, e_n, x^1_1) \in \delta_{o_S}$
      ii. $(y^r_v, e_v, y^r_{v+1}) \in \delta_{o_S}, v = 1, \cdots, n$, $r = 1, \cdots, m'$, $(y^r_n, e_n, y^{r+1}_1) \in \delta_{o_S}$, $k = 1, \cdots, m'-1$, $(y^{m'}_n, e_n, y^1_1) \in \delta_{o_S}$

Here, the symbol M_z of "F^{M_z}-indeterminate cycle" represents the Module z. The notion of "F^{M_z}-indeterminate cycle" is slightly different from the notion of "F-indeterminate cycle". If the fault label in 2(a) of Definition 34 corresponds to the fault f in module z, i.e., $f \in \Sigma_{fz}$, it requires that there exists at least one observable event from module z in the cycle of the diagnoser G_{d_S}. In other words, for an F^{M_z}-indeterminate cycle, the considered fault and at least one observable event of the cycle belong to the same module. Based on the notion of F^{M_z}-indeterminate cycle, the sufficient and necessary condition of modular diagnosability is proposed as follows:

Theorem 7 *Consider the language $L(G_S)$ generated by $G_S := \|_{z \in S}\ G_z$, $L(G_S)$ is modularly diagnosable w.r.t. (Σ_{oz}:$z \in S$) and (Σ_{fz}: $z \in S$), if there is no $F^{M_z}-$indeterminate cycle in the diagnoser G_{d_S}.*

According to the definition of F^{M_z}-indeterminate cycle, it can be deduced that an F^{M_z}-indeterminate cycle is an F-indeterminate cycle, but the inverse proposition is not true. Therefore, the modular diagnosability is less strict than monolithic diagnosability. The relationship between monolithic, local and modular diagnosability is given in the following proposition:

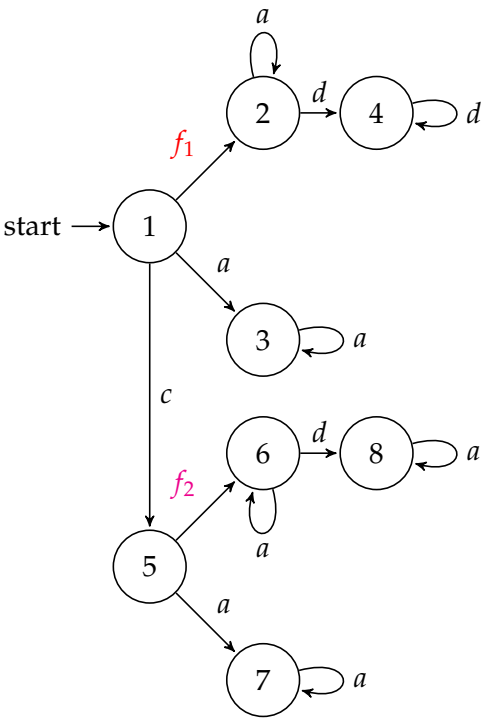
Proposition 10

1. *Let $H := \{1, 2, \cdots, m\}$, $S \subseteq H$, $G_S := \|_{z \in S}\ G_z$. If the language $L(G_S)$ is monolithically diagnosable w.r.t. (Σ_{oz}:$z \in S$) and (Σ_{fz}:$z \in S$) then the language $L(G_S)$ is modularly diagnosable w.r.t (Σ_{oz}:$z \in S$) and (Σ_{fz}:$z \in S$).*
2. *Let $H := \{1, 2, \cdots, m\}$, $S \subseteq H$, $G_S := \|_{j \in S}\ G_j$. If the language $L(G_z)$ of each module $z \in S$ is locally diagnosable w.r.t Σ_{oz} and Σ_{fz} then the language $L(G_S)$ is modularly diagnosable w.r.t (Σ_{oj}:$j \in S$) and Σ_{fz}.*

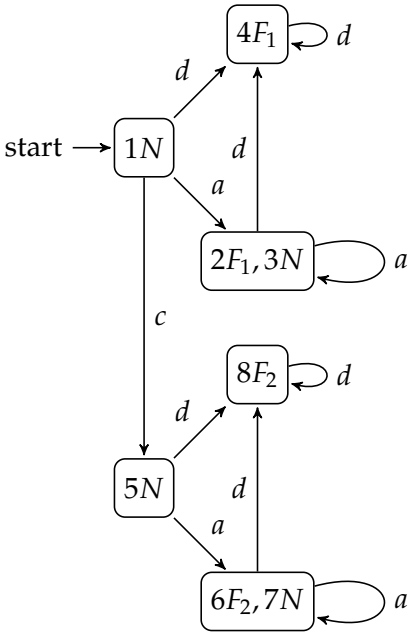
The inverse propositions of these two propositions are not true. (1) If the language $L(G_S)$ is modularly diagnosable, it cannot be deduced that the language $L(G_S)$ is monolithically diagnosable; (2) if the language $L(G_S)$ is modularly diagnosable, it cannot be deduced that each module $z \in S$ is locally diagnosable. The following example is given to further explain the difference between monolithic diagnosability and modular diagnosability.

Proposition 11 *[Con+06] Let $H := \{1, 2, \cdots, m\}$, $S \subset H$, $G_S := \|_{z \in S}\ G_z$. If the language $L(G_S)$ is modularly diagnosable w.r.t. (Σ_{oj}: $j \in S$) and Σ_{fz} ($z \in S$), then the language $L(G_H)$ is modularly diagnosable w.r.t. (Σ_{oj}: $j \in H$) and Σ_{fz}.*

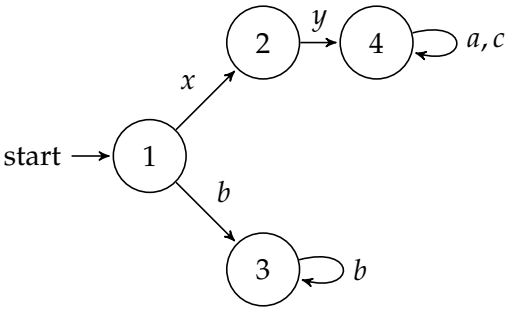
According to Proposition 11, in order to check the modular diagnosability G_H ($H := \{1, 2, \cdots, m\}$) w.r.t. (Σ_{oj}: $j \in H$) and Σ_{fz} ($z \in H$), it is not necessary to build directly the parallel composition of the local diagnoser G_{d_z} (which contains an $F^{M_z}-$indeterminate



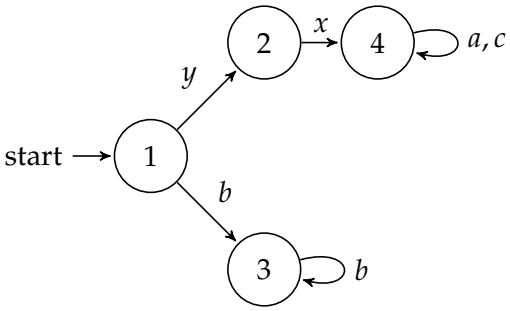
Module G_1



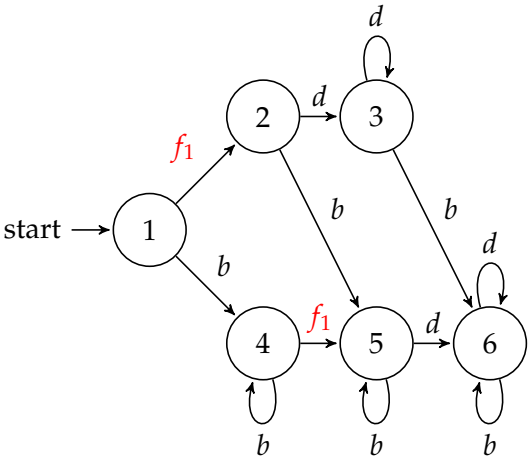
Local diagnoser of module G_1



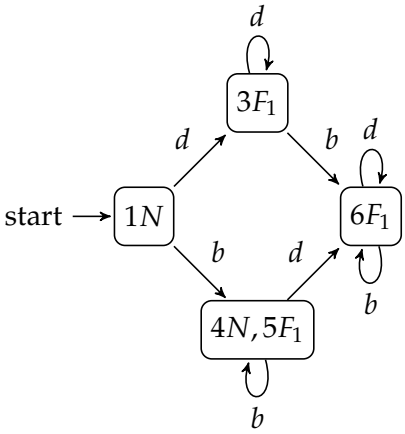
Module G_2



Module G_3



Monolithic system G_H



Monolithic diagnoser G_{dH}

Figure 4.5 – Modular diagnosability and monolithic diagnosability

cycle) with the diagnoser $G_{d_{S_c}}$, where $S_c \subseteq H \backslash \{z\}$. If the $F^{M_z}-$indeterminate cycle does not survive in $G_{d_z} || G_{d_{S_c}}$, i.e., $L(G_S)$ ($S = S_c \cup \{z\}$) is modularly diagnosable, then it can be deduced that $L(G_H)$ is modularly diagnosable.

Example 47 *Considering the system in Figure 4.5 modeled by a collection of modules: G_1, G_2 and G_3. For module G_1, $\Sigma_1 = \Sigma_{o1} \cup \Sigma_{u1}$, where $\Sigma_{o1} = \{a,\ c,\ d\}$ and $\Sigma_{u1} = \Sigma_{f1} = \{f_1,\ f_2\}$. f_1, f_2 belong to different fault classes, i.e., $f_1 \in \Sigma^1_{f1}$ and $f_2 \in \Sigma^2_{f1}$. For module G_2, $\Sigma_2 = \Sigma_{o2} \cup \Sigma_{u2}$, where $\Sigma_{o2} = \{a,\ b,\ c,\ x,\ y\}$ and $\Sigma_{u2} = \emptyset$. For module G_3, $\Sigma_3 = \Sigma_{o3} \cup \Sigma_{u3}$, where $\Sigma_{o3} = \{a,\ b,\ c,\ x,\ y\}$ and $\Sigma_{u3} = \emptyset$. First, the local diagnosability of the three modules is analyzed (module G_2 and module G_3 are locally diagnosable because of no existence of fault). The diagnoser of module G_1 is built. Module G_1 is not locally diagnosable w.r.t. Σ^1_{f1} and Σ^2_{f1}, because an F_1-indeterminate cycle and an F_2-indeterminate cycle are found.*

Afterwards, the monolithic system G_H is built by the parallel composition, i.e., $G_H := \|_{j \in H} G_j, H = \{1,\ 2,\ 3\}$. The monolithic diagnoser G_{dH} is built. Since there exist an F_1-indeterminate cycle, the monolithic system is not monolithically diagnosable. However, the system G_H is modularly diagnosable. The F_1-indeterminate cycle is not an $F_1^{M_1}$-indeterminate cycle, because the fault f_1 belongs to module G_1 but there is no observable event of this cycle that belongs to Σ_{o1} ($b \notin \Sigma_{o1}$).

From a practical viewpoint, if the system is modularly diagnosable, i.e., there is no F^{M_z}-indeterminate cycle, it is sufficient to use local diagnoser for on-line diagnosis. Even though there exist an F-indeterminate cycle in the local diagnoser, the modular diagnosability property ensures that the module will not stay forever in the states of the F-indeterminate cycle when the complete system is operating. If a local diagnoser indicates that the module reaches a state of F-indeterminate cycle, in a finite number of observable events of the module, the state of the local diagnoser will be out of the indeterminate cycle, because the indeterminate cycle will be blocked while considering the behavior of the complete system.

In [Con+06], an approach is developed to analyze the modular diagnosability of a system, which is modeled as a collection of modules $\{G_{j \in H}\}$. The main idea is presented as follows:

1. For all $j \in H$, build the diagnoser G_{d_j} of each module based on its automaton model G_j;
2. If all the modules are locally diagnosable, the system is modularly diagnosable. Otherwise, for each locally non-diagnosable module $z \in H$:
   a) For all $S_c \subseteq H/\{z\}$ s.t. G_S has common events with G_z:
      i. Build the parallel composition of G_{d_z} and $G_{d_{S_c}}$;
      ii. Verify if there exists an $F^{M_z}-$indeterminate cycle:

A. If the answer is yes, continue;

B. Else, output "The system is modularly diagnosable w.r.t. (Σ_{oj}:$j \in S$) and Σ_{fz}", where $S = S_c \cup \{z\}$ and break;

iii. Output "The system is not modularly diagnosable w.r.t. (Σ_{oj}:$j \in S$) and Σ_{fz}", where $S = S_c \cup \{z\}$.

The main advantage of the modular approach in [Con+06] is that if the modular diagnosability is fulfilled, it is sufficient to use the local diagnoser of each module to diagnose all faults without any communication. It is an easy solution to implement, in practice, for on-line diagnosis. However, if the system model is provided by a set of module models, the assumption that $\forall S \subseteq H$, $L(G_S)$ is live, is strong. For $S = H$, it needs to build the composed modules and the monolithic model to check the assumption, because the monolithic model may deadlock even thought each module is live. It violates the original idea of the modular architecture (to diagnose the fault without building the monolithic system and the monolithic diagnoser). Moreover, the operation of parallel composition may cause the combinatorial explosion problem.

In [Pen04], the diagnosability is analyzed for a system modeled by a set of communicating components. The local diagnosability of each component is studied in order to provide more accurate information than the monolithic diagnosability. In [MP13], the authors redefine the local diagnosability and modular diagnosability in a specification-based manner. A new approach for specification-based modular diagnosability analysis using verifier is proposed. In [Zho+08], the modular architecture is investigated in a decentralized setting. A new architecture, namely, "decentralized modular architecture" is proposed. The diagnostic decisions are given locally by the diagnosers of the modules, but it requires that each fault can be detected by at least one local diagnoser based on its own observation. A method is proposed for modular diagnosability analysis by reducing this property to an instance of a codiagnosability property. In [Sch13], an approach for verification of modular diagnosability with local specifications is proposed. This approach extends the abstraction-based language-diagnosability analysis to a modular architecture. The assumption of the liveness in [Con+06] is released, i.e., there can be a deadlock in modules or monolithic model. However, if there is a deadlock just after firing a fault, this method can neither be applied for the diagnosis. The methodology to find the proper Loop-preserving Observer (LPO) is not systematically provided. A method to decompose a failure specification into multiple local specifications needs to be developed to apply this approach. In [Pen+15], a colored Petri net (CPN) diagnoser is developed for fault diagnosis using a modular architecture in the framework of PN. The CPN diagnoser is exactly equivalent to the diagnoser in [Sam+95] but with a reduced graphical representation, so that it can be used to implement a modular diagnoser of a large-scale system. In [Cab+15b], the modular architecture is applied for on-line diagnosis. This approach is based on the construction of a PN diagnoser named synchronized Petri net diagnoser

(SPND). The synchronous diagnosability is defined and studied w.r.t. the language of modules.

The modular architecture is easy to implement in practice for on-line diagnosis. It is sufficient to use the local diagnoser of each module, if the modular diagnosability property is fulfilled. However, the scope of use is limited because of the strong assumption of liveness. Moreover, the verification of modular diagnosability may still cause computational complexity and combinatorial explosion problem.

4.1.3 Distributed diagnosis

Similar to the modular approach, the distributed approaches achieve the fault diagnosis using a set of local diagnosers without building a global diagnoser. The distributed architecture is shown in Figure 4.6. Each local model knows only its partial model and a local diagnoser is built based on the local model in order to perform the fault diagnosis. Additionally, the diagnostic information is communicated between the local diagnosers under a defined communication protocol. The protocol ensures the consistency among the diagnosers, to avoid the possible inconsistent fault information given by different local diagnosers.

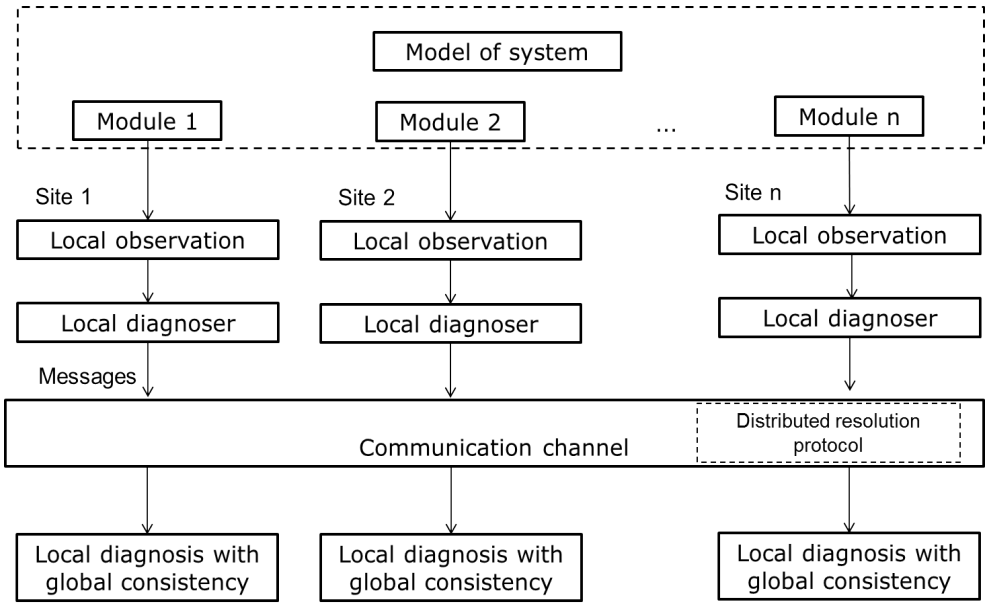


Figure 4.6 – Architecture of Distributed diagnosis approaches

In [GL03], a distributed approach is proposed for fault diagnosis of a system modeled by labeled Petri net (LPN). The architecture of this approach is shown in Figure 4.7. The input model of this approach is a monolithic LPN model and under some conditions, it is decomposed into several place-bordered LPN modules. Based on local observations, the local diagnosers are built. The communication channel is via the set of common places between modules. The variety of the number of tokens in the common places is considered

as the message generated by a local model. The other local models get the message and update their states. Each local diagnoser is capable to give its diagnostic decision after communicating with the other local diagnosers.

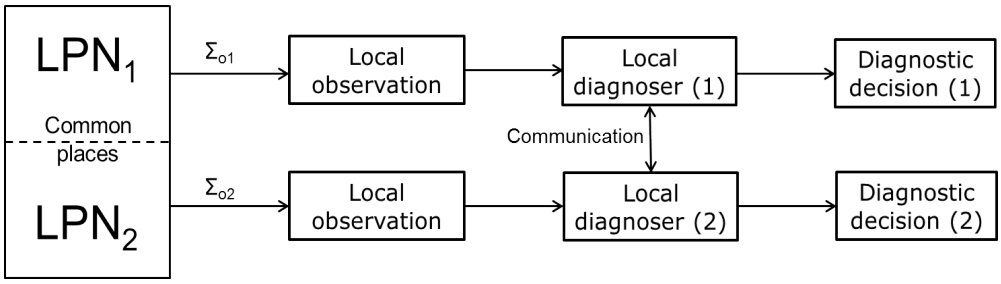


Figure 4.7 – Architecture of the approach in [GL03]

Let us consider a LPN model $LPN = (N, M_0, \Sigma, \mathcal{L})$. By using the approach in [GL03], the LPN is decomposed into two local LPN models: $LPN_1 = (N_1, M_{01}, \Sigma_1, \mathcal{L}_1)$ and $LPN_2 = (N_2, M_{02}, \Sigma_2, \mathcal{L}_2)$, where $N_1 = (P_1, T_1, pre_1, post_1)$ and $N_2 = (P_2, T_2, pre_2, post_2)$. $\Sigma_o = \Sigma_{o1} \cup \Sigma_{o2}$ and $\Sigma_{o1} \cap \Sigma_{o2} = \emptyset$. Moreover, for a fault transition $f \in T_1$ (respectively, T_2) and $f \in T_f^i$, there does not exist a fault transition $f' \in T_2$ (respectively, T_1) s.t. $f' \in T_f^i$. The two local LPN models are defined by the following conditions:

1. $\forall t \in T_o$, if $\mathcal{L}(t) \in \Sigma_{o1}$ (respectively, Σ_{o2}), $t \in T_1$ (respectively, T_2);

2. $\forall f \in T_f$, if $f \in T_f^k$ and $f \in T_1$ (respectively, T_2), there does not exist $f' \in T_f^k$ and $f' \in T_2$ (respectively, T_1);

3. Denote $P_c = P_1 \cap P_2$ the set of common places. $\forall t \in T$, if the firing of t puts tokens into P_c or removes tokens from P_c, then $t \in T_o$.

The communication protocol needs to be defined, because if the local diagnosers do not inform their change of markings to each other, the state estimation of the local diagnosers is incomplete or wrong. The defined protocol is simple, which allows the local diagnosers to send to each other the change of tokens' number in common places. Hence, the state of local diagnoser $i, i \in \{1, 2\}$can be defined as:

$$D^i = \begin{bmatrix} \cdots & M_j^i & \cdots \\ - & - & - \\ \cdots & l_{f,j}^i & \cdots \\ - & - & - \\ \cdots & l_{m,j}^i & \cdots \end{bmatrix}$$

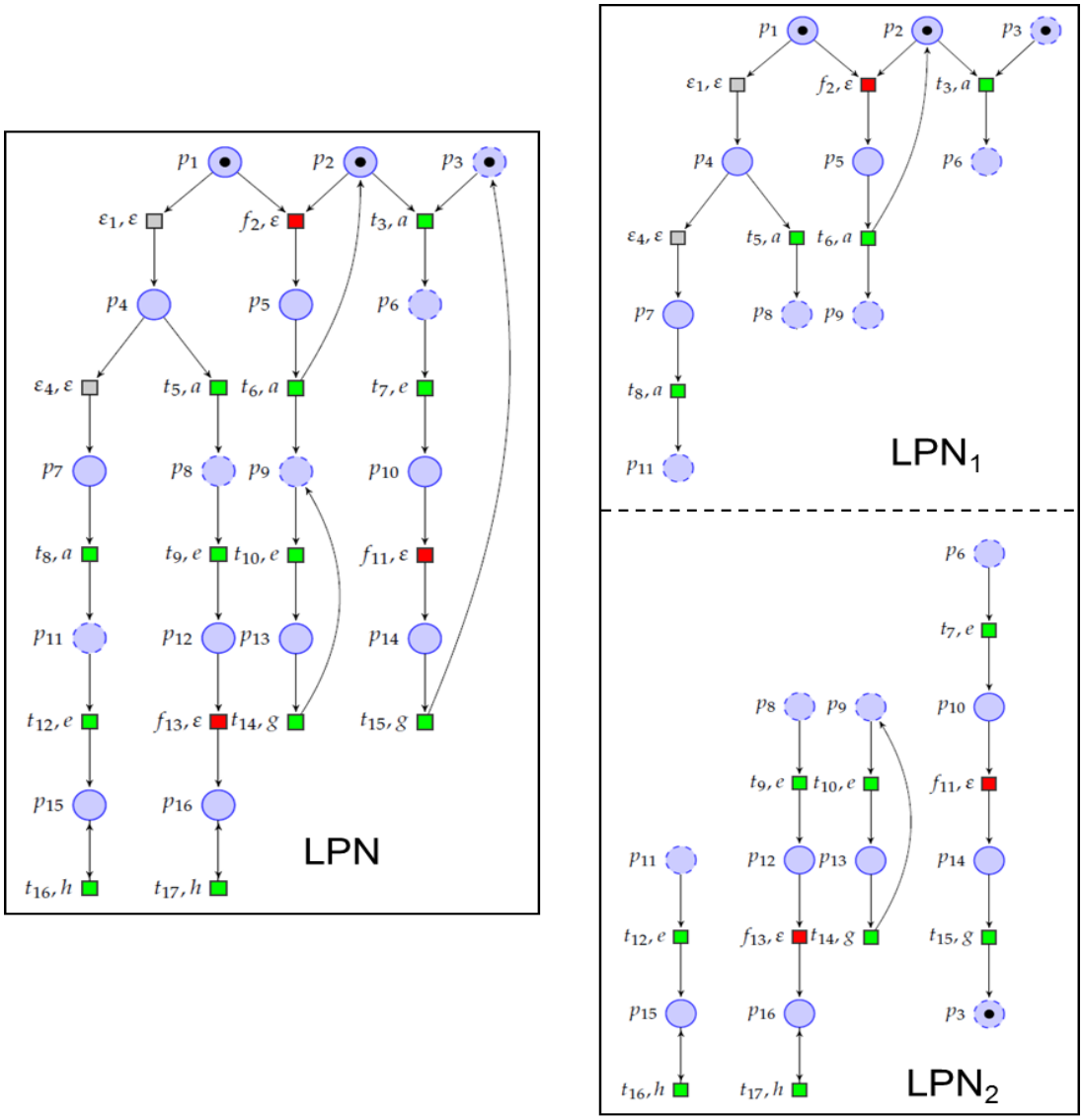


Figure 4.8 – The model of LPN and its decomposition: LPN_1 and LPN_2

where, M_j^i is the marking of local model and the markings in a diagnoser state are computed in the same way with the diagnoser approach in [Sam+95]. $l_{f,j}^i$ is the tag to indicate the occurrence of the fault (it is defined in the same way with the fault tag in [Sam+95] and because of the Assumption 2, there is no need to distinguish the fault tags in different local diagnosers). $l_{m,j}^i$ is the message label for the purpose of communication. If $t \in T_o$ (we consider only that t is observable because of the Assumption 3) is fired at M^i with its message label l_m^i, the obtained marking is $M^{i'} = M^i + C(\cdot, t)$, where C is the incidence matrix. The new message label is computed by the message label function MLP as follows:

$$l_m^{i'} = MLP(l_m^i, t) = \begin{bmatrix} l_m^i \\ C_{P_c}(\cdot, t) \end{bmatrix}$$

where C_{P_c} is obtain by remove the rows corresponding to all the places $p \notin P_c$ of the

incidence matrix C. By exchanging the message label, local diagnosers can compute the exact state and provide the correct diagnostic decision. It is proved that the state of monolithic diagnoser can be obtained by merging the states of local diagnosers.

Example 48 *Considering the LPN model in Figure 4.8. For the monolithic model LPN,* $T_o = \{t_3, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{12}, t_{14}, t_{15}, t_{16}, t_{17}\}$, $T_{reg} = \{\varepsilon_1, \varepsilon_4\}$ *and* $T_f = T_f^1 \cup T_f^2$, *where* $T_f^1 = \{f_2\}$ *and* $T_f^2 = \{f_{11}, f_{13}\}$. $\mathcal{L}(t_3) = \mathcal{L}(t_5) = \mathcal{L}(t_6) = \mathcal{L}(t_8) = a$, $\mathcal{L}(t_7) = \mathcal{L}(t_9) = \mathcal{L}(t_{10}) = \mathcal{L}(t_{12}) = e$, $\mathcal{L}(t_{14}) = \mathcal{L}(t_{15}) = g$, $\mathcal{L}(t_{16}) = \mathcal{L}(t_{17}) = h$ *and the label of all the unobservable transition is* ε. *Under the above assumptions, the LPN can be decomposed into* LPN_1 *and* LPN_2 *as it is shown in Figure 4.8.* $\Sigma_{o1} = \{a\}$ *and* $\Sigma_{o2} = \{e, g, h\}$. $T_{f1} = T_f^1 = \{f_2\}$ *and* $T_{f2} = T_f^2 = \{f_{11}, f_{13}\}$. *The set of common places is* $P_c = \{p_6, p_8, p_9, p_{11}\}$ *(shown with dashed lines). Assuming that the sequence of observable events "aeh" is observed. The initial states of the two local diagnosers can be obtained by using the same idea of the diagnoser approach in [Sam+95]:*

$$D_0^1 = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{11} & |l_f^1| \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & |0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & |1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & |1 \end{bmatrix},$$

$$D_0^2 = \begin{bmatrix} p_3 & p_6 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & |l_f^2| \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |0 \end{bmatrix}.$$

After observing 'a', the states of the two local diagnosers are computed that contain the message labels.

$$D_1^1 = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{11} & |l_f^1| & l_m^1(p_3 & p_6 & p_8 & p_9 & p_{11}) \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & |0| & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & |0| & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & |0| & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & |1| & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & |1| & -1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & |1| & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$D_1^2 = \begin{bmatrix} p_3 & p_6 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & |l_f^2| & l_m^2(p_3 & p_6 & p_8 & p_9 & p_{11}) \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |0| & -1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |0| & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |0| & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & |0| & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Similarly, the states after observing "g" and "h" are

$$D_2^1 = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{11} & |l_f^1| & l_m^1(p_3 & p_6 & p_8 & p_9 & p_{11} & p_3 & p_6 & p_8 & p_9 & p_{11}) \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |0| & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & |0| & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & |1| & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |0| & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |1| & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |1| & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix},$$

$$D_2^2 = \begin{bmatrix} p_3 & p_6 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & |l_f^2| & l_m^2(p_3 & p_6 & p_8 & p_9 & p_{11} & p_3 & p_6 & p_8 & p_9 & p_{11}) \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & |0| & -1 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & |0| & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & |0| & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & |0| & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & |1| & -1 & 1 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & |1| & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}.$$

$$D_3^1 = D_2^1,$$

$$D_3^2 = \begin{bmatrix} p_3 & p_6 & p_8 & p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & |l_f^2| & l_m^2(p_3 & p_6 & p_8 & p_9 & p_{11} & p_3 & p_6 & p_8 & p_9 & p_{11}) \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & |0| & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & |1| & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}.$$

It is worth noticing that it take lots of memory to store the message labels. Therefore, in [GL07], the communication messages between modules are simplified by using an encoding-based method. Moreover, the approach in [GL03] is improved by releasing the Assumption 2.

In [SW02; SW04], a distributed approach with communication is developed. Each local module has its own local diagnoser for diagnosis. Local diagnosers communicate with each other via undamaged communication channels for refinement purposes. In [JB05], the distributed diagnosis for large interacting systems is developed. This study works on the LPN systems that consist of a set of place-bordered LPN. It assumes that the interactions between modules are modeled by tokens that can unobservably pass from one module to another via common places. After the occurrence of an observable event, each module computes a set of minimal explanations of an observable sequence and estimate the number of tokens that could have been produced at the output places of a local model. The main idea is to use the notion of minimal explanations to explain the observable system behavior. In [Fan+13], the Integer Linear Programming (ILP) technique is applied for distributed diagnosis of place-bordered LPN system. A comparison between monolithic and distributed diagnosers is presented. Comparing to the study in [GL07], no off-line computation is necessary and there is no need for the redesign of local diagnosers. The distributed diagnosis is improved by few communications among local diagnosers.

The advantage of distributed architecture is the scalability and robustness of fault diagnosis. The communication between local diagnosers ensures the correctness of fault diagnostic decision. Nevertheless, the approaches are proposed with different assumptions and the assumptions sometimes reduces the scope of use in practice. Moreover, there is no method to analyze the diagnosability property of the system. Consequently, the monolithic approaches is applied to ensure that all faults can be detected by using distributed architecture.

4.1.4 Synthesis of literature review

In this section, a brief literature review is given about three diagnosis architectures: decentralized architecture, modular architecture and distributed architecture. Their main

distinguishing features are as follows:

Decentralized architecture: The input system model is the monolithic system model, which is partitioned into several sites. Each site has full knowledge of the global model but has only local observation of the system. A local diagnoser is built based on the local observation of the whole system. Local diagnosers cannot communicate directly with each other, but they are able to provide their local decision to a coordinator. The coordinator gives the final diagnostic decision based on a communication protocol.

Modular architecture: The input system model is a collection of module models. The different modules are allowed to have common events, and the monolithic model can be obtained by building the parallel composition of all modules. Local diagnosers are built based on each module model. If the modular diagnosability property is fulfilled, the local diagnoser is capable of giving the diagnostic decisions w.r.t. the faults of its own local module without any communication with other local diagnosers. The whole system diagnosability (modular diagnosability) is introduced by taking into account the fault sequence which, although locally indistinguishable in one module, becomes distinguishable due to concurrency with other modules.

Distributed architecture: The distributed architecture can be reviewed as a mix of the two previous architectures. The diagnostic decisions corresponding to the faults in local models are given by local diagnosers. There is no coordinator, but the diagnostic information can be communicated between each of the local diagnosers under a defined communication protocol.

These architectures are not mutually exclusive and the differences between them are sometimes vague. It is meaningless to discuss independently which architecture is better. Each architecture has its own scope of use based on different assumptions and different types of models. It is suggested to use the suitable diagnosis architecture for each specific diagnostic problem.

4.2 Modular diagnosability analysis using LPN model

This section focuses on the modular diagnosability analysis. We use the notion of modular diagnosability defined in [Con+06], but we provide a new approach for modular diagnosability analysis. The system is modeled by LPN, which gives a more compact representation of DES as it was presented before. In [Con+06], the system is modeled directly by a collection of automaton modules and the model of each module is given. However, this thesis starts from the global LPN model of the system. Under certain conditions, the global LPN model is decomposed into several modules. Afterwards, we propose an approach for modular diagnosability analysis in order to reduce the computational complexity.

4.2.1 Definition of LPN module, sound decomposition and modular diagnosability using LPN

In this section, the LPN module is defined at first. Then, the definition of *sound decomposition* is provided. The definition of modular diagnosability using LPN is introduced finally.

First, let us define the notion of an LPN module.

Definition 35 *Given a global LPN model* $LPN = (N, M_0, \Sigma, \mathcal{L})$. *An LPN module is a part of* LPN. $LPN_i = (N_i, M_{0i}, \Sigma_i, \mathcal{L}_i)$ *(*$N_i = (P_i, T_i, pre_i, post_i)$*) is a module such that:*

1. $P_i \subseteq P$;
2. $T_i \subseteq T$ *s.t.* $\forall t \in T_i$, $\bullet t \cap P_i \neq \emptyset$ *and* $t \bullet \cap P_i \neq \emptyset$, *where* $\bullet t$ ($t\bullet$) *is the pre-places (post-places) of* t;
3. $\Sigma_i = \{l | \ (\exists t \in T_i) \wedge (\mathcal{L}(t) = l)\}$;
4. M_{0i} *is defined as:* $\forall p_j \in P_i, M_{0i}(p_j) = M_0(p_j)$ *and* $\exists p_k \in P_i$ *s.t.* $M_{0i}(p_k) > 0$.

It is worth noticing that there exists at least one token in a place of each module. Therefore, each module is capable to model independently its local behaviors. In order to analyze modular diagnosability of the global LPN model, a decomposition is required, which covers the global LPN. Furthermore, the communication between two modules is via synchronized events. It means that only transitions can be shared between different modules. T_{oi} and T_{ui} denotes respectively the set of observable transitions and the set of unobservable transitions in module LPN_i. The set of fault transitions in LPN_i is denoted as $T_{fi} = \cup_{j=1}^{n} T_{fi}^{j}$.

Since PN is more powerful for modeling DES than automaton and the concurrent process is well represented, a system is favorably modeled by a monolithic PN even its function is modularly designed. For example, the multi-track level crossing benchmark in [Liu+16] (it will be analyzed in Section 6.2) consists of railway traffic, LC controller and barriers. It is modularly designed but a monolithic PN is obtained considering the concurrent processes. In order to apply the modular diagnosis, it is necessary to decompose the monolithic model into modules. Therefore, the definition of *sound decomposition* is given as follows:

Definition 36 *([Pen+15]) Given the global LPN model* $LPN = (N, M_0, \Sigma, \mathcal{L})$ *(*$N = (P, T, pre, post)$*). A set of modules* $LPN_1, \cdots, LPN_m$ *(*$H = \{1, \cdots, m\}$*) is a* sound *decomposition if:*

1. *Any place in* P *belongs to one and only one module* LPN_i;

2. *Any transition in T belongs to at least one module* LPN_i;

3. *For each observable event* l, *if* l *belongs to several sets of observable events* $\{\Sigma_{oi_1}, \cdots \Sigma_{oi_n}\}$ $(n \leq m)$, *then for any transition* $t \in T$ *s.t.* $\mathcal{L}(t) = l$, $\bullet t \subseteq \cup_{j=1}^{n} P_{i_j}$ *and for* $j \in \{1, \cdots, n\}$, $\bullet t \cap P_{i_j} \neq \emptyset$;

According to Condition 1 and Condition 2 of Definition 36, there is no place or transition that is not in any modules, i.e., the decomposition covers the global LPN model. Condition 1 states that the sets of places $\{P_1, \cdots, P_m\}$ are disjoint, i.e., each module does not share resources with any other module (each module has its own places). Condition 2 states that a transition can be shared by different modules. Condition 3 states that if an observable event l belongs to a set of modules, i.e., $l \in \Sigma_{oi_j}$, then for all $(t \in T) \wedge (\mathcal{L}(t) = l)$, at least one pre-place of the transition t, belongs to P_{i_j}.

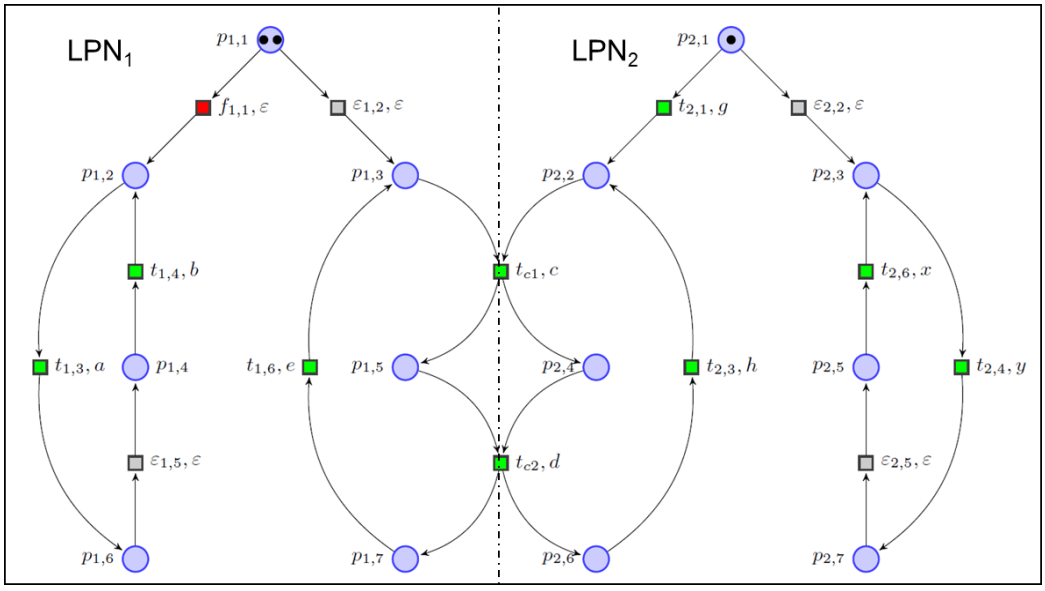


Figure 4.9 – The model of LPN and its sound decomposition: LPN_1 and LPN_2

Example 49 *Let us consider the LPN model in Figure 4.9. The LPN model* $LPN = (N, M_0, \Sigma, \mathcal{L})$ *can be decomposed as* $LPN_1 = (N_1, M_{01}, \Sigma_1, \mathcal{L}_1)$ *and* $LPN_2 = (N_2, M_{02}, \Sigma_2, \mathcal{L}_2)$, *which is a* sound *decomposition.* $P = P_1 \cup P_2$, *and* $T = T_1 \cup T_2$. $T_1 \cap T_2 = \{t_{c1}, t_{c2}\}$, *where* t_{c1} *and* t_{c2} *are shared observable transitions labeled by c and d. Only* LPN_1 *contains a fault transition* $f_{1,1}$.

Based on Definition 35 and Definition 36, the main assumptions of this study is given as follows:

1. The global LPN model is bounded and can be decomposed into a set of modules, which is a *sound* decomposition;

2. The transitions shared by several modules are observable (the unobservable transition including the fault transition are not shared);
3. The classes of fault transitions in one module is different from the classes of any other modules. Without loss of generality, there is at most one fault class in each module (if there are several fault classes, we can analyze one by one);
4. There could be deadlocks in one module;
5. There could be cycles of unobservable transitions.

The Assumption 2 implies that the communication between modules is via synchronization of observable transitions having the same label. The Assumption 3 implies that there do not exist two fault transitions that belong to different modules, such that the two fault transitions are in the same fault class. It means that if a module is locally diagnosable w.r.t. its fault classes, the faults in these fault classes can be detected in a finite delay by only using the local observation of this module. The strong assumption of liveness in [Con+06] is released. It is worth noticing that if $\exists S \subseteq H$, s.t. $L(LPN_S)$ is not live, then there must exist a marking at which no transition is enabled (i.e. the tokens in places will never move). $L(LPN_S)$ can become live by adding a regular unobservable transition whose pre-place and post-place is the same place that contains a token that does not move.

Before giving the definition of modular diagnosability for LPN model, some notions are given as follows:

Let T_X and T_Y be any sets of events. $P_{\{T_X,T_Y\}} : T_X^* \to T_Y^*$ denotes the natural projection and $P^{-1}_{\{T_Y,T_X\}} : T_Y^* \to 2^{T_X^*}$ is the inverse projection operator. $P_{\{T_X,T_Y\}}$ is defined as follows:

1. $P_{\{T_X,T_Y\}}(\lambda) := \lambda$
2. $P_{\{T_X,T_Y\}}(t) := \begin{cases} t & \text{if } t \in T_Y \\ \lambda & \text{if } t \notin T_Y \end{cases}$
3. $P_{\{T_X,T_Y\}}(\sigma t) := P_{\{T_X,T_Y\}}(\sigma)P_{\{T_X,T_Y\}}(t)$ (for $\sigma \in T_X^*, t \in T_X$)

The inverse projection operator is denoted as $P^{-1}_{\{T_Y,T_X\}}(s) = \{t \in T_X^* | P_{\{T_X,T_Y\}}(t) = s\}$.

Definition 37 *(Modular Diagnosability of LPN) Given the LPN model LPN. Let* $H := \{1, 2, \cdots, m\}$ *and* $LPN_1, \cdots, LPN_m$ *a sound decomposition of LPN.* $S \subseteq H$, $LPN_S := \|_{j \in S}\ LPN_j$ *and* $S^- \subseteq S$. *The language* $L(LPN_S)$ *is modularly diagnosable w.r.t* (Σ_{oj}:$j \in S$) *and* (T_{fj}:$j \in S^-$), *if* $\forall z \in S^-$, $\forall t_f \in T_{fz}$ *there do not exist two sequences of transitions* $\sigma_1, \sigma_2 \in T_S^*$, *which satisfy the following conditions:*

1. $\mathcal{L}_S(\sigma_1) = \mathcal{L}_S(\sigma_2)$;

2. $\forall t_f \in T_{fz}$, $t_f \notin \sigma_1$ *and* $\exists t_f \in T_{fz}$ *such that* $t_f \in \sigma_2$;

3. *Assuming that* $\sigma_2 = \sigma_{2,1} t_f \sigma_{2,2}$, *where* $\sigma_{2,1}, \sigma_{2,2} \in T_S^*$. $|P_{\{T_S,T_z\}}(\sigma_{2,2})|$ *is infinite.*

In other terms, the language $L(LPN_S)$ is modularly diagnosable w.r.t (Σ_{oj}:$j \in S$) and (T_{fj}:$j \in S^-$) if there do not exist two sequences with the same observation on Σ_{oS}, s.t. one does not contain any fault transition in T_{fz} ($\forall z \in S^-$); the other one contains a fault transition in T_{fz} and after the occurrence of the fault transition, the faulty sequence can be arbitrarily long w.r.t. the projection on T_z. If there does not exist such pair of sequences, it implies that the fault transition can be distinguished w.r.t. Σ_{oz}, while the transitions in T_z occur continuously after the fault transition.

It is worth noticing that this definition is not equivalent to Definition 33 (page 112):

1. According to the assumption of liveness in [Con+06] and there is no unobservable cycle, each module (or composed module) continues generate observable events after firing the fault of this module. Therefore, in Definition 33, for the post language t after firing the fault, $|P_{\{\Sigma_S,\Sigma_{oz}\}}(t)| \geq n \Rightarrow \mathcal{D}(vt) = 1$, i.e., in a finite number of observable events in Σ_{oz}, it can be deduced that the fault has occurred.

2. In this thesis, the assumption of liveness in [Con+06] is removed and we assume that there can be cycles of unobservable transitions. Therefore, in Definition 37, for the fault sequence $\sigma_2 = \sigma_{2,1} t_f \sigma_{2,2}$, where $\sigma_{2,1}, \sigma_{2,2} \in T_S^*$, $|P_{\{T_S,T_z\}}(\sigma_{2,2})|$ can be infinite, instead of $|P_{\{T_S,T_{oz}\}}(\sigma_{2,2})|$. The Definition 37 is equivalent to Definition 33, if we replace $|P_{\{T_S,T_z\}}(\sigma_{2,2})|$ by $|P_{\{T_S,T_{oz}\}}(\sigma_{2,2})|$.

In the following sections, the reduction rules for modular diagnosability is introduced at first. The local diagnosability analysis and modular diagnosability analysis are addressed afterwards.

4.2.2 Reduction rules for modular diagnosability

In this section, some reduction rules will be proposed for modular diagnosability analysis.

Similar to the monolithic diagnosability analysis, the given LPN model can be simplified before modular diagnosability analysis by using some reduction rules. The aim is to reduce the state space for modular diagnosability analysis.

The reduction rules in Section 3.2.1 can be applied to simplify the given LPN model for modular diagnosability analysis. However, we need to restrict the scope of use of these rules.

Constraint 1: The reduction rules in Section 3.2.1 can only be applied to the uncommon transitions.

Actually, the communication between modules is made via common transitions. If the common transitions are reduced, the information of the communication is lost and the modular diagnosability property may not be preserved.

Theorem 8 *By using reduction rules (1)-(7) in Section 3.2.1 under Constraint 1, the modular diagnosability of the reduced LPN model keeps consistent with the modular diagnosability of the initial LPN model.*

Proof: The proof is similar to the proof of Theorem 3 (page 60) and Theorem 4 (page 64). The Definition 37 of modular diagnosability is used instead of the definition of monolithic diagnosability.

Given the global LPN model $LPN = (N, M_0, \Sigma, \mathcal{L})$ ($N = (P, T, pre, post)$). A set of modules $LPN_1, \cdots, LPN_m$ ($H = \{1, \cdots, m\}$) is a *sound* decomposition. For $S \subseteq H$ and $z \in S$, if LPN_S is not modularly diagnosable w.r.t. $(\Sigma_{oj} : j \in S)$ and T_{fz} $(z \in S)$, assuming that there exist two sequences of transitions σ_1 and σ_2 such that the three conditions in Definition 37 are satisfied. We denoted σ_1 and σ_2 as a couple of troubled sequences for modular diagnosability.

According to Constraint 1, the common transitions cannot be reduced. Considering a firing sequence of transitions $\sigma \in T^*$ such that $\sigma = \sigma_i t_c \sigma_j$ where $\sigma_i, \sigma_j \in T^*$ and t_c is a common transition. If the reduction rules (1)-(7) can be applied on σ, the reduced sequence of σ is $\sigma' = \sigma'_i t_c \sigma'_j$ and $\widehat{\sigma}$ is a firing sequence of the reduced LPN model, according to Proposition 1.

Considering the pair of troubled sequences σ_1 and σ_2. If the rules (1)-(5) are applied, the reduced sequences σ'_1 and σ'_2 are still a pair of troubled sequences for modular diagnosability according to the proof of Theorem 3. If the rules (6) and (7) are applied, we need to consider the two situations in the proof of Theorem 4 and the reduced sequences σ''_1 and σ''_2 are still a couple of troubled sequences for modular diagnosability.

Otherwise, if LPN_S is modularly diagnosable w.r.t. $(\Sigma_{oj} : j \in S)$ and T_{fz} $(z \in S)$, there does not exist a pair of troubled sequences for modular diagnosability. After using the reduction rules (1)-(7), there is still no pair of troubled sequences. The reduced LPN model is modularly diagnosable.

Therefore, By using reduction rules (1)-(7) in Section 3.2.1 under Constraint 1, the modular diagnosability of the reduced LPN model keeps consistent with the modular diagnosability of the initial LPN model. □

According to Theorem 8, we can apply reduction rules to reduce the initial LPN model before decomposing the model and analyzing the modular diagnosability. The Theorem 8 guarantees that after using these rules, the modular diagnosability property is preserved.

Example 50 *Let us consider again the LPN model in Figure 4.9. By using the reduction rules (1), the unobservable transitions $\varepsilon_{1,5}$ and $\varepsilon_{2,5}$ are suppressed. By using the reduction rule (6), the transitions $t_{1,3}$, $t_{1,6}$, $t_{2,3}$ and $t_{2,4}$ are suppressed. The reduced model LPN' is shown in Figure 4.10. LPN' can be decomposed as LPN'_1 and LPN'_2, which is a sound decomposition. To analyze the modular diagnosability of the LPN model in Figure 4.9, it is sufficient to analyze the modular diagnosability of the reduced LPN model LPN'.*

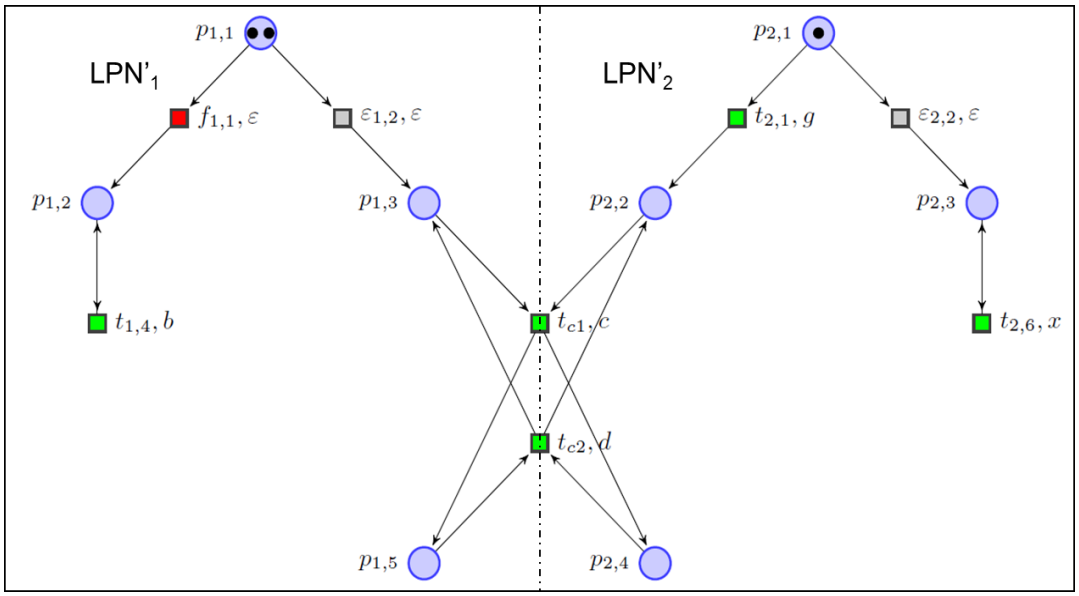


Figure 4.10 – The reduced LPN model LPN' (LPN) and its sound decomposition: LPN'_1 (LPN_1) and LPN'_2 (LPN_2)

Without abuse of notations, in the following section, when we analyze the reduced LPN model, we denote the reduced LPN model as LPN and its modules as LPN_1 and LPN_2.

4.2.3 Local diagnosability analysis

In this section, we propose an approach based on the Verifier Net (VN) approach.

To analyze the modular diagnosability of a given LPN model, the first step is to get the reduced LPN model by using the reduction rules. Once the reduced LPN is obtained, the LPN is decomposed as a sound decomposition. Afterwards, it is necessary to analyze the local diagnosability of each module. According to Proposition 10, if each module is locally diagnosable, the whole system is considered modularly diagnosable.

The local diagnosability of an LPN module is given as follows:

Definition 38 *(Local diagnosability) Given the LPN model LPN. Let $H := \{1, 2, \cdots, m\}$ and $LPN_1, \cdots, LPN_m$ a sound decomposition of LPN. For $z \in H$, $LPN_z = (N_z, M_{0z}, \Sigma_z, \mathcal{L}_z)$, where*

$N_z = (P_z, T_z, pre_z, post_z)$. LPN_z is locally diagnosable w.r.t. the fault class T_{fz} if there are not two sequences $\sigma_{z1}, \sigma_{z2} \in T_z^*$, which satisfy the following conditions:

1. $\mathcal{L}_z(\sigma_{z1}) = \mathcal{L}_z(\sigma_{z2})$;
2. $\forall t_f \in T_{fz}, t_f \notin \sigma_{z1}$;
3. $\exists t_f \in T_{fz}$ *such that* $t_f \in \sigma_{z2}$ *and* σ_{z2} *can be arbitrarily long after the occurrence of* t_f.

It is worth noticing that the definition of Local Diagnosability w.r.t. a module LPN_z is similar to Definition 8 by considering the monolithic LPN model as an LPN module, except that there can be unobservable cycles in Definition 38. If the LPN model contains only the unobservable cycle that is added to deal with the deadlock, most techniques for monolithic diagnosability presented in Chapter 3 are valid for local diagnosability. However, if the system contains initially unobservable cycles, the diagnoser-based techniques [Cab+14; Liu+14; Sam+95] cannot be applied for local diagnosaiblity analysis. In this case, the verifier-based approach [Cab+12; Jia+01; YL02] can be applied.

The VN of the considered LPN module is built, but a modified reachability graph (MRG) is built for the local diagnosability analysis instead of the reachability graph (RG) (it is assumed that the LPN model is bounded). Moreover, the MRG will be used for modular diagnosability analysis in the following section.

Before giving the algorithm of building the MRG, let us give some basic notions:

For an LPN module, $LPN_z = (N_z, M_{0z}, \Sigma_z, \mathcal{L}_z)$ where $N_z = (P_z, T_z, pre_z, post_z)$ and $T_z = T_{oz} \dot{\cup} T_{regz} \dot{\cup} T_{fz}$. The $T' - induced\ sub - LPN$ of LPN_z is denoted by $LPN'_z = (N'_z, M'_{0z}, \Sigma'_z, \mathcal{L}'_z)$, where $N'_z = (P'_z, T'_z, pre'_z, post'_z)$ and $T'_z = T_z \backslash T_{fz} = T_{oz} \dot{\cup} T_{regz}$. We denote $T'_z = T'_{oz} \dot{\cup} T'_{regz}$.

The VN of a local LPN module LPN_z is denoted as $\widetilde{LPN}_z = (\widetilde{N}_z, \widetilde{M}_{oz}, \widetilde{\Sigma}_{oz}, \widetilde{\mathcal{L}}_z)$, where $\widetilde{N}_z = (\widetilde{P}_z, \widetilde{T}_z, \widetilde{Pre}_z, \widetilde{Post}_z)$. $\widetilde{T}_z = \widetilde{T}_{oz} \dot{\cup} \widetilde{T}'_{regz} \dot{\cup} \widetilde{T}_{regz} \dot{\cup} \widetilde{T}_{fz} = \widetilde{T}_{oz} \dot{\cup} (T'_{regz} \times \{\lambda\}) \dot{\cup} (\{\lambda\} \times T_{regz}) \dot{\cup} (\{\lambda\} \times T_{fz})$ and $\widetilde{T}_{oz} = \{(t', t) \mid t' \in T'_{oz}, t \in T_{oz}, \mathcal{L}'_z(t') = \mathcal{L}_z(t)\}$. The incidence matrix is $\widetilde{C}_z$.

The state in MRG is the marking of VN associated with a fault tag, which is called an MFM (Modified Fault Marking) and is defined as follows:

Definition 39 *An MFM upon a sequence $\widetilde{\sigma} \in \widetilde{T}_z^*$ is*

$$MFM_z = \begin{bmatrix} Mark(MFM_z) \\ Tag(MFM_z) \end{bmatrix}$$

$Mark(MFM_z)$ is a marking of $\widetilde{LPN}_z$ and $Tag(MFM_z) \in \{N, F\}$, where $\widetilde{M}_{0z}\ [\ \widetilde{\sigma}\ > Mark(MFM_z)$ and $Tag(MFM_z) = F$ if $\exists t_f \in \widetilde{T}_{fz}, t_f \in \widetilde{\sigma}$; otherwise, $Tag(MFM_z) = N$.

Given two MFMs MFM_1 and MFM_2, it is denoted that $MFM_1 \,[\, \widetilde{\sigma} > MFM_2$ iff $Mark(MFM_1) \,[\, \widetilde{\sigma} > Mark(MFM_2)$; and $Tag(MFM_1) = Tag(MFM_2)$ if $\forall j \in \mathcal{N}, \widetilde{\sigma}^j \notin \widetilde{T}_{fz}$, otherwise, $Tag(MFM_2) = F$.

From the definition of MFM, an MFM consists of a marking of the VN and a binary tag indicating the occurrence of fault. For a given VN, the number of MFMs is at most twice of the number of nodes in the RG of the VN.

Definition 40 *The MRG_z w.r.t. the VN $\widetilde{LPN}_z$ is a tuple $(X_{MRGz}, \Sigma_{MRGz}, \delta_{MRGz}, MFM_{0z})$, where*

- *X_{MRGz} is the set of MFMs;*
- *Σ_{MRGz} is a finite set of events. It is worth noticing that the event in Σ_{MRG_z} is actually a transition in $\widetilde{T}_z$;*
- *$\delta_{MRGz}: X_{MRGz} \times \Sigma_{MRGz} \rightarrow X_{MRGz}$ is the transition function: $\delta_{MRGz}(MFM_{z1}, \widetilde{t}) = \{MFM_{z2} | MFM_{z1} \,[\, \widetilde{t} > MFM_{z2}\}$;*
- *$MFM_{0z} = [\widetilde{M}_{0z}^{\tau}, N]^{\tau}$ is the initial node.*

Hereafter, without abuse of notations, the "transition" of an automaton structure (e.g. MRG) is denoted as "transition*". For instance, the transition* in MRG is labeled by the transition of VN.

The algorithm of building MRG_z w.r.t. the VN $\widetilde{LPN}_z$ is as follows:

Definition 41 *A cycle in MRG of the VN $\widetilde{LPN}_z$ is called un $F-$*confused cycle*, if the following conditions are satisfied:*

1. *The tags of all the nodes in this cycle are F;*
2. *At least one transition of this cycle belong to $\widetilde{T}_{oz} \dot{\cup} \widetilde{T}_{regz} \dot{\cup} \widetilde{T}_{fz}$.*

The sufficient and necessary condition of local diagnosability is given as follows

Theorem 9 *An LPN module LPN_z is locally diagnosable, iff in the MRG of the VN $\widetilde{LPN}_z$, there does not exist any $F-$confused cycle.*

Proof: The proof of this theorem is similar to the proof of Theorem 2. Since we assume that the LPN model is bounded, the cycle in MRG is associated with a firable repetitive sequence in the VN and we do not need to check if it is associated with a repetitive sequence. Verifying the existence of cycles in the RG after firing a fault transition is

Algorithm 13 Algorithm for building MRG_z

1. Step 1: Label the root node q_0 with initial node MFM_{0z} and mark it NEW.
2. Step 2: **While** a node marked NEW exists **do**
   a) Step 2.1: Select a node q marked NEW and let MFM_z be its label;
   b) Step 2.2: **If** $(Tag(MFM_z) = N) \wedge (\forall t_f \in \widetilde{T}_{fz}, P(Mark(MFM_z), t_f, t_f, FAULT) = \emptyset)$ (the P function is presented in Algorithm 8 (page 88))
   Continue;
   Else
      i. Step 2.2.1: **For** all $\widetilde{t} \in \widetilde{T}_z$ enabled at $Mark(MFM_z)$, i.e., such that $Mark(MFM_z) \geq \widetilde{Pre}_z(\cdot, t)$:
         A. Step 2.2.1.1: Let $MFM_z[\widetilde{t} > MFM'_z$, where $Mark(MFM'_z) = Mark(MFM_z) + \widetilde{C}_z(\cdot, \widetilde{t})$ and $Tag(MFM'_z) = Tag(MFM_z)$ if $\widetilde{t} \notin \widetilde{T}_{fz}$, otherwise, $Tag(MFM_2) = F$;
         B. Step 2.2.1.2: Add a new node q' and label it MFM'_z;
         C. Step 2.2.1.3: add an arc labeled $\widetilde{t}$ (or $(\widetilde{t}, \widetilde{\mathcal{L}}_z(\widetilde{t}))$) from q to q';
         D. Step 2.2.1.4: **If** there already exists a node with label MFM'_z, **then** fusing this node with q'; **else** mark it NEW;
   c) Step 2.3: Unmark node q.

equivalent to verifying the existence of cycles in the MRG, whose nodes are all with a fault tag F.

Since it is assumed that there can be unobservable cycles, the Condition 2 of Definition 41 is added. If there exist a cycle whose nodes are all with a fault tag F but all the transitions of this cycle belong to $\widetilde{T}'_{regz}$, i.e., all the transitions of this cycle belong to the $T' - induced\ sub - LPN\ LPN'_z$, the existence of such a cycle does not violate the modular diagnosability property in Definition 38. LPN'_z does not contain any fault transition. The existence of such a cycle does not satisfy the condition 2 of Definition 38, i.e., it does not imply that after the occurrence of a fault transition, the sequence can be arbitrarily long. □

The Algorithm 13 is based on the algorithm of construction of RG. It is worth noticing that the condition in Step 2.2 is a stop condition to stop the investigation of a branch. For a selected node labeled MFM_j with the fault tag $Tag(MFM_j) = N$ and $\forall t_f \in \widetilde{T}_{fz}$, $P(Mark(MFM_j), t_f, t_f, FAULT) = \emptyset$) implies that there is no path to fire any fault transition at $Mark(MFM_j)$ and all the following nodes are normal nodes. According to Theorem 9, checking the local diagnosability consists of verifying the existence of the $F-$confused cycle whose nodes are with a tag F. Therefore, there is no need to continue the construction of the following normal nodes.

To verify the local diagnosability of an LPN module, we build at first its VN. Afterwards, we build its MRG instead of RG. The reasons are:

1. A cycle in RG of the VN can be reached at the same time by firing a sequence with a fault transitions and by firing a sequence without any fault transition. To verify the modular diagnosability, we need to focus on the sequence with a fault transition and the cycle that is reached by firing this sequence, i.e., the $F-$confused cycle in MRG.

2. Each node in MRG is associated with a fault tag. We use this tag as an indicator for modular diagnosability.

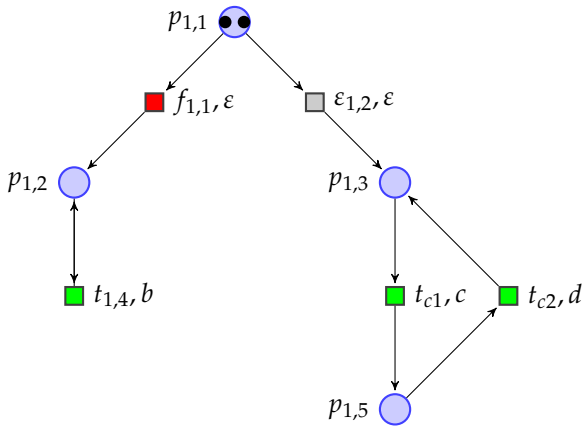


Figure 4.11 – The module LPN_1 of the LPN model in Figure 4.10

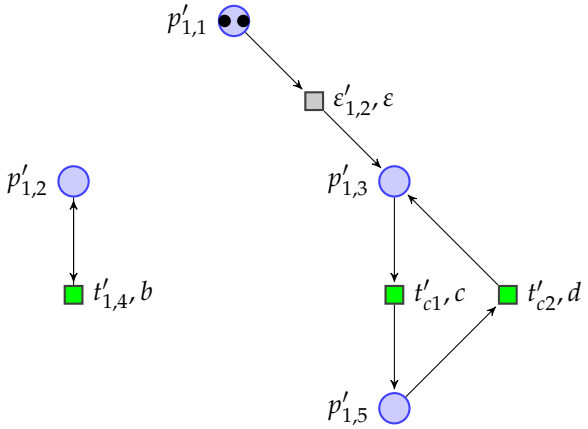


Figure 4.12 – The $T' - induced\ sub - LPN$ of LPN_1

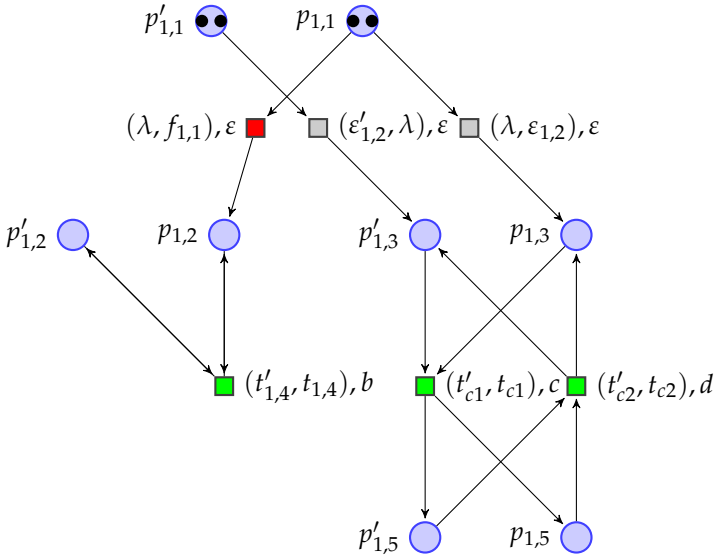


Figure 4.13 – The VN $\widetilde{LPN_1}$ of LPN_1

Table 4.2 – The MFMs in Figure 4.14

j	MFM_j	j	MFM_j
0	$[2\,0\,0\,0\,\|2\,0\,0\,0,\ N]^\tau$	10	$[1\,0\,1\,0\,\|0\,2\,0\,0,\ F]^\tau$
1	$[2\,0\,0\,0\,\|1\,1\,0\,0,\ F]^\tau$	11	$[1\,0\,1\,0\,\|0\,1\,1\,0,\ F]^\tau$
2	$[2\,0\,0\,0\,\|1\,0\,1\,0,\ N]^\tau$	12	$[0\,0\,2\,0\,\|1\,1\,0\,0,\ F]^\tau$
3	$[1\,0\,1\,0\,\|2\,0\,0\,0,\ N]^\tau$	13	$[1\,0\,1\,0\,\|0\,0\,2\,0,\ N]^\tau$
4	$[2\,0\,0\,0\,\|0\,2\,0\,0,\ F]^\tau$	14	$[0\,0\,2\,0\,\|1\,0\,1\,0,\ N]^\tau$
5	$[2\,0\,0\,0\,\|0\,1\,1\,0,\ F]^\tau$	15	$[0\,0\,2\,0\,\|0\,2\,0\,0,\ F]^\tau$
6	$[1\,0\,1\,0\,\|1\,1\,0\,0,\ F]^\tau$	16	$[1\,0\,0\,1\,\|0\,1\,0\,1,\ F]^\tau$
7	$[2\,0\,0\,0\,\|0\,0\,2\,0,\ N]^\tau$	17	$[0\,0\,2\,0\,\|0\,1\,1\,0,\ F]^\tau$
8	$[1\,0\,1\,0\,\|1\,1\,0\,0,\ N]^\tau$	18	$[0\,0\,2\,0\,\|0\,0\,2\,0,\ N]^\tau$
9	$[0\,0\,2\,0\,\|2\,0\,0\,0,\ N]^\tau$	19	$[0\,0\,1\,1\,\|0\,1\,0\,1,\ F]^\tau$

Example 51 *Let us consider the LPN model in Figure 4.10. Since only LPN_1 contains a fault, it needs only to analyze the local diagnosability of LPN_1 that is shown in Figure 4.11. The $T'-$induced sub $-$ LPN of LPN_1 is shown in Figure 4.12. The VN $\widetilde{LPN_1}$ is shown in Figure 4.13. Afterwards, we build the MRG_1 of $\widetilde{LPN_1}$ (shown in Figure 4.14) and the MFMs are shown in Table 4.2. LPN_1 is not locally diagnosable, because there exist two $F-$confused cycles shown with the shadow zone.*

The investigation of the branches after MFM_7, MFM_{13} and MFM_{18} are stopped, because the tag of these MFM is N and after firing two times the transition $(\lambda, \varepsilon_{1,2}), \varepsilon$, the fault transition $(\lambda, f_{1,1}), \varepsilon$ is never enabled. There is no need to continue the construction of these branches.

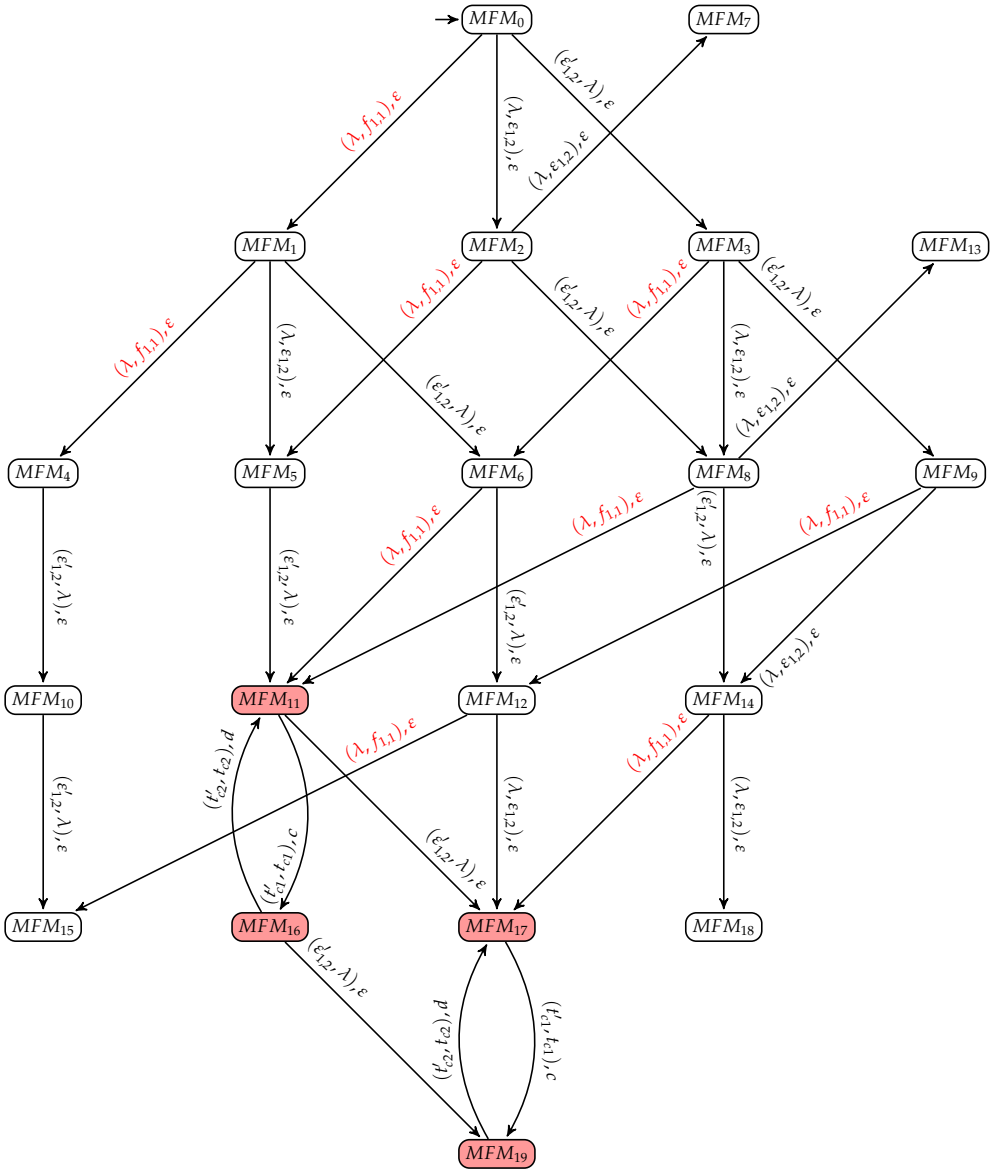


Figure 4.14 – MRG_1 of $\widetilde{LPN_1}$

The MRG contains 20 nodes but if we build the RG of $\widetilde{LPN_1}$, it contains 25 nodes. Theoretically, the number of nodes in MRG is twice the number of nodes in RG. However, the MRG can have fewer nodes than RG for the VN of some LPN model (as it is shown in Example 51).

In terms of the complexity, it can be proved that the complexity of building MRG is equal to that of build RG but the proof is omitted.

4.2.4 Incremental modular diagnosability analysis

This section introduces a new approach for modular diagnosability analysis using LPN. This approach is based on an incremental algorithm. The objectives are to develop an approach with a low computational complexity and to extend the proposition in [Con+06].

Indeed, by removing the assumption of liveness in [Con+06], the scope of use for modular diagnosability analysis is less limited.

After analyzing the local diagnosability of each module, if all modules are locally diagnosable, the system is modularly diagnosable. Otherwise, we concentrate on the modules that are not locally diagnosable to find out if the modular diagnosability property is verified or not, when these modules are coupled with the rest of the system.

Assuming that the module LPN_z ($z \in H$ and $H = \{1, 2, \cdots, m\}$) is not locally diagnosable i.e., there exist $F-$confused cycles in the MRG of the VN $\widetilde{LPN_z}$. Afterwards, we need to check these $F-$confused cycle to determine if they still survive while coupling with the other modules.

Similar to the approach in [MP13], by using LPN model, we can build the VN of $LPN_z||LPN_{S_c}$ ($S_c \subseteq H\backslash\{z\}$), then build its RG or MRG to check the modular diagnosability.

However, in this section, we will not build the VN of $LPN_z||LPN_{S_c}$ and its RG, but we will take advantage of the already constructed MRG of $\widetilde{LPN_z}$.

For the module LPN_z, its VN is $\widetilde{LPN_z}$ and the MRG is MRG_z. Assuming that LPN_z is not locally diagnosable, i.e., there exist an $F-$confused cycle. We mark all the nodes that belong to the $F-$confused cycles in order to build $CoAc(MRG_z)$ (we only need to focus on the $F-$confused cycles, so it is reasonable to keep only the states from which the $F-$confused cycles can be reached. We will illustrate it afterwards). For the composed module LPN_{S_c} ($S_c \subseteq H\backslash\{z\}$), we build the RG of LPN_{S_c} that is denoted as RG_{S_c}. Afterwards, we build the parallel composition $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$. If for all $S_c \subseteq H\backslash\{z\}$, the $F-$confused cycle survives, the system is not modularly diagnosable. Otherwise, the composed module built by $LPN_z||LPN_{S_c}$ is modularly diagnosable. Before proposing the algorithm of our approach, we will prove the correctness of our approach and give the sufficient and necessary condition of modular diagnosability.

Remark: The parallel composition $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$ is made on the set of common observable labels of the transitions in $\widetilde{LPN_z}$ and LPN_S. For example, $(t'_{c1}, t_{c1}), c$ is a label of the transition* in $CoAc(MRG_z)$ and t_{c1}, c is a label of a transition* in RG_{S_c}. They should be synchronized, because the label of the two transitions is the same. For the simplicity of notations, we use the label of transition* in $CoAc(MRG_z)$ as the label of synchronized transition* in $RG_{z||S_c}$. $\widetilde{T}_{cz} = \{\widetilde{t} \in \widetilde{T}_z | \exists t \in T_{S_c}, \widetilde{\mathcal{L}}_z(\widetilde{t}) = \mathcal{L}_{S_c}(t)\}$ is defined as the set of common transitions.

Let us consider the module LPN_z ($z \in H$ and $H = \{1, 2, \cdots, m\}$). Assuming that LPN_z is not locally diagnosable, i.e., there exist $F-$confused cycles in the MRG_z of the VN $\widetilde{LPN_z}$.

Definition 42 *$F^{M_z}-$confused cycle in $RG_{z||S_c}$*

Let us denote $RG_{z||S_c} = (X_{z||S_c}, \Sigma_{z||S_c}, \delta_{z||S_c}, x_{z||S_c})$. *For* $x_{z||S_c} \in X_{z||S_c}$, $x_{z||S_c} = (MFM_z, x_{S_c})$, *where* MFM_z *is a state in* MRG_z *and* $x_{S_c} = (x_{i^1}, \cdots, x_{i^q})$ $(i^1, \cdots, i^q \in S$ *and* q *is the cardinality of* $S)$ *is a state in* RG_{S_c}.

The cycle in $RG_{z||S_c}$

$$cl := (x^k_{z||S_c}, t_k, x^{k+1}_{z||S_c}, \cdots, x^h_{z||S_c}, t_h, x^k_{z||S_c})(h \geq k > 0)$$

where for all $j \in \{1, \cdots, h\}$, $x^{k+j}_{z||S_c} = \delta_{z||S_c}(x^{k+j-1}_{z||S_c}, t_j)$ *and* $x^k_{z||S_c} = \delta_{z||S_c}(x^h_{z||S_c}, t_h)$, *is called an* $F^{M_z}-$*confused cycle, if the following conditions are satisfied:*

1. *For each state* $x^r_{z||S_c}$ $(r \in \{k, k+1, \cdots, h\})$ *in the cycle,* $Tag(MFM^r_z) = F$;
2. $\exists t_p \in \widetilde{T}_{oz} \cup \widetilde{T}_{regz} \cup \widetilde{T}_{fz}$, *where* $p \in \{k, k+1, \cdots, h\}$;

The $F^{M_z}-$confused cycle (the symbol "M_z" stands for "Module LPN_z") requires that each state contains the tag F w.r.t. T_{fz} and at least one transition* of the cycle labeled by a transition in $\widetilde{T}_{oz} \cup \widetilde{T}_{regz} \cup \widetilde{T}_{fz}$.

Theorem 10 *Let* $H := \{1, 2, \cdots, m\}$, $S \subseteq H$, $LPN_S := \|_{j \in S}\ G_j$. *The language* $L(LPN_S)$ *is modularly diagnosable w.r.t.* $(\Sigma_{oj}: j \in S)$ *and* T_{fz} $(z \in S)$, *if and only if* $\forall S_c \subseteq S \backslash \{z\}$, *there is no* $F^{M_z}-$*confused cycle in* $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$.

Proof: (Necessity) We prove by contradiction. Assuming that there exists an $F^{M_z}-$confused cycle in $RG_{z||S_c}$. Denote $x_{z||S_c} = (MFM_z, x_{S_c})$ the state in the $F^{M_z}-$confused cycle, where $Tag(MFM_z) = F$. It can be deduced that there exist two firing sequences of transitions in LPN_S σ_1 and σ_2, such that:

1. $\mathcal{L}_S(\sigma_1) = \mathcal{L}_S(\sigma_2)$;
2. $\forall t_f \in T_{fz}$, $t_f \notin \sigma_1$; $\exists t_f \in T_{fz}$ such that $t_f \in \sigma_2$ and σ_2 can be arbitrarily long after the occurrence of fault.

Moreover, in the $F^{M_z}-$confused cycle, at least one transition belongs to $\widetilde{T}_{oz} \cup \widetilde{T}_{regz} \cup \widetilde{T}_{fz}$. Denote $\sigma_{1z} = P_{\{T_S, T_z\}}(\sigma_1)$ and $\sigma_{2z} = P_{\{T_S, T_z\}}(\sigma_2)$. σ_{1z} and σ_{2z} satisfy the following conditions:

1. Condition 1: $\mathcal{L}_z(\sigma_{1z}) = \mathcal{L}_z(\sigma_{2z})$;
2. Condition 2: σ_{1z} does not contain any fault in T_{fz}; σ_{2z} contains a fault transition in T_{fz} and can be arbitrarily long after the fault transition.

Because σ_{1z} and σ_{2z} cannot be distinguished in a finite number of transitions in T_z, neither σ_1 nor σ_2 can be distinguished in a finite number of transitions in T_z. Therefore, the definition of modular diagnosability (in Definition 37) is violated and $L(LPN_S)$ is not modularly diagnosable.

Remark: It is possible that a cycle exists in $RG_{z||S_c}$, and each state in this cycle denoted as $x_{z||S_c} = (MFM_z, x_{S_c})$, where $Tag(MFM_z) = F$. For each transition* $\widetilde{t}$ in this cycle, $\widetilde{t} \notin \widetilde{T}_{oz} \cup \widetilde{T}_{regz} \cup \widetilde{T}_{fz}$ but there exist $\widetilde{t}'$ in this cycle s.t. $\widetilde{t}' \in \widetilde{T}'_{regz}$. The existence of this cycle does not violate the definition of modular diagnosability in Definition 37. This cycle implies that there exist two sequences s.t. the Condition 1 is satisfied but the Condition 2 is not satisfied. It can not be deduced that the faulty sequence can be arbitrarily long after the occurrence of fault transition, since the original transition of $\widetilde{t}'$ belongs to T'_{reg} and the $T' - induced\ sub - LPN\ LPN'_z$ does not contain any fault transition. The two sequences may be distinguished after the occurrence of transitions in T_z following the sequence that contains a fault.

(Sufficiency) Assuming that there is no $F^{M_z}-$confused cycle in $RG_{z||S_c}$. We have three cases to consider:

1. there is no cycle in $RG_{z||S_c}$;
2. there is a cycle in $RG_{z||S_c}$, but there exists a state in this cycle $x_{z||S_c} = (MFM_z, x_{S_c})$, where $Tag(MFM_z) = N$;
3. there is a cycle in $RG_{z||S_c}$, and each state in this cycle denoted as $x_{z||S_c} = (MFM_z, x_{S_c})$, where $Tag(MFM_z) = F$. However, for each transition* labeled t in this cycle, $t \notin \widetilde{T}_{oz} \cup \widetilde{T}_{regz} \cup \widetilde{T}_{fz}$;

Case (1): Each sequence that contains a fault can be distinguished in a finite number of transitions in T_z. Therefore, $L(LPN_S)$ is modularly diagnosable.

Case (2): If there exists a state in the cycle that the coordinate $Tag(MFM_z)$ of the state $x_{z||S_c}$ is equal to N, it can be deduced that the coordinate $Tag(MFM_z)$ of all the other states in this cycle is also N. This cycle indicates two sequences that do not contain any fault transition. The modular diagnosability is not violated.

Case (3): There exists a cycle and the coordinate $Tag(MFM_z)$ of all the states in this cycle is F. However, there is no label of transition* in this cycle that belongs to $\widetilde{T}_{oz} \cup \widetilde{T}_{regz} \cup \widetilde{T}_{fz}$. The faulty sequences can alway be distinguished after finite number of labels following the fault transition in T_{fz}. It is worth noticing that if there exist a transition $\widetilde{t}' \in \widetilde{T}'_{regz}$ in this cycle, it does not violate the modular diagnosability because the original transition of $\widetilde{t}'$ belongs to T'_{reg} and the $T' - induced\ sub - LPN\ LPN'_z$ does not contain any fault transition.. According to Definition 33, $L(LPN_S)$ is modularly diagnosable. □

The Theorem 10 proposes a sufficient and necessary condition of the modular diagnosabiliy of the system. One can check the modular diagnosability of a system by verifying the existence of $F^{M_z}-$confused cycle w.r.t. T_{fz} ($z \in S$).

Remark: If there exists a cycle in MRG_z but not in $CoAc(MRG_z)$, it can be deduced that the tag of all the MFMs in this cycle is N. There is no $F^{M_z}-$confused cycle corresponding to this cycle according to the Condition (1) of Definition 42 even if we build $MRG_z||RG_{S_c}$. Therefore, we focus only on $CoAc(MRG_z)$ and we build $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$ to check the modular diagnosability in order to reduce combinatorial explosion problem.

Proposition 12 *Let $H := \{1, 2, \cdots, m\}$, $S \subset H$, $LPN_S := \|_{z \in S}\ LPNz$. If the language $L(LPN_S)$ is modularly diagnosable w.r.t. (Σ_{oj}: $j \in S$) and T_{fz} ($z \in S$), then the language $L(LPN_H)$ is modularly diagnosable w.r.t. (Σ_{oj}: $j \in H$) and T_{fz}.*

Proof: We prove the contrapositive: if language $L(G_H)$ is not modularly diagnosable w.r.t. (Σ_{oj}: $j \in H$) and T_{fz} ($z \in I$), then the language $L(LPN_S)$ is not modularly diagnosable w.r.t. (Σ_{oj}: $j \in S$) and T_{fz} ($z \in S$).

If language $L(LPN_H)$ is not modularly diagnosable w.r.t. (Σ_{oj}: $j \in H$) and T_{fz} ($z \in S$), there exist an $F^{M_z}-$confused cycle in $RG_{z||H_c} = CoAc(MRG_z)||RG_{H_c}$, where $H_c = H \backslash \{z\}$. It can be deduced that there exist an $F^{M_z}-$confused cycle in $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$, where $S_c = S \backslash \{z\}$, because $RG_{z||H_c}$ can be obtained by $(RG_{z||S_c})||_{j \in H_c \backslash S_c} RG_j$. □

According to Proposition 12, in order to check the modular diagnosability G_H ($H := \{1, 2, \cdots, m\}$) w.r.t. (Σ_{oj}: $j \in H$) and T_{fz} ($z \in H$), it is not necessary to build directly the parallel composition of $CoAc(MRG_z)$ with all the RGs of other modules in $H \backslash \{z\}$. If there exists $S \subset H$ and $z \in S$ s.t. there is no $F^{M_z}-$confused cycle in $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$ ($S_c = S \backslash \{z\}$), i.e., $L(LPN_S)$ is modularly diagnosable, then it can be deduced that $L(LPN_H)$ is modularly diagnosable.

Based on Theorem 10 and Proposition 12, we develop Algorithm 14 for modular diagnosability analysis as follows:

Remark: While building RG_{S_c} ($S_c \subseteq H \backslash \{z\}$), one can build directly the RG of LPN_{S_c} or build $RG_{S_c} = \|_{j \in S_c}\ RG_j$ as it is presented in Step 3.2.1. We choose the latter because if $RG_{z||S'_c} = CoAc(MRG_z)||RG_{S'_c}$ ($S'_c \subset S_c$) is already built, we can obtain $RG_{z||S_c}$ by building $RG_{z||S_c} = (RG_{z||S'_c})||RG_{S_c \backslash S'_c}$, where $RG_{S_c \backslash S'_c} = \|_{j \in S_c \backslash S'_c}\ RG_j$. It implies that the proposed algorithm is incremental.

Remark: If there are several fault classes in module LPN_z, it needs to build one VN and one MRG for each fault class. Then, we analyze the modular diagnosability by using the MRGs w.r.t. different fault classes.

Algorithm 14 Modular diagnosability algorithm

1. Step 1: Initialize the modular diagnosability verdict vector $\overrightarrow{v} := \{0\}^{|H|}$, where $H := \{1, 2, \cdots, m\}$ and component v_z of $\overrightarrow{v}$ is the verdict w.r.t. the fault in module LPN_z (T_{fz});
2. Step 2: For $H := \{1, 2, \cdots, m\}$, analyze the local diagnosability of each module LPN_z ($z \in H$) by using Algorithm 13 and if LPN_z is locally diagnosable, set v_z as 1;
3. Step 3: For all $z \in H$ such that LPN_z is not locally diagnosable;
   - Step 3.1: Mark all the states that belongs to the $F-$confused cycles of the MRG_z and build $CoAc(MRG_z)$;
   - Step 3.2: For all $S_c \subseteq H \backslash \{z\}$, s.t. LPN_S has shared transitions with LPN_z:
     i. Step 3.2.1: Build $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$, where $RG_{S_c} = \|_{j \in S_c} RG_j$;
     ii. Step 3.2.2: if there does not exist an F^{M_z}-confused cycle in $RG_{z||S_c}$:
     • Set v_z as 1 and *break*; (if there does not exist an F^{M_z}-confused cycle in $RG_{z||S_c}$, there is no need to continue the iteration of the Step 3.2 and the result is given that the system is modular diagnosable w.r.t. (Σ_{oj}: $j \in H$) and T_{fz}.)
4. Step 4: For all $z \in H$:
   • If $v_z = 0$, assert "The system is not modular diagnosable w.r.t. (Σ_{oj}: $j \in H$) and T_{fz}";
   • else if $v_z = 1$, assert "The system is modular diagnosable w.r.t. (Σ_{oj}: $j \in H$) and T_{fz}";

Example 52 *Let us consider again the LPN model in Figure 4.10. We only need to analyze the local diagosability of the module LPN_1, because the module LPN_2 does not contain any fault transition. The local diagnosability of LPN_1 is analyzed for Example 51. We mark all the nodes in $F-$confused cycle of MRG_1 and build the $CoAc(MRG_1)$ that is shown in Figure 4.15. The module LPN_2 and its RG RG_2 are shown in Figure 4.16 and Figure 4.17. The markings are shown in Table 4.3. Afterwards, $RG_{1||2} = CoAc(MRG_1)||RG_2$ is built, which contains 38 nodes. We will not show the graph of $RG_{1||2}$ is not illustrated because of its large scale. In $RG_{1||2}$, we can find two cycles*

$$\{MFM_{11}, M_1\} \xrightarrow{c} \{MFM_{16}, M_3\} \xrightarrow{d} \{MFM_{11}, M_1\} \xrightarrow{c} \cdots$$

$$\{MFM_{17}, M_1\} \xrightarrow{c} \{MFM_{19}, M_3\} \xrightarrow{d} \{MFM_{17}, M_1\} \xrightarrow{c} \cdots$$

The two cycles are both F^{M_1}-confused cycles according to Definition 42. Therefore, the system LPN_H ($H = \{1, 2\}$) is not modularly diagnosable.

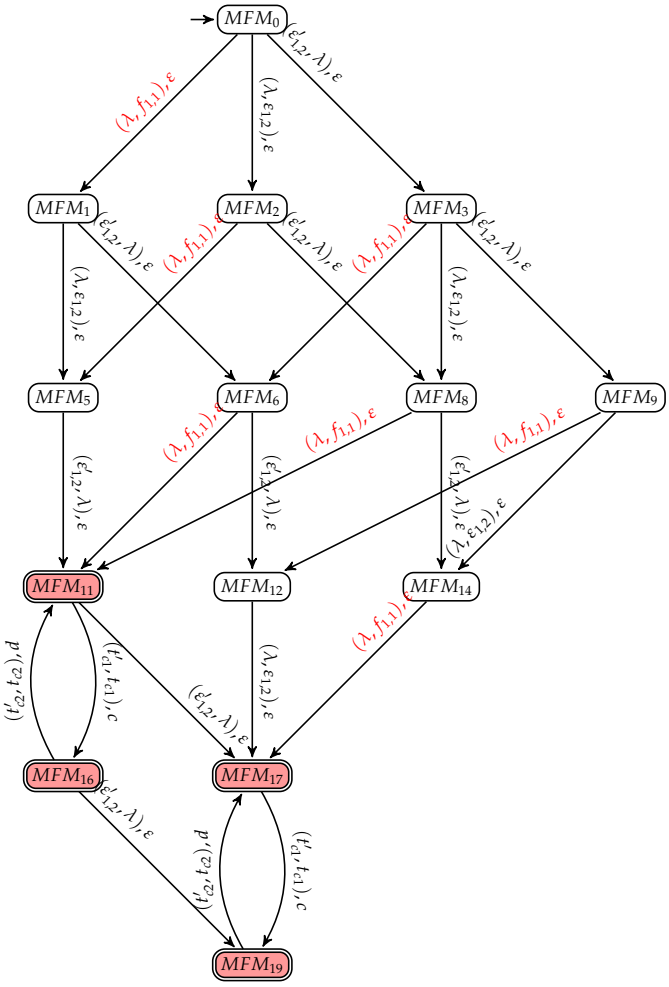


Figure 4.15 – $CoAc(MRG_1)$

Table 4.3 – The markings in Figure 4.17

j	M_j
0	$[1\,0\,0\,0\,]^\tau$
1	$[0\,1\,0\,0\,]^\tau$
2	$[0\,0\,1\,0\,]^\tau$
3	$[0\,0\,0\,1\,]^\tau$

4.2.5 $\varepsilon-$reduction technique to combat combinatorial explosion for modular diagnosability analysis

To analyze the local diagnosability of each module, we use the VN and MRG in order to reduce the computational complexity. Afterwards, we build the parallel composition of MRG_z and RG_{S_c} of a composed module. However, from Example 52, it can be noticed that the combinatorial explosion problem exists (especially when we build the parallel composition). Therefore, it is necessary to propose a method to reduce the combinatorial

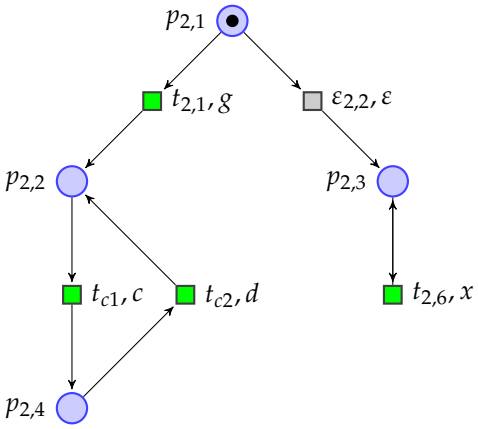


Figure 4.16 – LPN_2

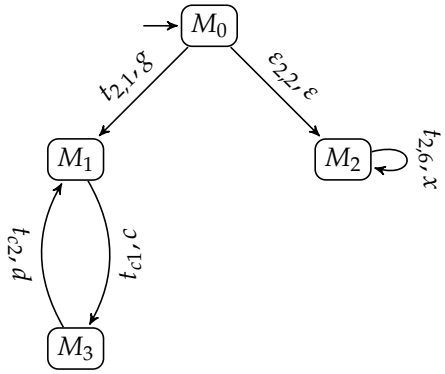


Figure 4.17 – RG of LPN_2 (RG_2)

explosion.

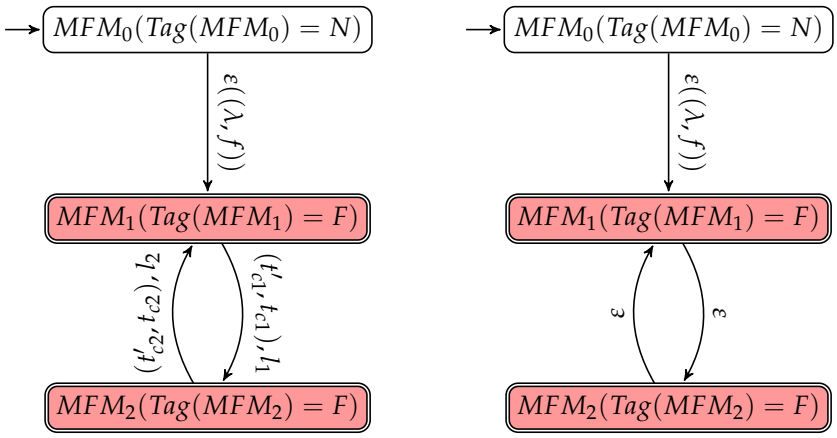
While building $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$ ($S_c \subseteq H\backslash\{z\}$), the $F^{M_z}-$confused cycle and the common transitions between LPN_z and LPN_{S_c} (i.e., common event between $CoAc(MRG_z)$ and RG_{S_c}) are the most important information, because we need to verify if the $F-$confused cycle in MRG_z survives or not after the parallel composition. Therefore, we can use some techniques (such as $\varepsilon-$reduction) to reduce the sizes of MRG_z and RG_{S_c} before building the parallel composition in order to reduce the combinatorial explosion problem.

Proposition 13 *Given an LPN model LPN. $LPN_1, \cdots, LPN_m, H = \{1, \cdots, m\}$ is a sound decomposition of LPN. Assuming that the module $LPN_z, z \in H$ is not locally diagnosable w.r.t. a fault transition $t_f \in T_{fz}$. The MRG of the VN $\widetilde{LPN_z}$ is MRG_z. From the initial node of MRG_z, the sequence of transitions that leads to the F-confused cycle is $\widetilde{\sigma} = \widetilde{t}_{a,1} \cdots \widetilde{t}_{a,p}(\widetilde{t}_{b,1} \cdots \widetilde{t}_{b,q})^*$ (the sequence belong to $(\widetilde{T}_z)^*$). Assuming a composed module LPN_{S_c} ($S_c \subseteq H\backslash\{z\}$) has shared transitions with LPN_z and there exists an F^{M_z}-confused cycle in $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$ corresponding to the $F-$confused cycle.*

Let us consider all the transitions labeled by uncommon transitions in T_{S_c} as $\varepsilon-$transitions.*

$RG_{S_c^*}$ is obtained from RG_{S_c} by using the $\varepsilon-$reduction to reduce the $\varepsilon-$transitions. We build $RG_{z||S_c^*} = CoAc(MRG_z)||RG_{S_c^*}$. From the initial node of $RG_{z||S_c^*}$, the sequence of transitions that leads to the F-confused cycle is $\widetilde{\sigma} = \widetilde{t}_{a,1} \cdots \widetilde{t}_{a,p}(\widetilde{t}_{b,1} \cdots \widetilde{t}_{b,q})^*$.

Proof: If there exist an F^{M_z}-confused cycle in $RG_{z||S_c} = CoAc(MRG_z)||RG_{S_c}$, corresponding to this cycle, there exist an F^{M_z}-confused cycle in $RG_{z||S_c^*} = CoAc(MRG_z)||RG_{S_c^*}$. While building the parallel composition $RG_{z||S_c^*} = CoAc(MRG_z)||RG_{S_c^*}$, the synchronization is on the shared transitions that are preserved in $RG_{S_c^*}$. Therefore, it is intuitive that each F^{M_z}-confused cycle in $RG_{z||S_c^*}$ corresponds to an F^{M_z}-confused cycle in $RG_{z||S_c}$. Since only the shared transitions are preserved in $RG_{S_c^*}$ and all the $\varepsilon-$transitions are reduced, the sequence that leads to the F^{M_z}-confused cycle in $RG_{z||S_c^*}$ is equal to the sequence that leads to the F^{M_z}-confused cycle in $RG_{z||S_c}$. □



(b) Case 1 (b) Case 2

Figure 4.18 – Cases to avoid while applying $\varepsilon-$reduction

Similarly, the uncommon transitions* in $CoAc(MRG_z)$ can also be considered as $\varepsilon-$transitions* and be reduced by $\varepsilon-$reduction technique. While using the $\varepsilon-$reduction technique for modular diagnosability analysis, there are two cases that need to avoid.

Case 1 A node MFM_j of the F^{M_z}-confused cycle in $CoAc(MRG_z)$ ($Tag(MFM_j) = F$) is suppressed because of the reduction of an $\varepsilon-$transition* from a node MFM_k, where $Tag(MFM_k) = N$. In this case, we may find the F^{M_z}-confused cycle contains a node MFM_k with $Tag(MFM_k) = N$, which violate the definition of F^{M_z}-confused cycle. (As shown in Figure 4.18(a), if the ε transition* is reduced, the F^{M_z}-confused cycle will contain the node MFM_0 with $Tag(MFM_0) = N$.)

Case 2 In the F^{M_z}-confused cycle of $CoAc(MRG_z)$, there does not exist any transition* labeled by a common transition. The F^{M_z}-confused cycle may disappear and the verdict of modular diagnosability may not be correct. (As shown in Figure 4.18(b), if all the ε transitions* are reduced, the F^{M_z}-confused cycle will disappear.)

To ensure the correctness by using $\varepsilon-$reduction technique, two constraints are proposed as follows:

Constraint 1 An $\varepsilon-$transition* in $CoAc(MRG_z)$ will not be reduced, if it is from a node MFM_k with $Tag(MFM_k) = N$ to a node MFM_j with $Tag(MFM_j) = F$.

Constraint 2 If in the F^{M_z}-confused cycle of $CoAc(MRG_z)$, there does not exist any transition* labeled by a common transition, choose one transition* in the F^{M_z}-confused cycle and do not consider this transition* as $\varepsilon-$transition*.

Algorithm 15 Modular diagnosability algorithm using $\varepsilon-$reduction

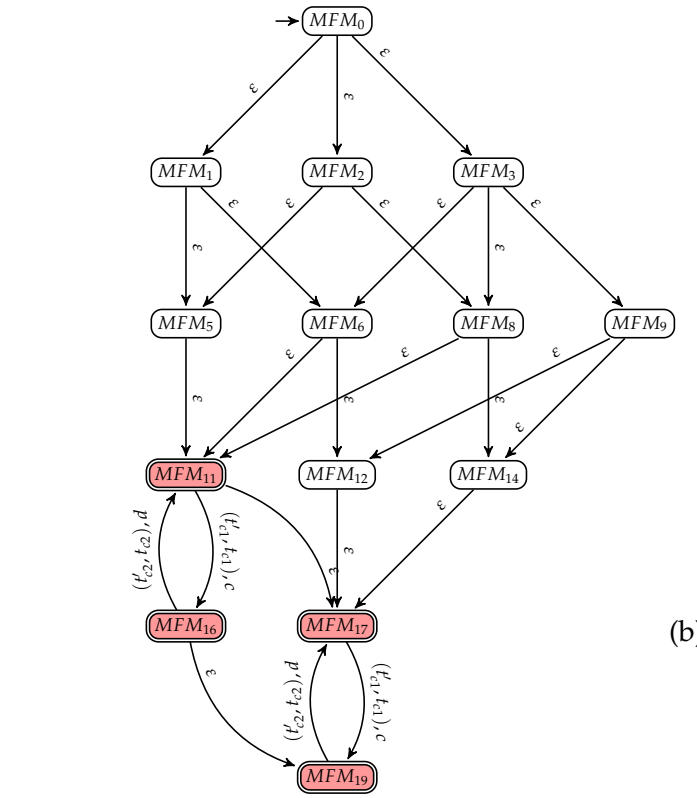
1. Step 1: Initialize the modular diagnosability verdict vector $\overrightarrow{v} := \{0\}^{|H|}$, where $H := \{1, 2, \cdots, m\}$ and component v_z of $\overrightarrow{v}$ is the verdict w.r.t. the fault in module LPN_z (T_{fz});
2. Step 2: For $H := \{1, 2, \cdots, m\}$, analyze the local diagnosability of each module LPN_z ($z \in H$) by using Algorithm 13 and if LPN_z is locally diagnosable, set v_z as 1;
3. Step 3: For all $z \in H$ such that LPN_z is not locally diagnosable;
   - Step 3.1: Mark all the states that belongs to the $F-$confused cycles of the MRG_z and build $CoAc(MRG_z)$;
   - Step 3.2: Rename all the uncommon transitions in $CoAc(MRG_z)$ as $\varepsilon-$transition*;
   - Step 3.3: Build $CoAc(MRG_{z^*})$ which is obtained from $CoAc(MRG_z)$ by using the $\varepsilon-$reduction to reduce the $\varepsilon-$transitions* w.r.t. the two constraints;
   - Step 3.4: For all $S_c \subseteq H \backslash \{z\}$, s.t. LPN_S has shared transitions with LPN_z:
     i. Step 3.4.1: Build $RG_{S_c} = \|_{j \in S_c} RG_j$;
     ii. Step 3.4.2: Rename all the uncommon transitions in RG_{S_c} as $\varepsilon-$transition*;
     iii. Step 3.4.3: Build $RG_{S_c^*}$ which is obtained from RG_{S_c} by using the $\varepsilon-$reduction to reduce the $\varepsilon-$transitions*;
     iv. Step 3.4.4: Build $RG_{z^*||S_c^*} = CoAc(MRG_{z^*})||RG_{S_c^*}$;
     v. Step 3.4.5: If there does not exist an F^{M_z}-confused cycle in $RG_{z^*||S_c^*}$:
     • Set v_z as 1 and *break*;
4. Step 4: For all $z \in H$:
   • If $v_z = 0$, assert "The system is not modular diagnosable w.r.t. (Σ_{oj}: $j \in H$) and T_{fz}";
   • else if $v_z = 1$, assert "The system is modular diagnosable w.r.t. (Σ_{oj}: $j \in H$) and T_{fz}";

Proposition 14 *Assuming there exists an F-confused cycle in $RG_{z||S_c^*} = CoAc(MRG_z)||RG_{S_c^*}$ ($RG_{S_c^*}$ is obtained from RG_{S_c} by using the $\varepsilon-$reduction to reduce the $\varepsilon-$transitions*). The sequence of transitions that leads to the F-confused cycle is $\widetilde{\sigma} = \widetilde{t}_{a,1} \cdots \widetilde{t}_{a,p}(\widetilde{t}_{b,1} \cdots \widetilde{t}_{b,q})^*$ (the sequence*

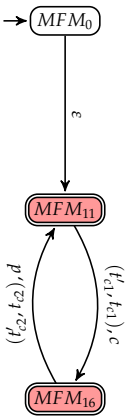
belong to $(\widetilde{T}_z)^*$). The transition in $\widetilde{T}_z$ whose label is the same with that of a transition in T_{S_c} are denoted as the common transition. $\widetilde{T}_{cz} \subset \widetilde{T}_z$ denotes the set of common transitions.

Let us consider the transitions in $CoAc(MRG_z)$ labeled by uncommon transitions in $\widetilde{T}_z$ as $\varepsilon-$transitions and $CoAc(MRG_{z^*})$ is obtained from $CoAc(MRG_z)$ by using the $\varepsilon-$reduction to reduce the $\varepsilon-$transitions w.r.t. the two constraints above. We build $RG_{z^*||S_c^*} = CoAc(MRG_{z^*})||RG_{S_c^*}$. From the initial node of $RG_{z^*||S_c^*}$, the sequence of transitions that leads to the F-confused cycle is $\widetilde{\sigma}' = P_{\{\widetilde{T}_z, \widetilde{T}_{cz} \cup \widetilde{T}^\alpha\}}(\widetilde{\sigma})$ (where if there exists an $F-$confused cycle that does not contain any common transitions $\widetilde{T}^\alpha$ contains the transition that is not considered as $\varepsilon-$transitions* in the cycle as it was presented in constraint 2; otherwise, $\widetilde{T}^\alpha = \emptyset$).*

Proof: The proof is similar to the proof of Proposition 13. By considering all the transitions* in $CoAc(MRG_z)$ labeled by uncommon transitions in $\widetilde{T}_z$ as $\varepsilon-$transitions, each F^{M_z}-confused cycle in $RG_{z^*||S_c^*} = CoAc(MRG_{z^*})||RG_{S_c^*}$ corresponds to an F^{M_z}-confused cycle in $RG_{z||S_c^*} = CoAc(MRG_z)||RG_{S_c^*}$. Since only the shared transitions are preserved in $CoAc(MRG_{z^*})$ and all the $\varepsilon-$transitions are reduced,the sequence that leads to the F^{M_z}-confused cycle in $RG_{z^*||S_c^*}$ is $\widetilde{\sigma}' = P_{\{\widetilde{T}_z, \widetilde{T}_{cz} \cup \widetilde{T}^\alpha\}}(\widetilde{\sigma})$. □



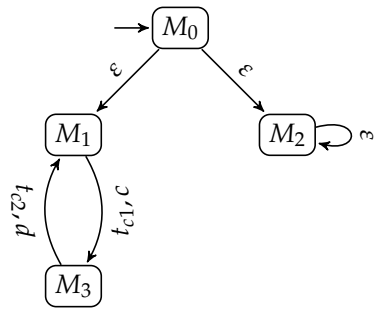
(a) $CoAc(MRG_1)$ with $\varepsilon-$transitions



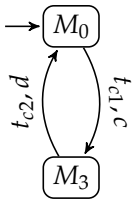
(b) $CoAc(MRG_{1^*})$ by using $\varepsilon-$reduction

Figure 4.19 – $CoAc(MRG_1)$ and $CoAc(MRG_{1^*})$

According to Proposition 13 and Proposition 14, while check modular diagnosability, we can build $RG_{z^*||S_c^*}$ instead of $RG_{z||S_c}$. By using the $\varepsilon-$reduction techniques, all the



(a) RG_2 with $\varepsilon-$transitions



(b) RG_{2^*} by using $\varepsilon-$reduction

Figure 4.20 – RG_2 and RG_{2^*}

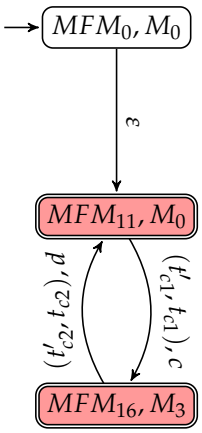


Figure 4.21 – $RG_{1^*||2^*}$

uncommon transitions* are reduced in oder to deal with the combinatorial explosion problem. The new algorithm is presented in Algorithm 15.

Example 53 *Let us consider the $CoAc(MRG_1)$ shown in Figure 4.15 and RG_2 shown in Figure 4.17. The labels of common transitions* in $CoAc(MRG_1)$ are $\{((t'_{c1}, t_{c1}), c), ((t'_{c2}, t_{c2}), d)\}$. The labels of common transitions in RG_2 are $\{(t_{c1}, c), (t_{c2}, d)\}$. The other transitions are considered as $\varepsilon-$transitions shown in Figure 4.19(a) and Figure 4.20(a). By using the $\varepsilon-$reduction technique, the $CoAc(MRG_{1^*})$ and RG_{2^*} are shown in Figure 4.19(b) and Figure 4.20(b). Afterwards, we build $RG_{1^*||2^*} = CoAc(MRG_{1^*})||RG_{2^*}$ that is shown in Figure 4.21. There exists an F^{M_1}-confused cycle shown in the shadow zone. Therefore, the system is not modular diagnosable.*

From Example 53, By using the $\varepsilon-$reduction technique, the combinatorial explosion problem is reduced. $CoAc(MRG_1)$ and RG_2 are simplified before building the parallel composition. $RG_{1^*||2^*} = CoAc(MRG_{1^*})||RG_{2^*}$ is built to verify the existence of F^{M_1}-confused cycle in order to check the modular diagnosability.

4.2.6 Complexity analysis

In this section, we analyze at first the computational complexity of the approaches for modular diagnosability analysis in the literature and then the computational complexity of our approach.

Assuming an automaton system G_H modeled by a collection of automaton modules ($H =: \{1, \cdots, m\}$, $G_H := \|_{j \in H}\, G_j$). For the approach in [Con+06], the local diagnosability of each module is analyzed by using the diagnoser approach. For each module $z \in H$ that is not locally diagnosable w.r.t. its set of fault events Σ_{fz} (i.e., there exists an $F-$indeterminate cycle [Sam+95] in the local diagnoser G_{dz}), the parallel composition of this diagnoser G_{dz} and the local diagnosers of other modules are built in order to verify if the $F-$indeterminate cycle survives. Assuming that $|X_z|$ is the maximal number of states of one module and $|\Pi_{fz}|$ is the maximal number of fault classes in one module. The complexity of constructing the local diagnoser is $\mathcal{O}(2^{|X_z| \times 2^{|\Pi_{fz}|}})$. In the worst case of this approach, the parallel composition of all the local diagnosers are built. Therefore, the complexity of the approach in [Con+06] is $\mathcal{O}((2^{|X_z| \times 2^{|\Pi_{fz}|}})^m)$, i.e., $\mathcal{O}(2^{m|X_z| \times 2^{|\Pi_{fz}|}})$. Thus, the complexity is exponential.

In [MP13], the idea is similar to the approach in [Con+06], but the verifier approach in [YL02] is applied. In the worst case, the verifier of the monolithic system is built. The number of states of the monolithic model is $|X_z|^m$, the number of the events is at most $|\Sigma| = |\cup_{z \in H} \Sigma_z|$, the number of fault classes of the monolithic system is $|\Pi_{fH}|$. The complexity of the approach in [MP13] is $\mathcal{O}(|X_z|^{2m} \times |\Sigma| \times |\Pi_{fH}|)$.

In [Sch13], the module models are simplified by using an abstraction-based technique before analyzing the modular diagnosability. In the worst case, the complexity to analyze an event-based modular diagnosability is linear in the number of states and second order polynomial in the number of transition of monolithic model. Therefore, the complexity of this approach is $\mathcal{O}(|X_z|^m \times (|X_z|^m \times |\Sigma|)^2)$, i.e., $\mathcal{O}(|X_z|^{3m} \times |\Sigma|^2)$.

Afterwards, let us analyze the complexity of our approach for modular diagnosability analysis using LPN model. Given an LPN model. A set of modules $LPN_1, \cdots, LPN_m$ ($H = \{1, \cdots, m\}$) is a *sound* decomposition. To analyze the local diagnosability of an LPN module, by building the VN and MRG, the complexity is equal to that of the VN approach in [Cab+12], which is analyzed in Section 3.2.5. Assuming that the module LPN_z contains the maximum number of states in the RG, which is denoted as $|X_z|$. The complexity to analyze the local diagnosability of module LPN_z is $\mathcal{O}(|P_z| \times |T_z|^2 + |X_z|^2 |T_z|^2)$, where $|P_z|$ is number of the places of LPN_z, $|T_z|$ is the number of the transitions of LPN_z and $|X_z|$ the number of the nodes in the RG of LPN_z. To analyze the modular diagnosability, in the worst case, we need to build the parallel composition of the MRG MRG_z with the RGs of all elementary LPN modules. To build the parallel composition $RG_{z||H'}$ ($H' = H \backslash \{z\}$), the complexity is $\mathcal{O}(|X_z|^{m+1} \times |T|)$, where $|T|$ is the number of transitions of the monolithic

system (the complexity of using $\varepsilon-$reduction technique is linear on the number of the transitions of the automaton and it can be neglected w.r.t. the complexity of building the parallel composition). Overall, the complexity of our approach is $\mathcal{O}(|P_z| \times |T_z|^2 \times |\Pi_{fH}| + |X_z|^{m+1} \times |T| \times |\Pi_{fH}|)$, where $|\Pi_{fH}|$ is the number of fault classes in the monolithic system. The complexity of our approach is polynomial. Regardless of the complexity for the construction of VN, the complexity of our approach is less than that of the other approaches for modular diagnosability analysis using LPN model.

4.3 Synthesis of the contributions (on modular diagnosability analysis)

In Chapter 4, we introduce the architectures of decentralized diagnosis, modular diagnosis and distributed diagnosis. The three architectures aim at dealing with the combinatorial explosion problem, because they all avoid building the diagnoser of monolithic systems. We present the way that we make the classification of the three architectures. The literature review of these architectures is given.

Afterwards, we focus on the modular diagnosis architecture, especially the modular diagnosability analysis. Different from the approaches in literature, we analyze the modular diagnosability of system modeled by LPN. The originalities of our approach are as follows:

1. For the approaches in [Con+06; MP13; Sch13], the system is modeled by a collection of automaton modules. However, the input model of our approach is the monolithic LPN model. We use the definition of LPN module (in Definition 35) to decompose the monolithic LPN into a *sound* decomposition (in Definition 36). We also release the liveness assumption in [Con+06]

2. We apply the reduction rules to simplify the monolithic LPN model before the decomposition. We prove that the modular diagnosability property is preserved after using these reduction rules. Indeed, the memory cost for analyzing the reduced model is lower.

3. While analyzing the local diagnosability of local modules, we propose a new approach based on the VN approach in [Cab+12]. The VN and its MRG are built and a sufficient and necessary condition for local diagnosability is given. The complexity of this approach is the same with that of the VN approach. However, the MRG is specially designed for modular diagnosability analysis.

4. A new approach for incremental modular diagnosability analysis is proposed. We take advantage of the structural property of MRG and we build the parallel composition of the MRG and the RG of the composed LPN module in order to check the modular diagnosability property. A sufficient and necessary condition for modular

diagnosability is proposed. We also use the $\varepsilon-$reduction technique to reduce the combinatorial explosion problem. The complexity of our approach is polynomial.

CHAPTER 5

CASE STUDY

Contents

In order to evaluate the proposed approach of this thesis, we provide some experimental results. The manufacturing benchmark in [Hos+13] and the multi-track level crossing benchmark in [Liu+16] are analyzed.

5.1 Manufacturing benchmark

In this section, the diagnosability property of the benchmark proposed in [Hos+13] is analyzed. In [Giu08], the author describes a manufacturing benchmark for fault diagnosis to test different diagnosis approaches. In [Hos+13], a new version of the manufacturing benchmark in [Giu08] is proposed. This benchmark in [Hos+13] describes a manufacturing system characterized by three parameters: n, m and k, where:

1. n is the number of production lines;
2. m is the number of units of the final product that can be produced and each unit of product is composed of n parts (produced in parallel\ simultaneously);

3. k is the number of operations that each part must undergo in each line.

The manufacturing benchmark provided in [Hos+13] is shown in Figure 5.1. The fault transitions are represented by red boxes. The other transitions are normal transitions.They can be observable or unobservable depending on different experimental conditions and they are represented by blue boxes.

Some experimental simulations using this benchmark, were executed in [Bou16; Hos+13] by using different approaches, such as the diagnoser approach [Sam+95], verifier approach [YL02], MBRG/BRD approach [Cab+14] and variant diagnoser approach [Bou16].

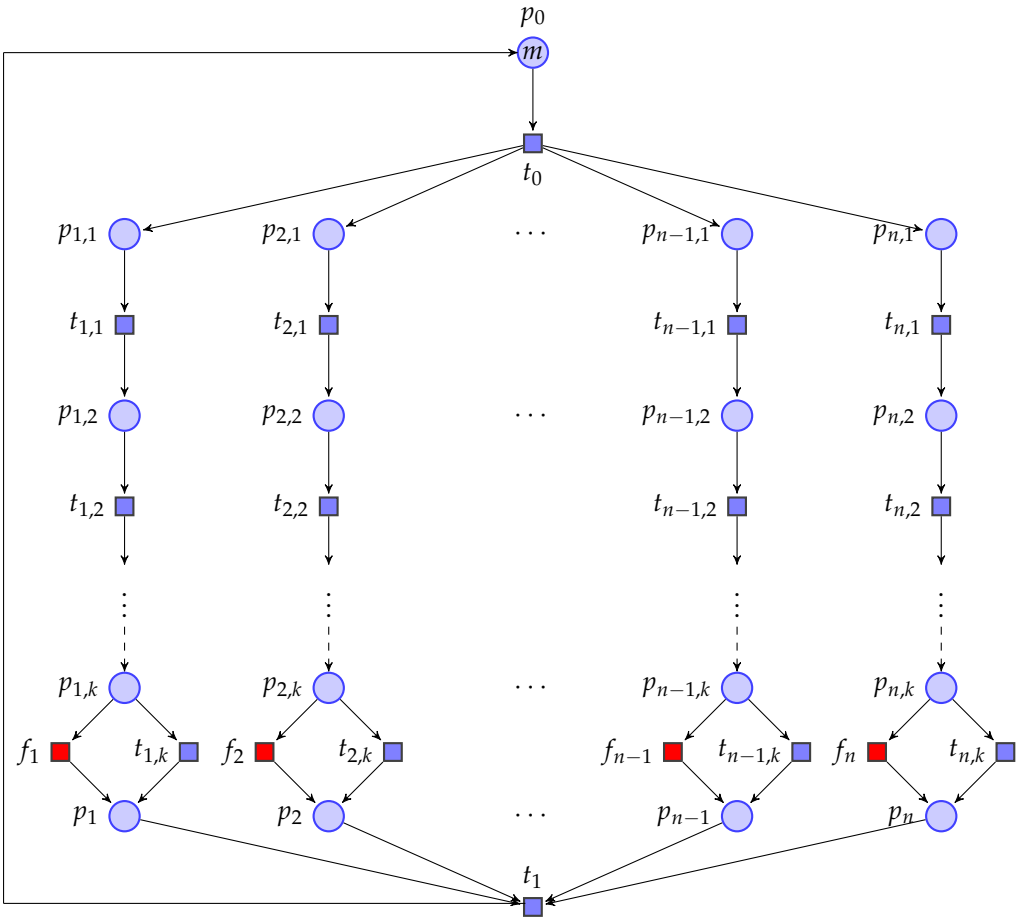


Figure 5.1 – The PN benchmark in [Hos+13]

In order to analyze the diagnosability property of the benchmark in [Hos+13], the following assumptions are proposed:

1. t_0 and t_1 are *exclusively labeled observable transitions* (ELOTs);

2. $\forall i \in \{1, \cdots, n\}$ and $\forall j \in \{1, \cdots, k\}$, the transition $t_{i,j}$ is either a regular unobservable transition or an ELOT. It means that each sensor monitors uniquely the behavior of one production line.

The experimental results in [Bou16; Hos+13] are obtained with these two assumptions. These assumptions could be released, but we use them in order to well explain the application of reduction rules.

5.1.1 Monolithic diagnosability analysis of the manufacturing benchmark

For the given model in Figure 5.1, as it is assumed, there exist some unobservable transitions and ELOTs. In order to simplify the LPN model, the reduction rules (1) and (6), proposed in Section 3.2.1, can be applied to reduce the transitions $t_{i,j}$ ($i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, k-1\}$) which are regular unobservable transitions or ELOTs.

Thus, one can apply the reduction rules proposed in Section 3.2.1 to simplify at first the LPN model. The reduction rules (1) and (6) can be applied to reduce the transition $t_{i,j}$ ($i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, k-1\}$) which is a regular unobservable transition or an ELOT.

By using the reduction rules:

1. For the production line i, if $t_{i,1}, \cdots t_{i,k-1}$ are all regular unobservable transitions, all these transitions and corresponding places can be reduced;
2. For the production line i, if $\exists j \in \{1, \cdots, k-1\}$ s.t. $t_{i,j}$ is an ELOT, all the transitions and corresponding places can be reduced except for $t_{i,h}$ ($h \in \{1, \cdots, k-1\}$), which is the last ELOT in this production line.

In the following experimental test, $\forall i \in \{1, \cdots, n\}$, $t_{i,k-1}$ is considered as an ELOT. The reduced PN model is shown in Figure 5.2.

We implement the reduction rules using MATLAB. We correct the error of the algorithm proposed in [Mm+13]. Instead of working on the incidence matrix of the PN model, we use the pre-incidence matrix and post-incidence matrix (An illustration and a new algorithm is provided in Appendix B). The algorithm is efficient. For example, when $k = 6$ and $n = 8$, it takes less than 0.1 second to get the pre-incidence matrix and post-incidence matrix of the reduced PN model. The test is performed with a 64-bit PC (CPU: Intel Core i7, 2.4 GHz, RAM: 8GB).

Since the reduction rules is a complementary technique for all the diagnosability analysis approaches, the diagnosability property is preserved by using the reduction rules. Hence, the reduced model can be analyzed for diagnosability analysis, instead of the initial model, in order to improve the efficiency.

In order to evaluate our proposed techniques, three cases of the benchmark are discussed:

- **Case 1**: $m = 1$ and $\forall i \in \{1, \cdots, n\}$, all the transitions $t_{i,k}$ are ELOTs;

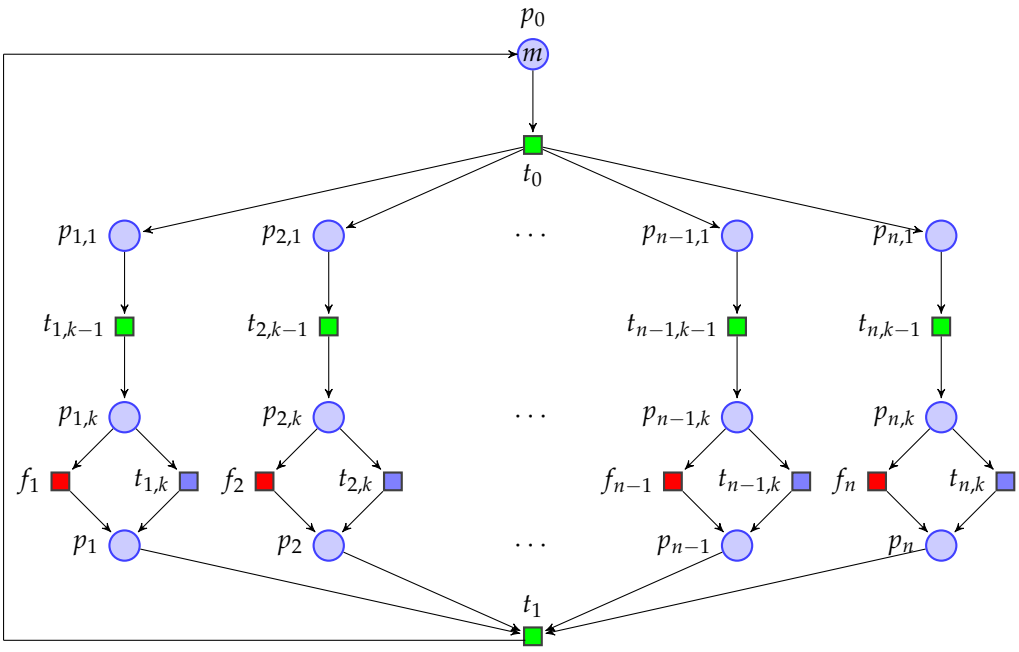


Figure 5.2 – Reduced PN benchmark model

- **Case 2**: $m = 1$ and $\forall i \in \{1, \cdots, n\}$, all the transitions $t_{i,k}$ are regularly unobservable;
- **Case 3**: $m \geq 2$ and $\forall i \in \{1, \cdots, n\}$ all the transitions $t_{i,k}$ are ELOTs or all the transitions $t_{i,k}$ are regularly unobservable.

5.1.1.1 Case 1

In this case, $m = 1$ and $\forall i \in \{1, \cdots, n\}$, all the transitions $t_{i,k}$ are ELOTs. The reduced PN model is shown in Figure 5.3.

It is worth noticing that if $m = 1$, the PN model is *safe* and *live*. The sufficient condition proposed in Section 3.2.2 is used to analyze the diagnosability.

In this case, the Constraint 3.3 (Section 3.2.2 page 72) has no solution. For facility of readers, let us recall the Constraint 3.3 as follows:

$$\begin{cases} \sum_i (d_i \cdot \overrightarrow{V}(\overrightarrow{\Omega}^N_{min,i})) = \sum_r (b_r \cdot \overrightarrow{V}(\overrightarrow{\Omega}^F_{min,r})) \\ \sum_r b_r \geq 1 \\ if\ d_i < 0 \ \Rightarrow \ a_i = 0; a'_i = d_i \\ if\ d_i > 0 \ \Rightarrow \ a_i = d_i; a'_i = 0 \\ b_r \in \mathbb{N},\ d_i \in \mathbb{Z} \end{cases}$$

First, each minimal T-invariant contains the transitions t_0 and t_1 and they are ELOTs.

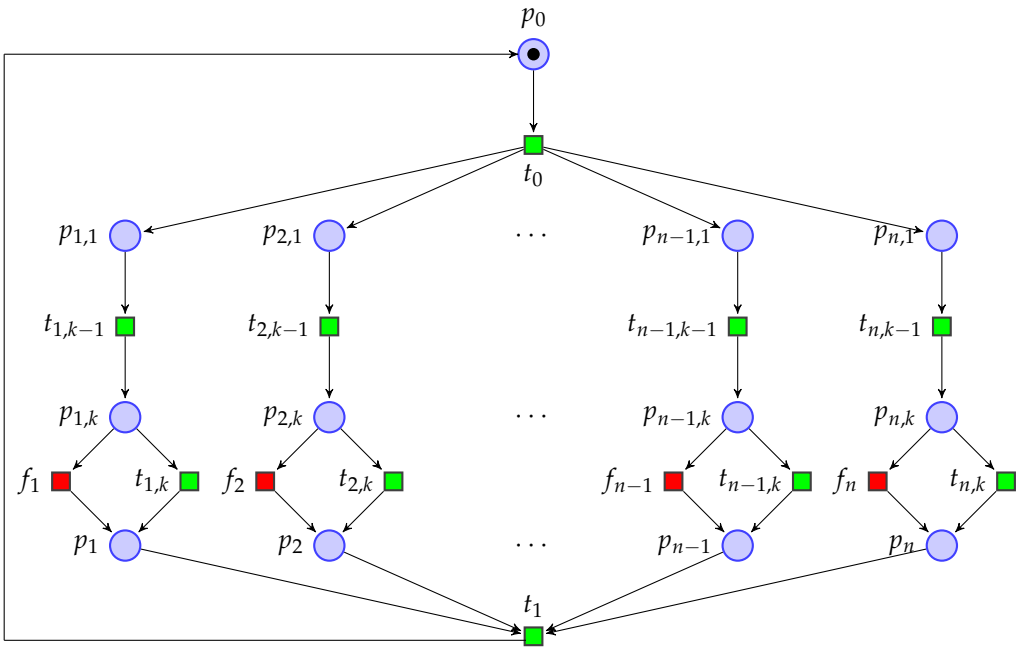


Figure 5.3 – Reduced PN benchmark model in **Case 1**

Therefore, $\sum_i d_i = \sum_r b_r$. The set of normal minimal T-invariant $\mathcal{I}_N$ contains only one minimal T-invariant $\overrightarrow{\Omega}^N_{min,1}$. $\overrightarrow{\Omega}^N_{min,1}$ contains the transitions $\{t_0,\ t_{1,k-1},\ t_{1,k},\ \cdots,\ t_{n,k-1},\ t_{n,k},\ t_1\}$. For a faulty minimal T-invariant, if it contains just one fault transition f_h, then this minimal T-invariant contains $\{t_0,\ t_{1,k-1},\ t_{1,k},\ \cdots,\ t_{h,k-1},\ f_h,\ \cdots,\ t_{n,k-1},\ t_{n,k},\ t_1\}$. If a minimal T-invariant contains two fault transitions f_p and f_q, it contains $\{t_0,\ t_{1,k-1},\ t_{1,k},\ \cdots,\ t_{p,k-1},\ f_p,\ \cdots,\ t_{q,k-1},\ f_q,\ \cdots,\ t_{n,k-1},\ t_{n,k},\ t_1\}$. By analogy, if a minimal T-invariant contains several fault transitions, one can also obtain its set of transitions.

The constraint 3.3 is rewritten as:

$$\begin{cases} d_1 \cdot \overrightarrow{V}(\overrightarrow{\Omega}^N_{min,1}) = \sum_r (b_r \cdot \overrightarrow{V}(\overrightarrow{\Omega}^F_{min,r})) \\ d_1 = \sum_r b_r \\ \sum_r b_r \geq 1 \\ b_r \in \mathbb{N},\ d_1 \in \mathbb{N} \end{cases}$$

$t_{1,k}, \cdots, t_{n,k}$ are ELOTs. For the faulty minimal T-invariant that contains a fault transition f_h, it does not contain the transition $t_{h,k}$ which is an ELOT. Hence, the constraint has no solution. Therefore, the PN model is diagnosable.

5.1.1.2 Case 2

In this case, $m = 1$ and $\forall i \in \{1, \cdots, n\}$, all the transitions $t_{i,k}$ are regularly unobservable. The reduced model is shown in Figure 5.4. The PN model is still *safe* and *live*.

It is worth noticing that $\forall h \in \{1, \cdots, n\}$, the pre-place and post-place of $t_{h,k}$ and f_h are the same. Assuming that $t_{h,k}$ is regularly unobservable. In this case, the constrain has a solution. Note that $\overrightarrow{\Omega}^N_{min,1}$ is the normal minimal T-invariant and $\overrightarrow{\Omega}^F_{min,h}$ is the fault minimal T-invariant that contains $\{t_0,\ t_{1,k-1},\ t_{1,k},\ \cdots,\ t_{h,k-1},\ f_h,\ \cdots,\ t_{n,k-1},\ t_{n,k},\ t_1\}$. We have

$$\overrightarrow{V}(\overrightarrow{\Omega}^N_{min,1}) = \overrightarrow{V}(\overrightarrow{\Omega}^F_{min,h})$$

Hence, we need to use other approaches such as the diagnoser approach to check the diagnosability.

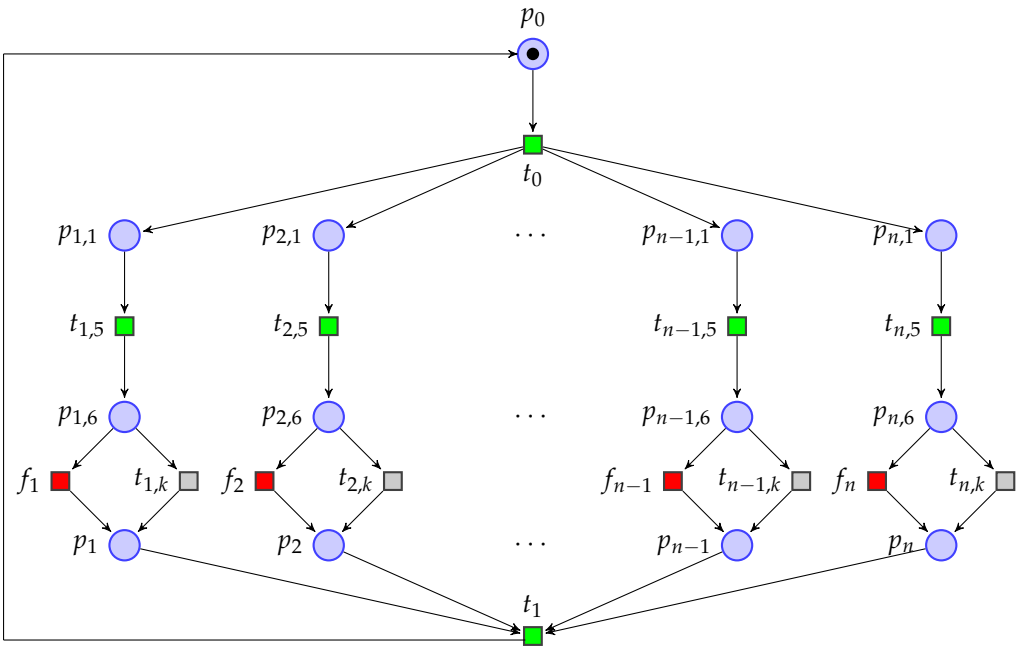


Figure 5.4 – Reduced PN benchmark model in **Case 2** for $k = 6$

In order to evaluate the difference between analyzing the initial model and the reduced model, the following configuration of the benchmark is considered:

- The parameters are defined as $m = 1, k = 6$ and $n = 3, 4, \cdots, 7$. Moreover, transitions t_0, t_1 and $t_{h,1}$, $t_{h,3}$, $t_{h,5}$ for $h \in \{1, \cdots, n\}$ are ELOTs. The transitions f_h for $h \in \{1, \cdots, n\}$ are fault transitions and the other transitions are regularly unobservable.

We use the diagnoser approach [Sam+95] and verifier approach [YL02] to analyze respectively the initial models and reduced models.

The reachability graph of the considered PN model is generated with the help of the PN analysis tool TINA [Ber+04]. The file format is ".aut" and it is transformed into a ".fsm" file.

Afterwards, we use the `dcycle.exe, diag_UR.exe, verifier_dia.exe` functions of UMDES Library [Laf00] to get the experimental results of diagnosability analysis. The experiments are executed with a 64-bit PC (CPU: Intel Core i7, 2.4 GHz, RAM: 8GB). We set 5 hours as the time limit for the analysis.

The experiment result for analyzing the initial model is shown in Table 5.1 and that for analyzing the reduced model is shown in Table 5.2, where:

- $|P|$ and $|T|$ are respectively the number of places and transitions of the PN model;
- $|RG_S|$ and $|RG_T|$ are respectively the number of states and transitions of the reachability graph;
- $|X_D|$ and $|\mathcal{T}_D|$ are obtained by using the diagnoser approach in [Sam+95]. $|X_D|$ is the number of states of the diagnoser and $|\mathcal{T}_D|$ is the time for generating the diagnoser and analyzing the diagnosability;
- $|X_V|$ and $|\mathcal{T}_V|$ are obtained by using the verifier approach in [YL02]. $|X_V|$ is the number of states of the verifier and $|\mathcal{T}_V|$ is the time for generating the verifier and analyzing the diagnosability;
- "Diag" is the diagnosability verdict.

Table 5.1 – The experimental result for analyzing initial models

	LPN		RG		Diagnoser approach		Verifier approach		Diag
n	$\|P\|$	$\|T\|$	$\|RG_S\|$	$\|RG_T\|$	$\|X_D\|$	$\|\mathcal{T}_D\|$	$\|X_V\|$	$\|\mathcal{T}_V\|$	
3	22	23	344	1031	130	0.2s	3147	0.4s	Non-diagnosable
4	29	30	2402	9606	514	107s	32337	45s	
5	36	37	16808	84037	2050	3650s	*	o.t.	
6	43	44	117650	705896	*	o.t.	*	o.t.	
7	50	51	823544	5764803	*	o.t.	*	o.t.	

*: No result obtained in 5 hours. o.t.: out of time (simulation time $\geq$ 5 hours).

Table 5.2 – The experimental result for analyzing reduced models

	LPN		RG		Diagnoser approach		Verifier approach		Diag
n	$\|P\|$	$\|T\|$	$\|RG_S\|$	$\|RG_T\|$	$\|X_D\|$	$\|\mathcal{T}_D\|$	$\|X_V\|$	$\|\mathcal{T}_V\|$	
3	10	11	28	83	18	$\approx$0s	213	0.1s	Non-diagnosable
4	13	14	82	326	34	0.1s	881	0.2s	
5	16	17	244	1217	66	0.6s	3645	0.8s	
6	19	20	730	4376	130	69s	14993	12s	
7	22	23	2188	15311	258	4h44m15s	*	o.t.	

*: No result obtained in 5 hours. o.t.: out of time (simulation time $\geq$ 5 hours).

By using the reduction rules, some transitions and places are suppressed. For the same n, the reduced PN model contains fewer places and transitions. It is reasonable that for the same number of m ($m = 1$), the reachability graph of the reduced PN model contains fewer states and transitions. Analogously, the diagnoser and the verifier of the reduced PN model contain also fewer states. The memory cost and time cost for diagnosability analysis are lower by using the reduced PN model.

The technique of reduction rules is a strong complement for the diagnosability analysis approaches. By simplifying a priori the given PN model, it becomes feasible for the approaches to analyze a large-scale system. For example, for $n \geq 6$ ($n \geq 5$ for the verifier approach), the diagnoser approaches cannot get any simulation result in 5 hours by using the initial model. However, by using the reduced model, the analysis using diagnoser approach can be terminated in 5 hours for $n = 6, 7$ ($n = 5, 6$ for the verifier approach). Moreover, the result shows that the diagnosability property of the reduced PN model keeps consistent with that of the reduced model.

5.1.1.3 Case 3

When $m \geq 2$, the states space of PN model explodes very quickly w.r.t. the increase of k and n. Even for the reduced PN model, when $m = 2$ and $n = 5$, no result is obtained in 5 hours by using diagnoser approach and verifier approach. It is not feasible to check the diagnosability property by using most of the approaches mentioned before. However, we can use the on-the-fly diagnosability analysis using T-invariants to analyze the diagnosability.

In this case, one assumes that $\forall i \in \{1, \cdots, n\}$, all the transitions $t_{i,k}$ are ELOTs. (Since in **Case 2**, when $m = 1$ and $\forall i \in \{1, \cdots, n\}$, all the transitions $t_{i,k}$ are regularly unobservable, the PN model is not diagnosable. If we increase the number of tokens, the system is still not diagnosable. However, the on-the-fly diagnosability analysis using T-invariants is also valid when all the transitions $t_{i,k}$ are regularly unobservable.) The reduced model is shown in Figure 5.5.

As it was analyzed in **Case 1**, there is only one minimal T-invariant $\overrightarrow{\Omega}^{N}_{min,1}$ which contains the transitions $\{t_0,\ t_{1,k-1},\ t_{1,k},\ \cdots,\ t_{n,k-1},\ t_{n,k},\ t_1\}$. Let us consider the fault transition f_1. The processes (Section 3.2.4) of this approach are as follows:

- Step 1: From the initial marking, the Algorithm 5 (page 86) is applied to generate a minimal path to enable the fault transition f_1. By using the Algorithm 8 ($MODE = F$) (page 88), it is found out that the firing sequence of transitions $t_0 t_{1,k-1}$ needs to be fired in order to enable f_1. After that it is possible to obtain an F-uncertain state. The current distribution of the tokens is that there are $m - 1$ tokens in p_0 and there is one token in each place of $p_{1,k}$, $p_{2,1}$, $p_{3,1}$, $\cdots$, and $p_{n,1}$.

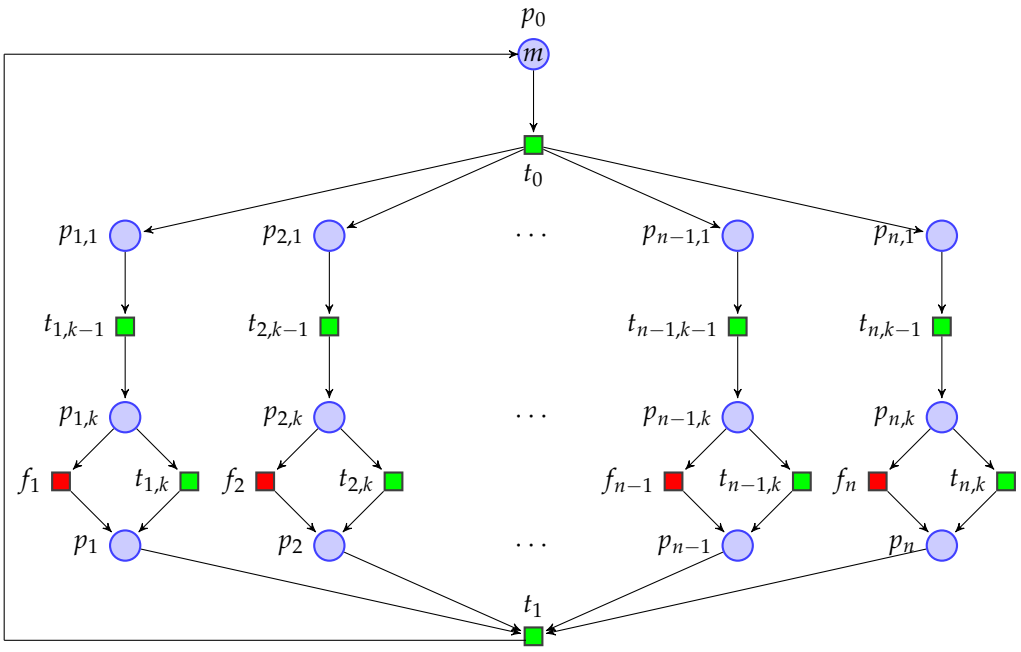


Figure 5.5 – Reduced PN benchmark model in **Case 3**

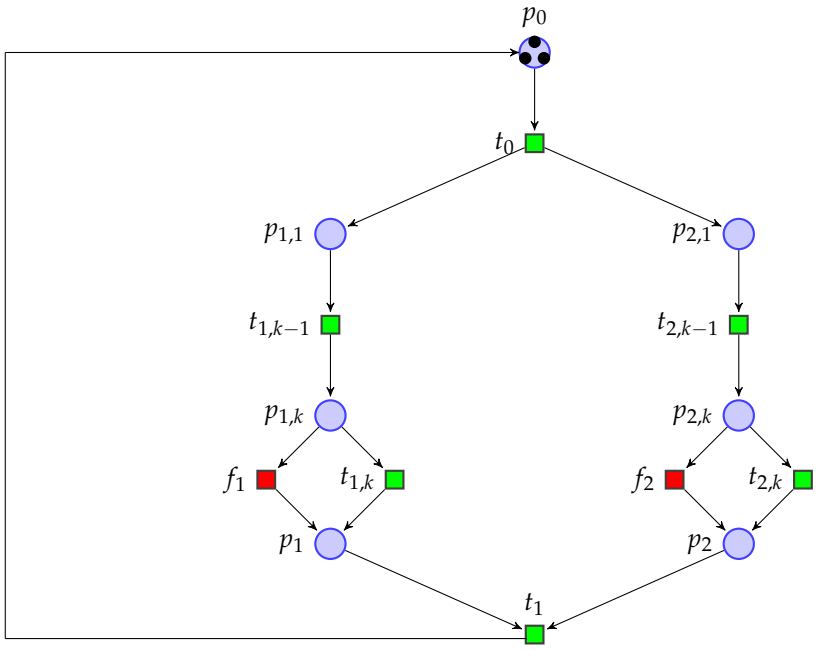


Figure 5.6 – Reduced PN benchmark model with $m = 3$ and $n = 2$

- Step 2: Afterwards, the Algorithm 6 (page 86) is applied to generate a minimal path to enable an observable transition in the normal minimal T-invariant $\overrightarrow{\Omega}^{N}_{min,1}$. No matter which observable transition is chosen, the result does not change. For example, we chose the transition $t_{1,k-1}$. By using the Algorithm 8 ($MODE = T$), it is found out that the firing sequence of transitions t_0 needs to be fired in order to enable $t_{1,k-1}$. The current distribution of the tokens is that there are $m - 2$ tokens in

p_0; there is one token in each place of $p_{1,1}$ and $p_{1,k}$; there are two tokens in each place of $p_{2,1}$, $p_{3,1}$, $\cdots$, and $p_{n,1}$.

- Step 3: Then, the Algorithm 7 (page 87) is applied to find a firing sequence of $\overrightarrow{\Omega}^{N}_{min,1}$. At the current state, one of the firing sequence is $t_{1,k-1}t_{1,k}\cdots t_{n,k-1}t_{n,k}t_1t_0$. We will get an indeterminate cycle corresponding to f_1.

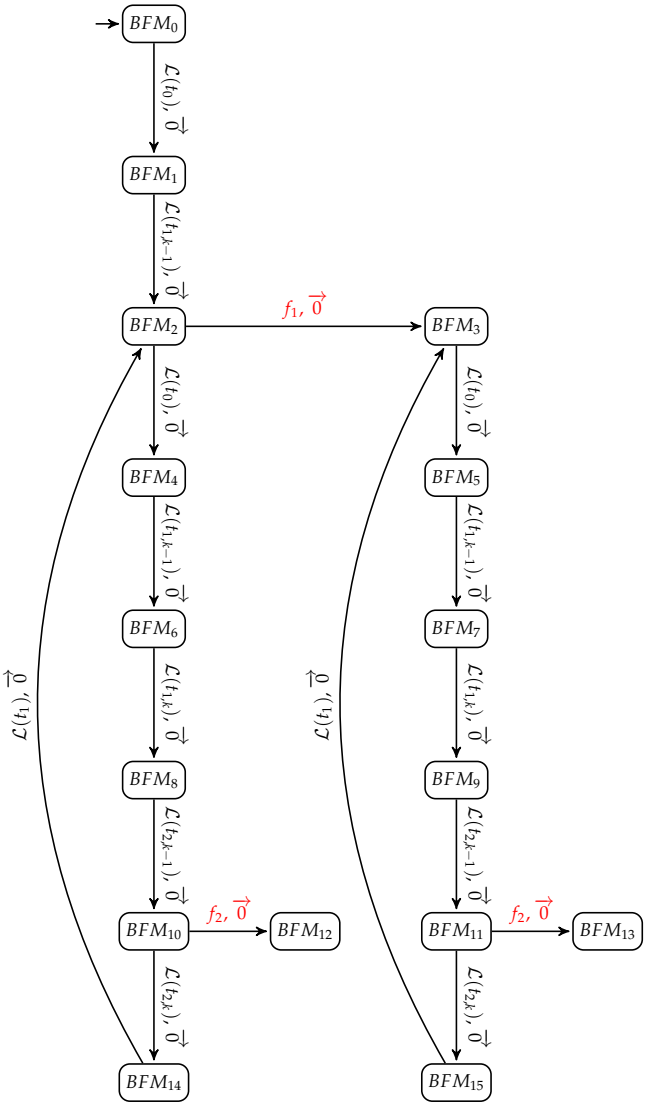


Figure 5.7 – BFG of the PN model in Figure 5.6

Example 54 *In this example, we assume that* $m = 3$ *and* $n = 2$*. The PN model is shown in Figure 5.6. Let us consider* f_1 *and* f_2 *belong to the same fault class. The initial marking is*

$$M_0 = \begin{matrix} p_0 & p_{1,1} & p_{1,k} & p_1 & p_{2,1} & p_{2,k} & p_2 \end{matrix}$$
$$\begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\tau$$

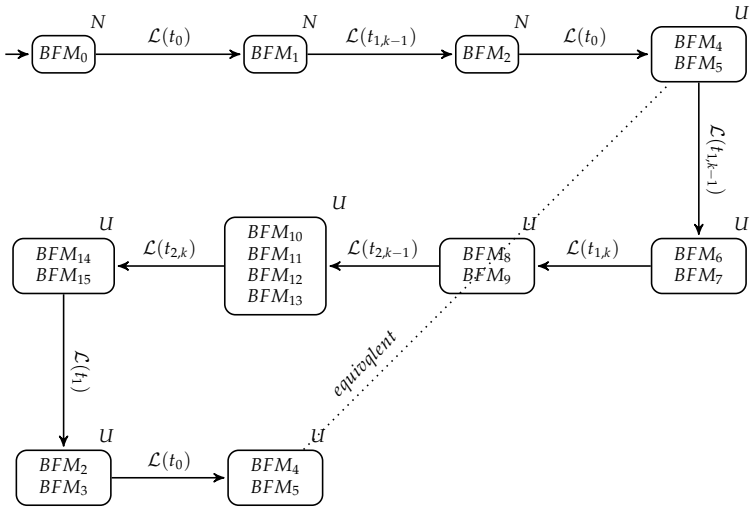


Figure 5.8 – BFST of the PN model in Figure 5.6

Table 5.3 – BFMs and e-vectors of the BFG and BFST (Figure 5.7 and Figure 5.8)

j	BFM_j	j	BFM_j
0	$[3\,0\,0\,0\,0\,0\,0 \mid 0]^\tau$	9	$[1\,0\,0\,2\,2\,0\,0 \mid 1]^\tau$
1	$[2\,1\,0\,0\,1\,0\,0 \mid 0]^\tau$	10	$[1\,0\,1\,1\,1\,1\,0 \mid 0]^\tau$
2	$[2\,0\,1\,0\,1\,0\,0 \mid 0]^\tau$	11	$[1\,0\,0\,2\,1\,1\,0 \mid 1]^\tau$
3	$[2\,0\,0\,1\,1\,0\,0 \mid 1]^\tau$	12	$[1\,0\,1\,1\,1\,0\,1 \mid 1]^\tau$
4	$[1\,1\,1\,0\,2\,0\,0 \mid 0]^\tau$	13	$[1\,0\,0\,2\,1\,0\,1 \mid 1]^\tau$
5	$[1\,1\,0\,1\,2\,0\,0 \mid 1]^\tau$	14	$[1\,0\,1\,1\,1\,0\,1 \mid 0]^\tau$
6	$[1\,0\,2\,0\,2\,0\,0 \mid 0]^\tau$	15	$[1\,0\,0\,2\,1\,0\,1 \mid 1]^\tau$
7	$[1\,0\,1\,1\,2\,0\,0 \mid 1]^\tau$	16	$[2\,0\,1\,0\,1\,0\,0 \mid 0]^\tau$
8	$[1\,0\,1\,1\,2\,0\,0 \mid 0]^\tau$	17	$[2\,0\,0\,1\,1\,0\,0 \mid 1]^\tau$

The only normal minimal T-invariant is

$$\overrightarrow{\Omega}^N_{min,1} = \begin{matrix} t_0 & t_{1,k-1} & t_{1,k} & f_1 & t_{2,k-1} & t_{2,k} & f_2 & t_1 \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}\tau$$

By following the above processes, we can build the BFG and BFST of the PN model, which are shown in Figure 5.7 and Figure 5.8.

Step 1: Considering the fault transition f_1. From the initial BFM_0, the sequence $t_0 t_{1,k-1}$ needs to be fired to enable f_1. The sequence of events is $\mathcal{L}(t_0 t_{1,k-1})$. Then, after the BFS that contains BFM_2, it is possible that the next BFS is F-uncertain.

Step 2: We choose transition $t_{1,k-1}$ which belongs to the normal T-invariant. By using Algorithm 8 ($MODE = T$), t_0 needs to be fired in order to enable $t_{1,k-1}$. Therefore, the following BFS is built by firing the event $\mathcal{L}(t_0)$ and we get the BFS that contains BFM_4 and BFM_5.

Step 3: From the current BFS, there exists a firing sequence which is $t_{1,k-1} t_{1,k} t_{2,k-1} t_{2,k} t_1 t_0$.

The other nodes of BFST are built by following the sequence of event $\mathcal{L}(t_{1,k-1}t_{1,k}t_{2,k-1}t_{2,k}t_1t_0)$. *Afterwards, an F-indeterminate cycle is found, so the PN model is not diagnosable.*

By using this approach, the efficiency of the depth-first search algorithm is improved, because the priority of the investigation is defined. We orient the path to find quickly the indeterminate cycle. For this manufacturing PN model, even when we increase m and n, we can still get the result that the PN model is not diagnosable by building only the relative part of the indeterminate cycle.

5.1.2 Modular diagnosability analysis of the manufacturing benchmark

In this section, the manufacturing benchmark is analyzed by using the modular diagnosis architecture.

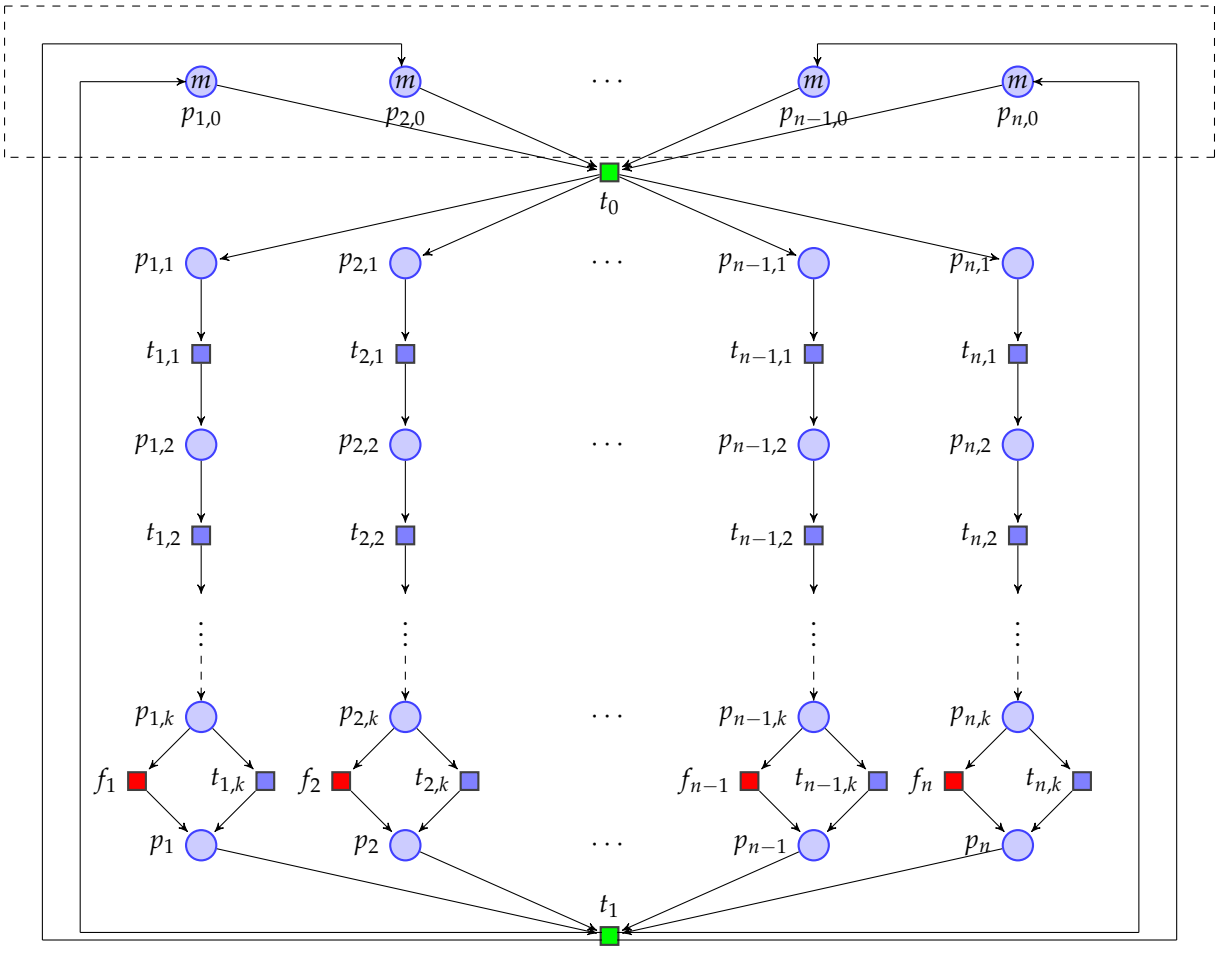


Figure 5.9 – The modified model of the PN model in Figure 5.1

Let us consider the manufacturing benchmark in Figure 5.1. The most intuitive idea is to decompose the model as n PN modules. Each module represents a production line and the shared transitions of the n modules are t_0 and t_1. In this section, we assume that t_0 and t_1 are ELOTs.

Afterwards, we need to verify if the PN model can be decomposed as a *sound* decomposition. According to Definition 36 (page 123) of *sound* decomposition, the transition t_1 can be decomposed, because there is a pre-place of t_1 in each production line module. However, t_0 cannot be decomposed, because it has only one pre-place (p_0). Therefore, we need to modify the PN model in order to apply the modular diagnosability analysis. The modification should not change the behavior of the system, i.e., the language of the PN model should not be changed.

The idea of modifying the initial model is to add places such that each production line module can have one pre-place of t_0. The modified model is shown in Figure 5.9. This PN model is equivalent to the initial model by considering the places $p_{1,0}, \cdots, p_{n,0}$ as the place p_0 in Figure 5.1.

For the PN model Figure 5.1, the firing of t_0 consumes one token from the place p_0 and the firing of t_1 gives one token to the place p_0. It is equivalent that for the PN model Figure 5.9, the firing of t_0 consumes one token of each place of $p_{1,0}, \cdots, p_{n,0}$ and the firing of t_1 gives one token to each place of $p_{1,0}, \cdots, p_{n,0}$.

From the practical point of view, the modified model in Figure 5.9 considers the product as a collection of n components (one token from each place of $p_{1,0}, \cdots, p_{n,0}$), instead of one entire product (one token in p_0) for the initial model in Figure 5.1.

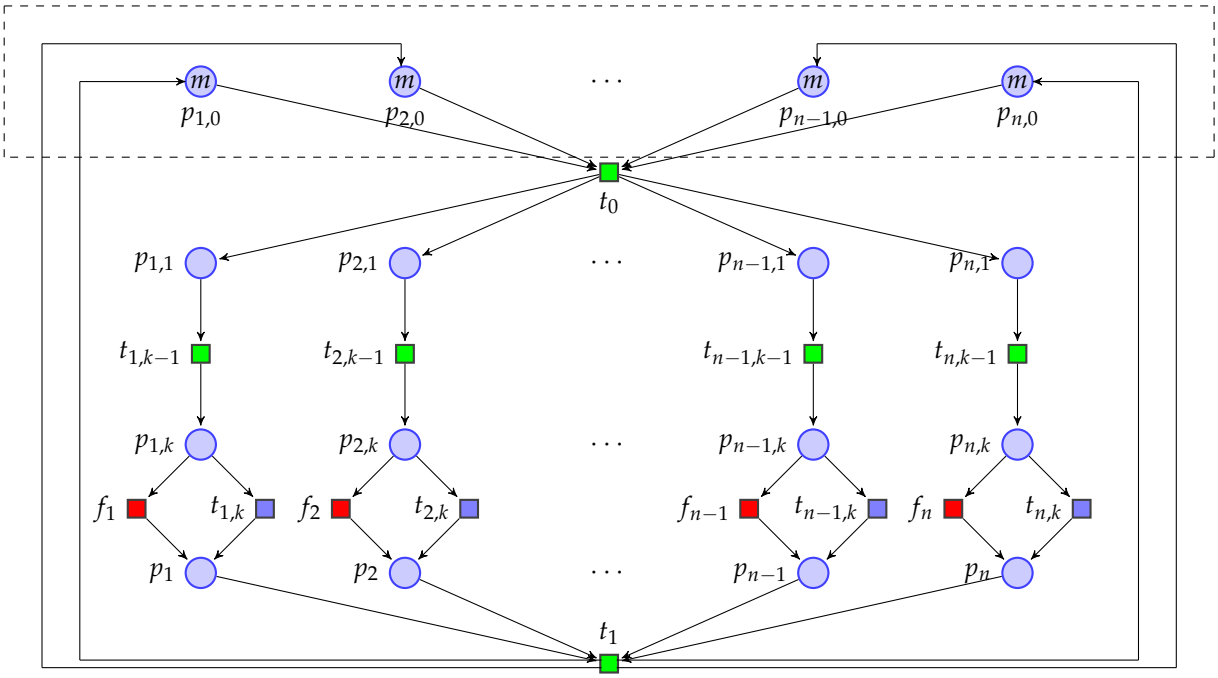


Figure 5.10 – The reduced model of the modified model in Figure 5.9

By modifying the model, we can decompose the PN model as a *sound* decomposition. Assuming that $\forall i, j \in \{1, \cdots, n\}$, f_i and f_j belong to different fault classes. The monolithic system is decomposed into n PN modules. Each module represents one production line

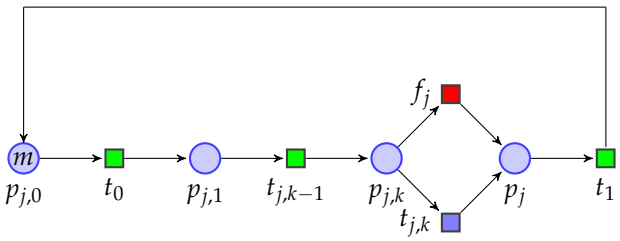


Figure 5.11 – Module j of reduced model

that products one component and each module has similar structure shown in Figure 5.11. (If $\exists i, j \in \{1, \cdots, n\}$, f_i and f_j belong to the same fault classes, we can consider the production line i and j as one module.)

Assuming that $\forall i \in \{1, \cdots, n\}$, $t_{i,k-1}$ is an ELOT. The reduced model is shown in Figure 5.10.

By the *sound* decomposition, the model of the module j is shown in Figure 5.11.

Let us consider two cases as follows:

- **Case 1**: $m = 1$ and $\forall i \in \{1, \cdots, n\}$, all the transitions $t_{i,k}$ are ELOTs;
- **Case 2**: $m = 1$, $h \in \{1, \cdots, n\}$, the transition $t_{h,k}$ is regularly unobservable and $\forall i \in \{1, \cdots, n\} \backslash \{h\}$, all the transitions $t_{i,k}$ are ELOTs.

5.1.2.1 Case 1

Above all, the local diagnosability of each module needs to be analyzed. According to Proposition 10, if each module is locally diagnosable, the result can be given that the system is modularly diagnosable. Since each module has the similar structure, it is sufficient to analyze the local diagnosability of one module.

If $m = 1$ and $\forall i \in \{1, \cdots, n\}$, all the transitions $t_{i,k}$ are ELOTs, the model of module j is shown in Figure 5.12. According to the analysis in Section 5.1.1.1 ($m = 1$ and $n = 1$), for $\forall j \in \{1, \cdots, m\}$, the module j is locally diagnosable. Therefore, the whole system is modularly diagnosable.

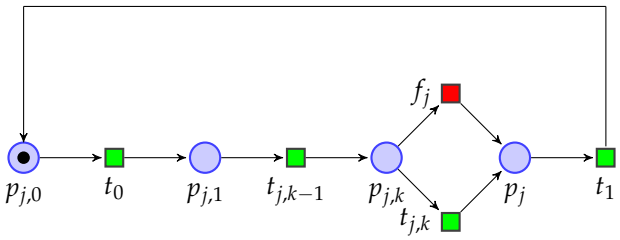


Figure 5.12 – Module j in **Case 1**

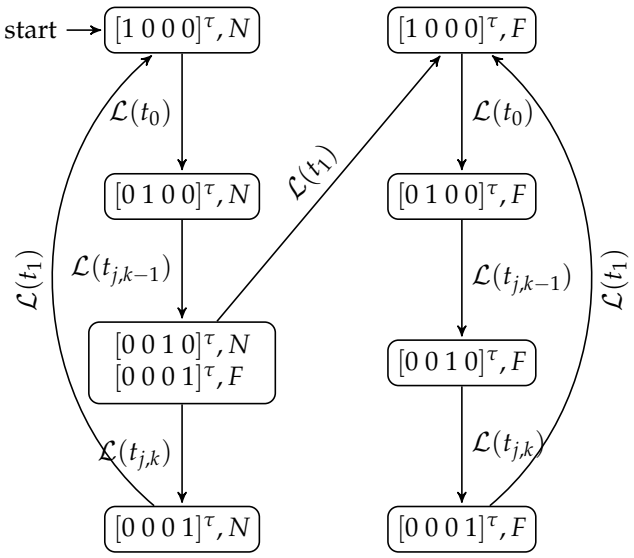


Figure 5.13 – Local diagnoser of module j in **Case 1**

In this case, for on-line diagnosis of a modularly diagnosable system, it is sufficient to build the local diagnoser to diagnose the fault in each module, instead of building the monolithic diagnoser. The local diagnoser of module j is shown in Figure 5.13. The on-line diagnosis of the fault transition f_j can be executed by using the local diagnoser. The fault can be diagnosed, because if the label $\mathcal{L}(t_1)$ is observed just after the occurrence of $\mathcal{L}(t_{j,k-1})$, then it is deduced that the fault has occurred.

Table 5.4 – Comparison of monolithic diagnosis and modular diagnosis

	Monolithic Diagnosis		Modular diagnosis	
n	$\|RG\|$	$\|X_D\|$	$\|RG\|$	$\|X_D\|$
3	28	56	4×3	8×3
4	82	164	4×4	8×4
5	244	488	4×5	8×5
6	730	1460	4×6	8×6
7	2188	*	4×7	8×7

*: No result obtained in 5 hours.

The memory cost for modular diagnosis is lower than that of monolithic diagnosis for this PN model. In stead of building the monolithic diagnoser, n local diagnoser is built. A comparison of the memory cost of monolithic diagnosis and modular diagnosis is provided in Table 5.4 ($|RG|$ is the number of the states of the state space; $|X_D|$ is the number of the states of the diagnoser). By using a modular diagnosis, the combinatorial explosion problem is reduced.

5.1.2.2 Case 2

If $m = 1, h \in \{1, \cdots, n\}$, the transition $t_{h,k}$ is regularly unobservable and $\forall i \in \{1, \cdots, n\} \backslash \{h\}$, all the transitions $t_{i,k}$ are ELOTs, all the modules except for the module h are all locally diagnosable. For module h in Figure 5.14, we use the approach of VN and MRG in Section 4.2.3 to analyze its local diagnosability.

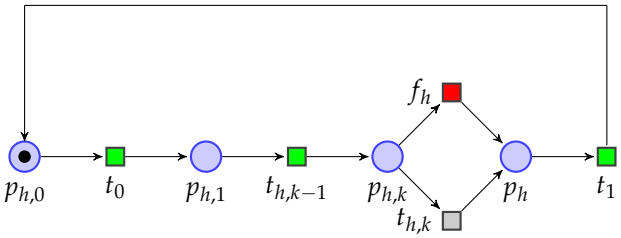


Figure 5.14 – Module h in **Case 2**

The VN and its MRG (denoted as MRG_h) are built in Figure 5.15 and Figure 5.16.

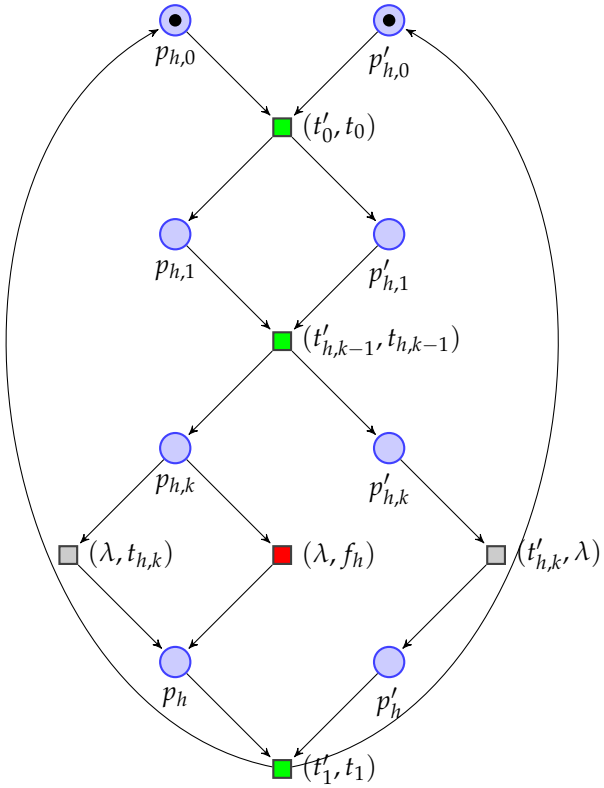


Figure 5.15 – VN of the module h

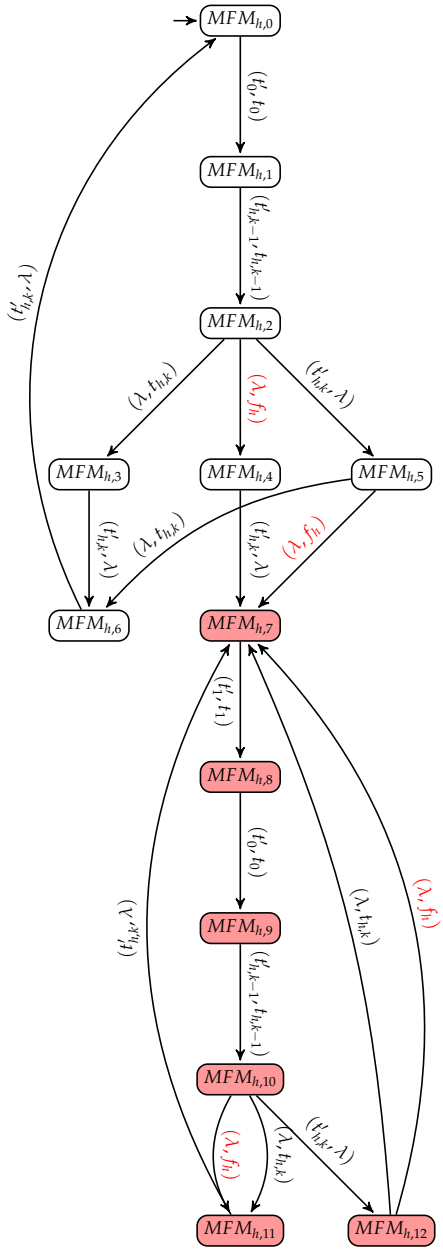


Figure 5.16 – MRG of the VN in Figure 5.15

Table 5.5 – The MFMs in Figure 5.16

i	$MFM_{h,i}$	i	$MFM_{h,i}$
0	$[1\,0\,0\,0\,\vert 1\,0\,0\,0,\ N]^\tau$	7	$[0\,0\,0\,1\,\vert 0\,0\,0\,1,\ F]^\tau$
1	$[0\,1\,0\,0\,\vert 0\,1\,0\,0,\ N]^\tau$	8	$[1\,0\,0\,0\,\vert 1\,0\,0\,0,\ F]^\tau$
2	$[0\,0\,1\,0\,\vert 0\,0\,1\,0,\ N]^\tau$	9	$[0\,1\,0\,0\,\vert 0\,1\,0\,0,\ F]^\tau$
3	$[0\,0\,1\,0\,\vert 0\,0\,0\,1,\ N]^\tau$	10	$[0\,0\,1\,0\,\vert 0\,0\,1\,0,\ F]^\tau$
4	$[0\,0\,1\,0\,\vert 0\,0\,0\,1,\ F]^\tau$	11	$[0\,0\,1\,0\,\vert 0\,0\,0\,1,\ F]^\tau$
5	$[0\,0\,0\,1\,\vert 0\,0\,1\,0,\ N]^\tau$	12	$[0\,0\,0\,1\,\vert 0\,0\,1\,0,\ F]^\tau$
6	$[0\,0\,0\,1\,\vert 0\,0\,0\,1,\ N]^\tau$		

Since there exist $F-$confused cycles shown with the shadow zone, the module h is not locally diagnosable. We mark all the states of $F-$confused cycles. Afterwards, we build the parallel composition of $CoAc(MRG_h)$ and RG of another module. Since the other modules have the similar structure. We build the RG (denoted as RG_j) of module j which is shown in Figure 5.17.

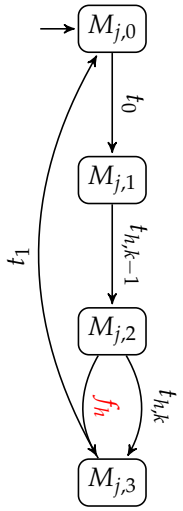


Figure 5.17 – RG of module j

Table 5.6 – The markings in Figure 5.17

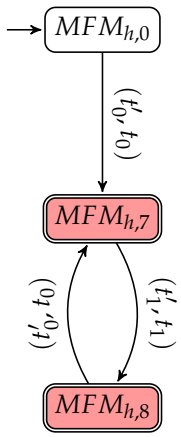
i	$M_{j,i}$
0	$[1\,0\,0\,0\,]^\tau$
1	$[0\,1\,0\,0\,]^\tau$
2	$[0\,0\,1\,0\,]^\tau$
3	$[0\,0\,0\,1\,]^\tau$

All the transitions except for the shared transitions (t'_0, t_0) (t_0) and (t'_1, t_1) (t_1) are considered as ε transitions. We can use the ε-reduction technique to reduce MRG_h and RG_j before building the parallel composition.

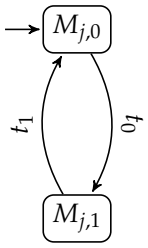
MRG_h and RG_j are built by using $\varepsilon-$reduction, which are shown in Figure 6.1

Afterwards, we build $RG_{h^*||j^*} = CoAc(MRG_{h^*})||RG_{j^*}$ that is shown in Figure 5.19. There exists an F^{M_h}-confused cycle shown in the shadow zone. Therefore, the system composed by the module h and j is not modularly diagnosable.

Since there is an F^{M_h}-confused cycle, we need to verify if the confused cycle still survives while building the parallel composition of the MRG_h with the RG of other composed modules. However, since each module has the similar structure, the RG of the composed module has the same structure after using $\varepsilon-$reduction. After building the



(b) $CoAc(MRG_{h^*})$



(b) RG_{j^*}

Figure 5.18 – $CoAc(MRG_{h^*})$ and RG_{j^*} by using $\varepsilon-$reduction

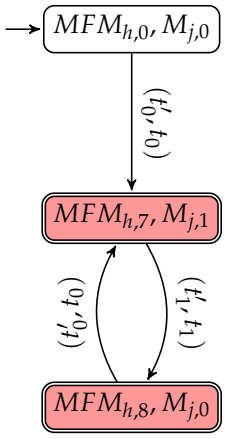


Figure 5.19 – $RG_{h^*||j^*} = CoAc(MRG_{h^*})||RG_{j^*}$

parallel composition of the MRG_h with the RG of any composed modules, there exists still an F^{M_h}-confused cycle. Therefore, the manufacturing PN model in this case is not modularly diagnosable. Hence, we cannot use the local diagnoser of each module for modular diagnosis of the fault.

In this case, the initial PN model needs to be modified in order to obtain a modularly diagnosable manufacturing system which can be used in practice. One solution is to add a sensor to observe the occurrence of $t_{h,k}$ ($t_{h,k}$ becomes an ELOT). Hence, the system of **Case 2** becomes a modularly diagnosable manufacturing system, as the one analyzed in **Case 1** in Section 5.1.2.1.

5.2 Multi-track level crossing benchmark

This section deals with the diagnosability analysis of a multi-track level crossing system which is modeled by LPN.

A level crossing (LC) is an intersection where a railway line crosses a road or path at

the same level. LC safety is a critical issue, because level crossing accidents often generate serious problems, such as devices destruction, traffic disturbances and human damages.

The monolithic model of the LC is shown in Figure 5.20. The LPN model is *live* and *bounded*. It can be considered as a composition of railway traffic, LC controller and barriers.

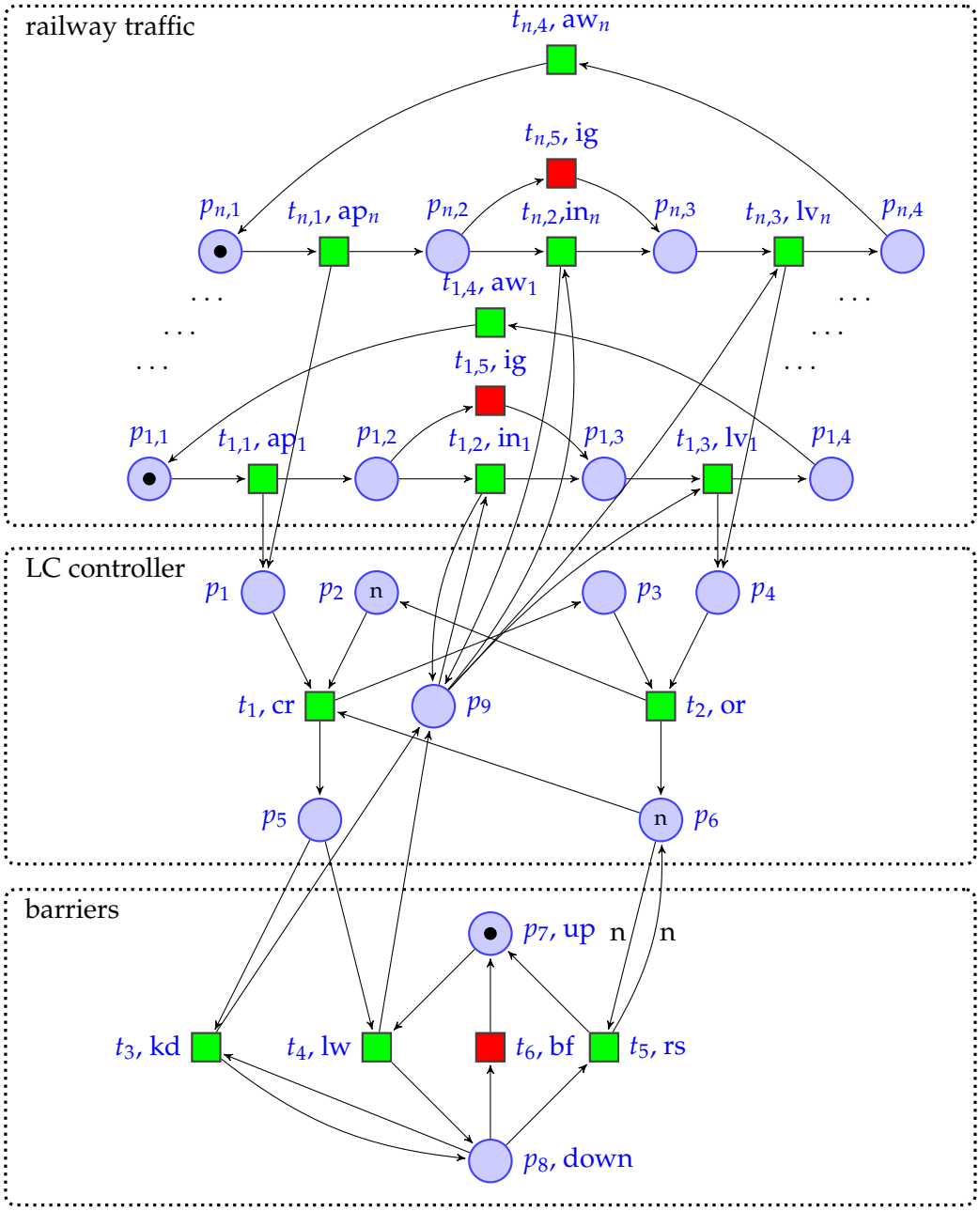


Figure 5.20 – The level crossing benchmark [Liu14]

In this LC benchmark, two classes of faults are modeled by unobservable transitions. The first one is modeled by transition (t_6, "bf") (here "bf" denotes "barrier fault") and $t_6 \in T_f^1$, which indicates a barrier failure that results in a premature barrier raising. The other one is modeled by transition ($t_{i,5}$, ig) (here "ig" denotes "ignore") and $\forall i \in$

$\{1, \cdots, n\}$, $t_{i,5} \in T_f^2$, which indicates that the train may enter the LC crossing zone before the barriers are ensured to be lowered. Each of these two classes of faults can lead to accidents .

This LC model can be extended to a n railway track in order to obtain a large system. The introduction of the LC benchmark is given in Appendix C. For more details for the LC benchmark, readers can refer to the work in [GL16; Liu14; Liu+16].

5.2.1 Monolithic diagnosability analysis of the LC benchmark

In [Bou16; Liu14; Liu+16], some experimental tests were executed for monolithic diagnosability analysis by using different approaches. Readers can refer to these works for the simulation results.

Our approach proposed for monolithic diagnosability analysis can also be applied for analyzing this LC benchmark. Since we have not implemented our algorithms, the experimental test will be a perspective for this work. The theoretical analysis of the monolithic diagnosability is as follows:

- For the fault transition $t_6 \in T_f^1$, in the case of $n-$track LC ($n \geq 1$), t_6 can be fired or not right after firing the sequence $t_{i,1}t_1t_4$. The system can remain F_1-uncertain (F_1 represents the first fault class T_f^1) for as long as 6 steps during the firing of sequence $t_{i,2}t_{i,3}t_{i,4}t_{i,1}t_2t_1$. Afterwards, the system will be normal if t_4 is fired; otherwise the firing of t_3 implies that the fault t_6 has occurred.

- For the fault transition $t_{1,5}$ in T_f^2, in the case $n = 1$, the system is normal if $t_{1,2}$ and $t_{1,3}$ are fired alternatively in any sequence of transitions. Otherwise, the fault transitions t_5 must have been fired. Therefore, the system is diagnosable according to the order of $t_{1,2}$ and $t_{1,3}$.

- For the fault transitions in T_f^2, in the case $n \geq 2$, $\exists i, j \in \{1, \cdots, n\}$ and $i \neq j$, such that:
  The fault transition $t_{i,5}$ can be fired or not right after $t_{i,1}$. For both cases, the sequence $(t_1t_4t_{j,1}t_{j,2}t_{j,3}t_{j,4}t_2t_5)^*$ can be fired, which corresponds to an indeterminate cycle. Therefore, the system is not diagnosable.

However, it is worth noticing that the diagnoser of LC benchmark can be too large to use with the increase of the track number. No matter which approach is used for the diagnosability analysis, a monolithic diagnoser is still required for on-line diagnosis. As it was presented in [Liu14; Liu+16], for $n = 9$, it is already out of memory to build the reachability graph, not to mention the construction of the monolithic diagnoser.

5.2.2 Modular diagnosability analysis of the LC benchmark

In this section, the LC benchmark will be analyzed in the framework of modular diagnosis. The first step is to verify if one can decompose the monolithic model as a sound decomposition. Since in the part of barriers model, the fault transition t_6 belongs to the fault class T_f^1 and in the part of railway traffic model, the fault transitions $t_{1,5}, \cdots, t_{n,5}$ belong to the fault class T_f^2, the intuitive idea is to decompose the monolithic model as three modules: railway traffic module, LC controller module and barriers module.

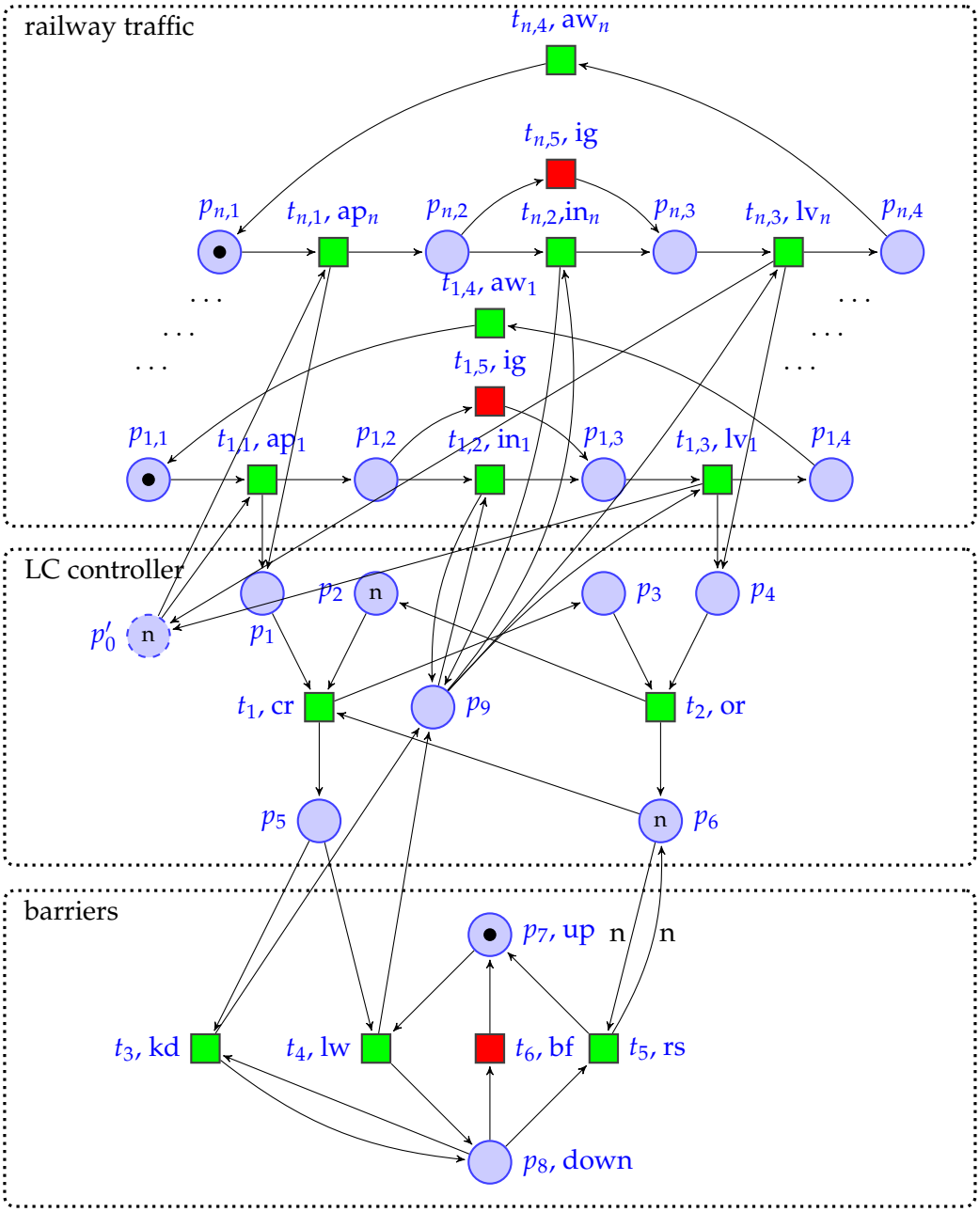


Figure 5.21 – The modified level crossing benchmark

The shared transitions between the modules (transitions $t_{1,1}, t_{1,2}, t_{1,3}, \cdots, t_{n,1}, t_{n,2}, t_{n,3}$

between railway traffic module and LC controller module; transitions t_3, t_4, t_5 between LC controller module and barriers module) are all observable. The only problem to decompose the model is that the transitions $t_{1,1}, \cdots, t_{n,1}$ have no pre-place in the LC controller module to suffice the *sound* decomposition, so the model needs to be modified.

One can add a place p'_0 as a pre-place of the transitions $t_{1,1}, \cdots, t_{n,1}$, with $M(p'_0) = n$. When a train arrives in track j of LC, the transition $t_{j,1}$ is fired, which consumes one token from $p_{j,1}$ and one token from p'_0. In order to make the LC model *live* and *bounded*, when the train leaves the LC system, we need to put one token into p'_0. Therefore, we make the place p'_0 as a post-place of the transitions $t_{1,3}, \cdots, t_{n,3}$. The place p'_0 works as a counter of the trains in the LC system. One token in p'_0 is taken away when a train arrives and one token is put back when a train leaves. Hence, for a $n-$track LC system, the initial number of tokens in p'_0 is n. The modified model is shown in Figure 5.21.

For a $n-$track LC system, there are at most n trains in the railway traffic at the same time. Therefore, if a train arrives in track j i.e., there is one token in place $p_{j,1}$, there exists at least one token in place p'_0 to enable the transition $t_{j,1}$. Hence, by adding the place p'_0, the behavior of the LC system does not change.

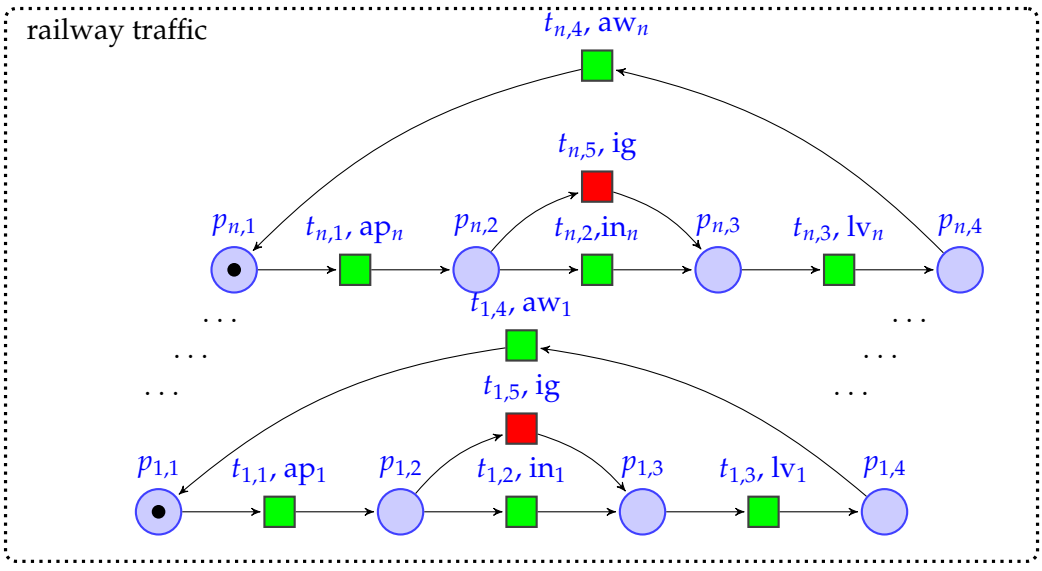


Figure 5.22 – The railway traffic module

By modifying the initial model, it becomes feasible to decompose the monolithic model as a *sound* decomposition. The monolithic model is decomposed into railway traffic module in Figure 5.22 (Module 1), LC controller module in Figure 5.23 (Module 2) and barriers module in Figure 5.24 (Module 3). It is worth noticing that each module is *live* and *bounded*; all the shared transitions are observable; the fault transitions in each module belong to the same fault classes. Therefore, the modular diagnosability analysis approach proposed in Chapter 4 is applicable, as well as the approach proposed in [Con+06].

The local diagnosability of each module is analyzed *a priori*. There is no fault transition

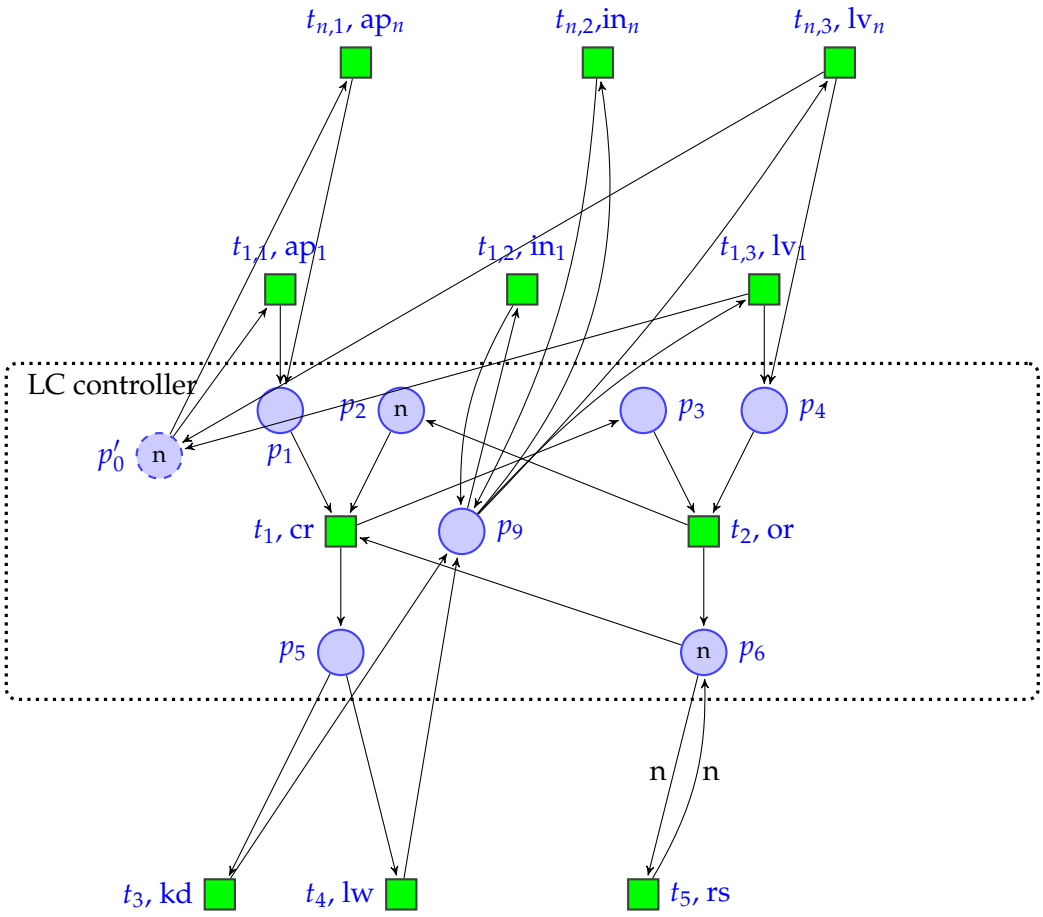


Figure 5.23 – The LC controller module

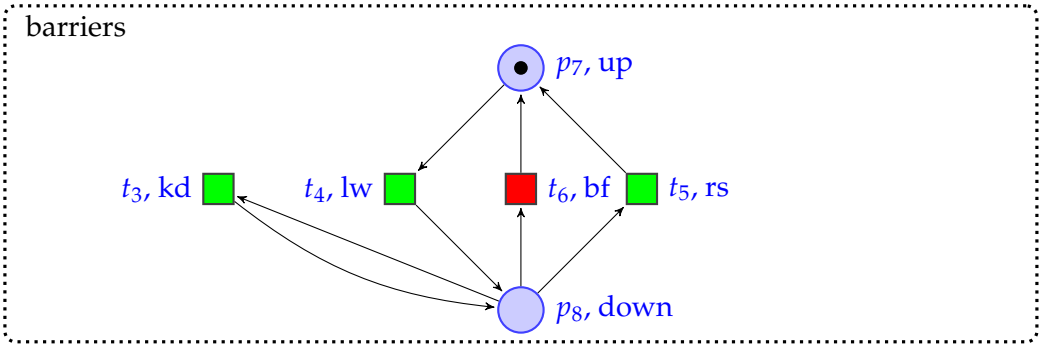


Figure 5.24 – The barriers module

in the LC controller module, so it is locally diagnosable. The barriers module is locally diagnosable w.r.t. the fault transition t_6. The normal behavior of the barriers module is represented by the sequence of events $(lw(kd)^*rs)^*$. If the fault transition t_6 occurs, between two adjacent transition t_4 (event lw), there is no transition t_5 (event rs). It means that if we observe a sequence $(lw(kd)^*rs)^*lw(kd)^*lw$, we are sure that the fault has occurred.

- If $n = 1$ (only one track), the railway traffic module is locally diagnosable, w.r.t. the

fault transition $t_{1,5}$, because if we observe the event ap_1 ($t_{1,1}$) followed by the event lv_1 ($t_{1,3}$), we are sure that the fault has occurred. Therefore, in this case, since all modules are locally diagnosable, the LC system is modularly diagnosable.

- If $n \geq 2$ (more that 2 tracks), the railway traffic module is not locally diagnosable w.r.t. the class of fault transitions $T_f^2 = \{t_{1,5}, \cdots, t_{n,5}\}$. $\exists i, j \in \{1, \cdots, n\}$ and $i \neq j$, the fault transition $t_{i,5}$ can be fired or not right after $t_{i,1}$. For both cases, the sequence $(t_{j,1}t_{j,2}t_{j,3}t_{j,4})^*$ can be fired, which corresponds to an $F^{M_1}-$indeterminate cycle. In another way, there exist two sequences of transitions $\sigma_1 = t_{i,1}(t_{j,1}t_{j,2}t_{j,3}t_{j,4})^*$ and $\sigma_2 = t_{i,1}t_{i,5}(t_{j,1}t_{j,2}t_{j,3}t_{j,4})^*$, such that σ_1 and σ_2 has the same observation; σ_1 is normal but σ_2 contains a faulty transition and can be arbitrarily long after its occurrence. Therefore, the system is not locally diagnosable.

  Afterwards, we need to check if the $F^{M_1}-$indeterminate cycle still survives while analyzing the composed module constructed by module 1 and other modules. By taking advantage of the result in Section 5.2.1, while analyzing the monolithic model, there exists an indeterminate cycle w.r.t. the fault transition $t_{i,5}$ corresponding to the sequence of transitions $(t_1t_4t_{j,1}t_{j,2}t_{j,3}t_{j,4}t_2t_5)^*$. The transitions $t_{j,1}$, $t_{j,2}$, $t_{j,3}$ and $t_{j,4}$ belong to the railway traffic module (Module 1). According to Definition 34 (page 112), there exists an $F^{M_1}-$indeterminate cycle while analyzing the monolithic model. Based on Theorem 7 (page 110, Section 4.1.2), the LC system is not modularly diagnosable.

It is worth noticing that if $n \geq 2$, the railway traffic module is composed by n independent components. Each one models one track of the railway traffic. However, we cannot consider one track model as one module while analyzing the modular diagnosability, because the fault transitions $\{t_{1,5}, \cdots, t_{n,5}\}$ belong to the same fault class (According to the assumptions to apply the modular diagnosability analysis, the fault transitions in each module belong to the same fault class).

- Considering that for $n \geq 2$, let us redefine the fault transitions $\{t_{1,5}, \cdots, t_{n,5}\}$ belong to n different fault classes i.e., for $j \in \{1, \cdots, n\}$, the fault transition $t_{j,5}$ belongs to the fault class $T_f^{2,j}$. In this case, the railway traffic module in Figure 5.22 can be further decomposed into n modules and each module j ($j \in \{1, \cdots, n\}$) pertains to the j^{th} track shown in Figure 5.25.

The monolithic model in Figure 5.21 is decomposed into $n + 2$ modules. As it was analyzed before, the LC controller module and the barriers module are locally diagnosable. The n tracks have the same structure, so it is sufficient to analyze the local diagnosability of one track module. Since the module j (j^{th} track) is locally diagnosable, all the modules are locally diagnosable. Hence, the LC system is modularly diagnosable.

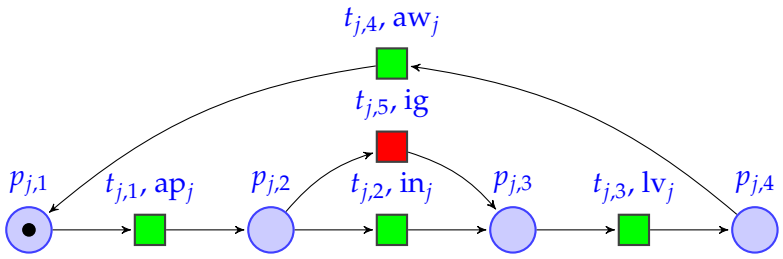


Figure 5.25 – Module j of the level crossing benchmark

Since the LC system becomes modularly diagnosable by redefining the fault classes, the construction of a monolithic diagnoser is not necessary for on-line diagnosis. We can build $n + 2$ local diagnosers and each local diagnoser is sufficient to diagnose the fault of its module. As it was shown in [Liu14; Liu+16], when n is large ($n \geq 7$), building a monolithic diagnoser is not feasible because of the combinatorial explosion problem. However, theoretically, with the increase of n, it is always doable by building the local diagnosers for on-line diagnosis of the modularly diagnosable LC system.

5.3 Synthesis of the two case studies

In this chapter, the diagnosability properties of manufacturing benchmark and multi-track level crossing benchmark are analyzed. We apply our proposed approaches for monolithic diagnosability analysis and modular diagnosability analysis.

While analyzing the manufacturing benchmark, we apply the reduction rules to simplify the initial model *a priori*. The experimental test shows that by using the same approach, the time cost and memory cost of analyzing the reduced model is lower than that of analyzing the initial model. The technique of reduction rules is a complement of most approaches for diagnosability analysis. We apply the sufficient condition for the diagnosability of *safe* and *bounded* PN and the on-the-fly diagnosability analysis using T-invariant to analyze the manufacturing benchmark under different configurations.

Moreover, we analyze the modular diagnosability of the manufacturing benchmark and the multi-track level crossing benchmark. Since the initial models cannot be decomposed as a *sound* decomposition, we modify *a priori* the given model in order to fulfill the assumptions to apply the modular diagnosability analysis. Afterwards, we use our approach (proposed in Section 4.2) to analyze the modular diagnosability of the two benchmarks.

Comparing to the monolithic analysis, if the modular diagnosis can be applied to a large-scale system, the time cost and memory cost are much lower than that of monolithic diagnosis. Moreover, the modular diagnosis (by using the local diagnoser of each module) is sufficient to diagnose all the faults of a modularly diagnosable system.

It is worth noticing that different classification of faults leads to a different decomposition of the monolithic model, as well as a different result for modular diagnosability analysis. It is an open issue to define some rules for the module decomposition and the fault classification. However, for a system (such as the two benchmarks) that contains repeating structures, it is favorable to consider each structure as one module, because all repeating structures are analyzed by analyzing only one of them.

CHAPTER 6

CONCLUSIONS AND PERSPECTIVES

Contents

6.1 Conclusions

This thesis deals with the fault diagnosis of discrete event systems (DES) modeled by labeled Petri nets (LPN). Particularly, the monolithic diagnosability and modular diagnosability issues are studied. Some approaches exploiting the structural properties of LPN are proposed to cope with the computational complexity and combinatorial explosion.

- For monolithic diagnosability analysis: (1) we have first proposed some reduction rules to simplify the given LPN model before analyzing the diagnosability property and we have proved that the diagnosability of reduced LPN model keeps consistent with that of initial LPN model; (2) we have proposed a new sufficient condition for the diagnosability of a *safe* and *live* LPN. We have developed an approaches based on checking the T-invariants to verify this sufficient condition by using linear programing technique; (3) the on-the-fly diagnosability analysis for *bounded* LPN in [Liu+14] has been improved by using minimal explanations, which provides a compact manner to build the state space; (4) the T-invariants are applied to define the priorities of investigating branches in order to find quickly the existing indeterminate cycle; (5) the on-the-fly diagnosability analysis using Verifier Nets has been proposed for diagnosability analysis of both *bounded* and *unbounded* LPN model, which achieves a compromise between computation efficiency and combinatorial explosion limitation. By using these approaches, we have proposed a synthetic solution for diagnosability

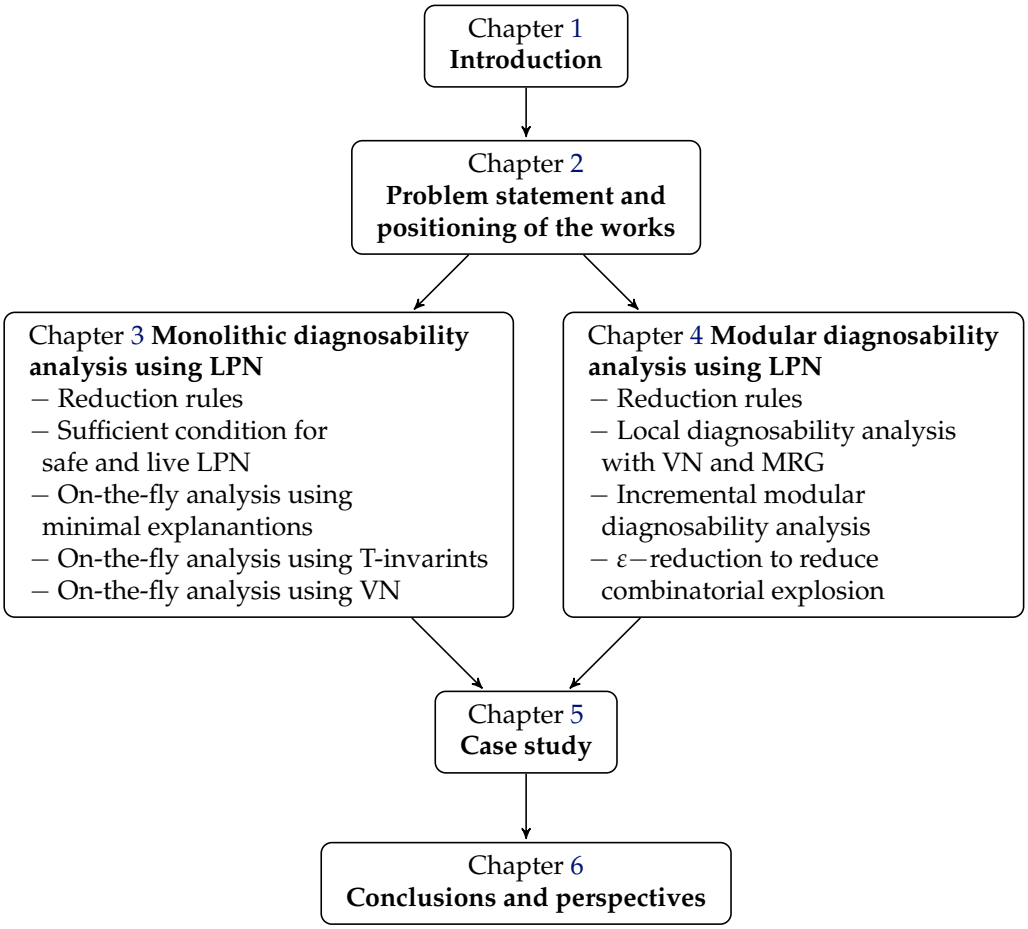


Figure 6.1 – Structure of this thesis

analysis of different types of LPN models. For a given LPN model, we can apply the reduction rules a priori to simplify the LPN model. The sufficient condition for diagnosability analysis of a *safe* and *live* LPN; the on-the-fly diagnosability analysis using minimal explanations and T-invariants can be applied for diagnosability analysis of a *bounded* LPN that does not deadlock after the occurrence of a fault; the on-the-fly diagnosability analysis using VN can be applied for diagnosability analysis of an *unbounded* LPN.

- A new approach is presented for modular diagnosability analysis by releasing the strong assumption of liveness in [Con+06]. We have applied the reduction rules and have proved that by using these rules, the modular diagnosability of reduced system is preserved. Then we decompose a given LPN model as a sound decomposition, which is a collection of LPN modules. The local diagnosability of each module is analyzed by building Verifier Net (VN) and its Modified Reachability Graph (MRG). For each model that is not locally diagnosable, the parallel composition of its MRG and the Reachability Graph (RG) of a composed module is built. The $\varepsilon-$reduction technique is used to reduce the combinatorial explosion. The computational complexity of this approach is lower than that of the approaches in literature.

6.2 Perspectives

In the future, this work on fault diagnosis of DES could be extended on the following perspectives:

1. For monolithic diagnosability analysis:
    - More reduction rules could be proposed with the proof that the diagnosability property is preserved. Some observable transitions that are not ELOT may also be suppressed;
    - A software tool needs to be developed for the on-the-fly diagnosability analysis using minimal explanations and T-invariants, in order to compare the performance with other approaches for diagnosability analysis of large-scale LPN models;
    - For the on-the-fly diagnosability analysis using VN, the priorities of investigating branches could be defined even for an *unbounded* LPN model. It can be defined if we can develop some efficient methods to obtain the repetitive sequences.
2. For modular diagnosability analysis:
    - The assumptions considering that the shared transitions are observable transitions, could be removed and the approaches need to be developed under the assumptions: (i) some share transitions are unobservable; (ii) the shared transition is observable to several modules but is unobservable to the others;
    - New approaches for on-line modular diagnosis need to be developed. The diagnoser-based approach has an important computational complexity and in certain cases, these approaches are no longer applicable.

Résumé

Cette thèse porte sur le diagnostic des systèmes à événements discrets (SED) complexes, modélisés par des réseaux de Petri labellisés (RdP-L). Ce travail est accompli au sein de l'équipe de recherche MOSES (Modèles et Outils formels pour des Systèmes à Evénements discrets Sûrs) du laboratoire CRIStAL (Centre de Recherche en Informatique, Signal et Automatique de Lille, UMR 9189), sous la direction du Prof. Armand Toguyéni et le co-encadrement de Dr. Manel Khlif-Bouassida.

Avec le progrès des nouvelles technologies, les systèmes sont de plus en plus performants et leur complexité est, par conséquent, en augmentation continue. Lorsque au cours d'exploitation, un système peut être soumis à des défaillances critiques voire catastrophiques, il est nécessaire de mettre en œuvre un diagnostic en ligne afin de réagir rapidement pour confiner ces défaillances.

Ce travail de thèse concerne le diagnostic en ligne des SED. Deux propriétés principales caractérisent le fonctionnement d'un SED : (1) le mécanisme de transition d'un état à un autre est déclenché par un événement ; (2) l'espace d'états est discret. Les deux outils formels les plus utilisés, dans le cadre des SED, sont les automates et les réseaux de Petri (RdP). En pratique, de nombreux systèmes peuvent être modélisés en tant que SED : tels que les systèmes de télécommunication, les systèmes de transport, les systèmes d'alimentation ou les systèmes de production manufacturière. Le comportement d'un SED est surveillé par des événements observables, qui peuvent être générés par des capteurs. Néanmoins, il existe également des événements non observables qui ne sont pas directement générés par des capteurs. Dans le cadre de cette thèse, les comportements fautifs sont modélisés par des événements non observables. Le diagnostic est fait en ligne

et l'analyse de la diagnosticabilité est fait hors ligne. Le diagnostic en ligne consiste à déduire l'occurrence de fautes (représentés par des événements non observables) et leur classe en utilisant les événements observables, au cours du fonctionnement du système. La diagnosticabilité représente la capacité du système à détecter une faute dans un délai fini après son occurrence, basée sur des observations. La diagnosticabilité est vérifiée hors ligne (au stade de la conception du système) et doit être garantie avant la mise en œuvre du système.

Dans la littérature, les chercheurs ont proposé des approches pour le diagnostic en ligne et l'analyse de la diagnosticabilité. La première approche proposée est celle du "Diagnostiqueur" proposé par Sampath en 1995. Cette approche est introduite pour le diagnostic et aussi l'analyse de la diagnosticabilité d'un SED modélisé par un automate. La complexité de calcul exponentielle et l'explosion combinatoire sont les problèmes majeurs de cette approche. Par la suite, afin de combattre l'explosion combinatoire, certains chercheurs se sont intéressé à l'exploitation des réseaux de Petri (RdP) qui modélisent des manière plus naturelle le parallélisme et la synchronisation. Cependant, les problème de la complexité de calcul et de l'explosion combinatoire existent encore dans une certaine mesure. Pour faire face à ces problèmes d'analyse de la diagnosticabilité, de nombreuses techniques ont été proposées dans la littérature.

D'autre alternatives ont été proposées pour combattre la complexité. Ainsi, au lieu d'analyser le système monolithique, les architectures de diagnostic décentralisé, de diagnostic modulaire et de diagnostic distribué ont été proposées dans la littérature, afin de réduire la complexité de l'analyse de la diagnosticabilité. L'objectif principal est d'obtenir la même qualité de diagnostic monolithique qu'avec les approches monolithiques sans construire le diagnostiqueur monolithique. Cette

thèse s'intéresse au architecture modulaire, afin de traiter la complexité de calcul et l'explosion combinatoire.

Les contributions de cette thèse consistent en deux parties : une partie sur l'analyse de la diagnosticabilité monolithique et l'autre partie sur l'analyse de la diagnosticabilité modulaire des SED modélisés par les RdP-L.

En ce qui concerne l'analyse de la diagnosticabilité monolithique pour des RdP-L, de nouvelles techniques sont proposées cette thèse, sous des hypothèses différentes :

1. Certaines règles de réduction sont proposées pour simplifier le modèle RdP-L initial avant d'analyser sa diagnosticabilité. En effet, certaines transitions et places pourraient être supprimées sous certaines conditions. Il est prouvé que la diagnosticabilité est préservée après l'utilisation de ces règles de réduction. Ces règles sont un complément pour la plupart des approches existantes d'analyse de la diagnosticabilité basée sur les RdP-L. Le coût de mémoire est plus faible pour analyser le modèle réduit.

2. Une nouvelle condition suffisante de la diagnosticabilité d'un RdP-L *sauf* et *vivant* est proposée, ce qui améliore la condition suffisante de Wen en 2005. Nous avons proposé une méthode basée sur la technique de la programmation linéaire pour vérifier cette condition suffisante. Si la condition suffisante n'est pas vérifiée, il faut construire l'espace d'états et le modèle de diagnostic afin de vérifier l'existence d'un cycle indéterminé.

3. L'analyse à-la-volée de Liu est améliorée en utilisant la technique des explications minimales. Cette technique permet de construire un espace d'états du système et le modèle de diagnostic d'une manière compacte pour réduire

l'explosion combinatoire. Les nouveaux algorithmes sont proposés pour construire à-la-volée et en parallèle le BFG (Basis Fault Marking Graph, en français graphe des marquages étendus basiques) et le BFST (Basis Fault Marking Set Tree, en français arbre d'ensembles de marquages étendus basiques). Il est prouvé que les algorithmes se terminent bien et le verdict de la diagnosticabilité est correct.

4. L'analyse à-la-volée est également améliorée en utilisant des T-semiflows. En utilisant des T-semiflows, les priorités d'exploration des branches sont définies. Des nouveaux algorithmes sont proposés afin de chercher rapidement un cycle indéterminé. Pour un RdP-L non-diagnosticable, l'efficacité de l'analyse à-la-volée est particulièrement améliorée.

5. L'analyse à-la-volée utilisant des Verifier Nets (VN) a également été proposée. Elle peut être utilisée pour les RdP-L bornés et non-bornés. La complexité de calcul est polynomiale pour l'analyse de la diagnosticabilité d'un RdP-L borné. Cette contribution permet d'obtenir un compromis entre l'efficacité du calcul et la limitation d'explosion combinatoire.

En ce qui concerne l'analyse de la diagnosticabilité modulaire, nous avons proposé une nouvelle méthode. Cette méthode peut être utilisée pour analyser la diagnosticabilité modulaire d'un RdP-L qui peut être décomposé sainement.

1. La décomposition modulaire d'un RdP-L est d'abord proposée. Basé sur cette notion, la définition de la diagnosticabilité modulaire d'un modèle de RdP-L est proposée. De plus, l'hypothèse de vivacité dans l'approche de Contant en 2006 est relâchée. Nous supposons qu'il peut exister des blocages dans un module.

2. Certaines règles de réduction sont appliquées pour simplifier le RdP-L avant d'analyser la diagnosticabilité modulaire. Il est prouvé que la propriété de la diagnosticabilité modulaire est préservée après l'utilisation de ces règles de réduction.

3. En analysant la diagnosticabilité locale des modules locaux, une nouvelle approche est proposée basée sur l'approche de VN. Une nouvelle structure appelée le graphe d'accessibilité modifié (GAM) du VN a été définie. Une condition suffisante et nécessaire de la diagnosticabilité locale est donnée. La complexité de cette approche est la même que celle de l'approche du VN.

4. Une nouvelle approche pour l'analyse de la diagnosticabilité modulaire est proposée. La composition parallèle de GAM et le graphe d'accessibilité d'un module composé est construit, afin de vérifier la diagnosticabilité modulaire. Une condition suffisante et nécessaire de la diagnosticabilité modulaire est proposée. La technique de $\varepsilon-$réduction est utilisée pour simplifier les structures, avant de faire la composition parallèle, afin de réduire le problème d'explosion combinatoire. La complexité de cette approche est polynomiale et plus faible que d'autres approches dans la littérature.

Certaines évaluations expérimentales sont fournies pour évaluer nos différentes approches proposées. Elle sont basées sur l'utilisation de deux benchmarks de la littérature : un système de production manufacturière et un passage à niveau ferroviaire. La diagnosticabilité monolithique et la diagnosticabilité modulaire des deux modèles sont analysées. Pour conclure, les résultats montrent que les techniques proposées dans cette thèse, sont capables d'analyser efficacement la diagnosticabilité monolithique ou la diagnosticabilité modulaire des SED complexes.

BIBLIOGRAPHY

[Bas+12] F. Basile, P. Chiacchio, and G. De Tommasi. "On K-diagnosability of Petri Nets via Integer Linear Programming". In: *Automatica* 48.9 (2012), pp. 2047–2058 (pages 55, 69, 197).

[Ber86] G. Berthelot. "Checking Properties of Nets Using Transformation". In: *Advance in Petri Nets*. Springer, 1986, pp. 19–40 (pages 56, 57).

[Ber87] G. Berthelot. "Transformations and Decompositions of Nets". In: *Petri Nets: Central Models and Their Properties* Vol. 254 (1987), pp. 359–376 (pages 56, 57, 59).

[Ber+04] B. Berthomieu, P. Ribet, and F. Vernadat. "The Tool TINA Construction of Abstract State Spaces for Petri Nets and Time Petri Nets". In: *International Journal of Production Research* 42.14 (2004), pp. 2741–2756 (pages 154, 215).

[Bha+95] G. Bhat, R. Cleaveland, and O. Grumberg. "Efficient On-the-Fly Model Checking for CTL". In: *Logic in Computer Science, 1995. LICS'95. Proceedings., Tenth Annual IEEE Symposium on. IEEE*. 1995, pp. 388–397 (pages 44, 50).

[BS02] R. Boel and J. van Schuppen. "Decentralized Failure Diagnosis for Discrete-Event Systems with Costly Communication Between Diagnosers". In: *WODES'02: Sixth International Workshop on Discrete Event Systems, 2002.* 2002, pp. 175–181 (pages 109, 197).

[Bou16] A. Boussif. "Contributions to Model-based Diagnosis of Discrete-Event Systems". PhD thesis. 2016 (pages 150, 151, 169).

[BG15] A. Boussif and M. Ghazel. "Diagnosability Analysis of Input/Output Discrete-Event Systems using Model-checking". In: *5th International Workshop on Dependable Control of Discrete Systems - DCDS'2015*. Elsevier Ltd., 2015, pp. 71–78 (pages 36, 197).

[Bou+15] A. Boussif, M. Ghazel, and K. Klai. "Combining Enumerative and Symbolic Techniques for Diagnosis of Discrete-Event Systems". In: *9th nternational Workshop on Evaluation of Computer and Communication Systems*. 2015 (pages 36, 197).

[Cab+09a] M. Cabasino, A. Giua, and C. Seatzu. "Diagnosability of Bounded Petri Nets". In: *Proc. of the 48th IEEE Conf. on decision and control. Shanghai, China. December*. 2009, pp. 1254–1260 (pages 41, 69, 197).

[Cab+09b] M. Cabasino, A. Giua, and C. Seatzu. "Diagnosis of Discrete Event Systems Using Labeled Petri Nets". In: *DCDS09: 2nd IFAC Workshop on Dependable Control of Discrete Systems (Bari, Italy)*. 2009 (pages 41, 197).

[Cab+10a] M. Cabasino, A. Giua, A. Paoli, and C. Seatzu. "Decentralized diagnosis of Petri nets". In: *2010 American Control Conference, IEEE*. 2010, pp. 3371–3377 (pages 108, 197).

[Cab+10b] M. Cabasino, A. Giua, and C. Seatzu. "Fault Detection for Discrete Event Systems Using Petri Nets with Unobservable Transitions". In: *Automatica* 46.9 (2010), pp. 1531–1539 (page 73).

[Cab+11] M. Cabasino, A. Giua, A. Paoli, and C. Seatzu. "Decentralized Diagnosability Analysis of Discrete Event Systems using Petri Nets". In: *18th IFAC World Congress*. Milano, Italy, 2011, pp. 6060–6066 (pages 109, 197).

[Cab+12] M. Cabasino, A. Giua, S. Lafortune, and C. Seatzu. "A New Approach for Diagnosability Analysis of Petri Nets Using Verifier Nets". In: *IEEE Transactions Automatic Control* 57.12 (2012), pp. 3104–3117 (pages 6, 37, 44, 45, 51–53, 56, 69, 91, 93, 98–100, 129, 146, 147, 197).

[Cab+14] M. Cabasino, A. Giua, and C. Seatzu. "Diagnosis of Discrete Event Systems Using Labeled Petri Nets". In: *IEEE Transactions on Automation Science and Engineering* 11.1 (2014), pp. 144–153 (pages 10, 37, 39, 41, 44, 45, 63, 72, 73, 91, 129, 150, 198).

[Cab+15a] F. Cabral, M. Moreira, O. Diene, and J. Basilio. "A Petri Net Diagnoser for Discrete Event Systems Modeled by Finite State Automata". In: *IEEE Transactions on Automatic Control* 60.1 (2015), pp. 59–71 (pages 37, 197).

[Cab+15b] F. Cabral, M. Moreira, and O. Diene. "Online Fault Diagnosis of Modular Discrete-Event Systems". In: *IEEE 54th Annual Conference on Decision and Control (CDC)* (2015), pp. 4450–4455 (pages 116, 198).

[CL07] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2007 (pages 26, 29, 30, 45).

[ĆP13] G. Ćirović and D. Pamučar. "Decision Support Model for Prioritizing Railway Level Crossings for Safety Improvements: Application of the Adaptive Neuro-fuzzy System". In: *Expert Systems with Applications* 40.6 (2013), pp. 2208–2223 (page 205).

[Con+06] O. Contant, S. Lafortune, and D. Teneketzis. "Diagnosability of Discrete Event Systems with Modular Structure". In: *Discrete Event Dynamic Systems* 16 (2006), pp. 9–37 (pages 6, 103, 104, 110–113, 115, 116, 122, 125, 126, 134, 135, 146, 147, 171, 178, 198).

[Cor+90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction of Algorithms*. Cambridge, MA: MIT Press, 1990 (page 99).

[DA05] R. David and H. Alla. *Discrete, Continuous,and Hybrid Petri Nets*. Springer, 2005 (page 38).

[Deb+00] R. Debouk, S. Lafortune, and D. Teneketzis. "Coordinated Decentralized Protocols for Failure Diagnosis of Discrete Event Systems". In: *Discrete Event Dynamic Systems* 10.1 (2000), pp. 33–86 (pages 104, 105, 107, 198).

[Dot+09] M. Dotoli, M. Fanti, A. Mangini, and W. Ukovich. "On-line Fault Detection in Discrete Event Systems by Petri Nets and Integer Linear Programming". In: *Automatica* 45.11 (2009), pp. 2665–2672 (pages 55, 198).

[Edw+97] S. Edwards, L. Lavagno, E. Lee, and A. Sangiovanni-Vincentelli. "Design of Embedded Systems: Formal Models, Validation, and Synthesis". In: *Proceedings of the IEEE* 85.3 (1997), pp. 366–387 (page 4).

[Fan+13] M. Fanti, A. Mangini, and W. Ukovich. "Fault Detection by Labeled Petri Nets in Centralized and Distributed Approaches". In: *IEEE Transactions on Automation Science and Engineering* 10.2 (2013), pp. 392–404 (pages 121, 198).

[Far+11] G. Faraut, L. Piétrac, and E. Niel. "Process Tracking by Equivalent States in Modal Supervisory Control". In: *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA* (2011) (page 4).

[Fer+92] J. Fernandez, L. Mounier, C. Jard, and T. Jéron. "On-the-fly Verification of Finite Transition Systems". In: *Formal Methods in System Design* 1.2-3 (1992), pp. 251–273 (pages 44, 50, 100).

[GL03] S. Genc and S. Lafortune. “Distributed Diagnosis of Discrete-Event Systems Using Petri Nets”. In: *International Conference on Application and Theory of Petri Nets*. 2003, pp. 316–336 (pages 117, 118, 121, 198).

[GL07] S. Genc and S. Lafortune. “Distributed Diagnosis of Place-Bordered Petri Nets”. In: *IEEE Transactions on Automation Science and Engineering* 4.2 (2007), pp. 206–219 (pages 104, 121, 198).

[Gha17] M. Ghazel. “A Control Scheme for Automatic Level Crossings under the ERTMS/ETCS Level 2/3 Operation”. In: *IEEE Transactions on Intelligent Transportation Systems* (2017) (page 4).

[GE07] M. Ghazel and E.-M. El Koursi. “Automatic Level Crossings : From Informal Functional Requirements ’ Specifications to the Control Model Design”. In: *IEEE International Conference on Systems Engineering*. 2007 (page 205).

[GEK14] M. Ghazel and E.-M. El-Koursi. “Two-Half-Barrier Level Crossings Versus Four-Half-Barrier Level Crossings : A Comparative Risk Analysis Study”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.3 (2014), pp. 1123–1133 (page 205).

[GL16] M. Ghazel and B. Liu. “A Customizable Railway Benchmark to Deal with Fault Diagnosis Issues in DES”. In: *2016 13th International Workshop on Discrete Event Systems, WODES 2016*. Vol. 1. 2016, pp. 177–182 (page 169).

[Gha09] M. Ghazel. “Using Stochastic Petri Nets for Level-Crossing Collision Risk Assessment”. In: *IEEE Transactions on Intelligent Transportation Systems* 10.4 (2009), pp. 668–677 (page 205).

[Giu08] A. Giua. “A Benchmark for Diagnosis”. In: *9th International Workshop on Discrete Event Systems, WODES 2008*. 2008, pp. 1–2 (page 149).

[GS05] A. Giua and C. Seatzu. “Fault Detection for Discrete Event Systems Using Petri Nets with Unobservable Transitions”. In: *Proc. 44th IEEE Conf. on Decision and Control, and the European Control Conference*. 2005, pp. 6323–6328 (page 87).

[Gou+14] H. E. Gougam, A. Subias, and Y. Pencolé. “Discriminability Analysis of Supervision Patterns by Net Unfoldings”. In: *12th IFAC International Workshop on Discrete Event Systems-WODES’14*. Vol. 12. 2. IFAC, 2014, pp. 459–464 (pages 56, 198).

[Gra09] A. Grastien. “Symbolic Testing of Diagnosability”. In: *20th International Workshop on Principles of Diagnosis (DX-09)*. 2009, pp. 131–138 (pages 37, 198).

[Haj+12] S. Hajjar, E. Dumitrescu, and E. Niel. "A Component-based Safe Design Method for Train Control Systems". In: *6th European Congress on Embedded Real-Time Software and Systems*. 2012 (page 4).

[Haj+13] S. Hajjar, E. Dumitrescu, and E. Niel. "Safe Design Method of Embedded Control Systems. Case Study". In: *ournal Européen des Systèmes Automatisés (JESA)* 47 (2013), pp. 403–421 (page 4).

[Hos+13] M. Hosseini, B. Lennartson, M. Cabasino, and C. Seatzu. "A Survey on Efficient Diagnosability Tests for Automata and Bounded Petri Nets". In: *IEEE International Conference on Emerging Technologies and Factory Automation, (ETFA)*. 2013, pp. 1–6 (pages 7, 149–151).

[Jia+01] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. "A Polynomial Algorithm for Testing Diagnosability of Discrete Event Systems". In: *IEEE Transactions on Automatic Control* 46.8 (2001), pp. 1318–1321 (pages 31, 44, 45, 129, 198).

[JB05] G. Jiroveanu and R. Boel. "Distributed Diagnosis for Petri Nets Models with Unobservable Interactions via Common places". In: *IEEE 44th IEEE Conference on Decision and Control*. 2005, pp. 6305–6310 (pages 121, 198).

[JB10] G. Jiroveanu and R. Boel. "The Diagnosability of Petri Net Models Using Minimal Explanations". In: *IEEE Transactions on Automatic Control* 55.7 (2010), pp. 1663–1668 (pages 39, 45, 55, 69, 198).

[KM69] R. Karp and R. Miller. "Parallel Program Schemata: a Mathematical Model for Parallel Computation". In: *Journal of Computer and system Sciences* 3.2 (1969), pp. 147–195 (page 95).

[KG09] L. Khoudour and M. Ghazel. "Towards Safer Level Crossings: Existing Recommendations , New Applicable Technologies and a Proposed Simulation Model". In: *European transport research review* 1.1 (2009), pp. 35–45 (page 205).

[Laf00] S. Lafortune. *UMDES Software Library*. 2000 (page 155).

[Lef14a] D. Lefebvre. "Fault Diagnosis and Prognosis With Partially Observed Petri Nets". In: *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS* 44.10 (2014), pp. 1413–1424 (pages 56, 198).

[Lef14b] D. Lefebvre. "On-Line Fault Diagnosis With Partially Observed Petri Nets". In: *IEEE Transactions on Automatic Control* 59.7 (2014), pp. 1919–1924 (pages 56, 198).

[Lef16] D. Lefebvre. “Diagnosability of Petri Nets with Observation Graphs”. In: *Discrete Event Dynamic Systems: Theory and Applications* 26.3 (2016), pp. 539–559 (pages 55, 198).

[LD07] D. Lefebvre and C. Delherm. “Diagnosis of DES With Petri Net Models”. In: *IEEE Transactions on Automation Science and Engineering* 4.1 (2007), pp. 114–118 (pages 55, 198).

[LS85] N. G. Leveson and J. Stolzy. “Analyzing Safety and Fault Tolerance using Time Petri Nets”. In: *International Joint Conference on Theory and Practice of Software Development*. 1985, pp. 339–355 (pages 206, 209).

[Li+15a] B. Li, M. Khlif-bouassida, and A. Toguyéni. “Diagnosticabilité de Réseaux de Petri Labellisés basée sur les Explications Minimales et les T-semiflots”. In: *10ème Colloque sur la Modélisation des Systèmes Réactifs (MSR 2015)*. 2015 (pages 26, 102).

[Li+15b] B. Li, B. Liu, and A. Toguyéni. “On-the-fly Diagnosability Analysis of Labeled Petri Nets Using Minimal Explanations”. In: *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes - SAFEPROCESS'2015*. 2015, pp. 326–331 (pages 26, 63, 69, 78, 102, 198).

[Li+15c] B. Li, M. Khlif-bouassida, and A. Toguyéni. “On-the-fly Diagnosability Analysis of Labeled Petri Nets Using T-invariants”. In: *5th International Workshop on Dependable Control of Discrete Systems - DCDS'2015*. 2015, pp. 64–70 (pages 26, 56, 63, 82, 102, 198).

[Li+16a] B. Li, M. Khlif-bouassida, and A. Toguyéni. “Diagnosis and Diagnosability Analysis of Labeled Petri Nets Using Reduction Rules”. In: *13th International Workshop on Discrete Event Systems, WODES'2016*. 2016, pp. 171–176 (pages 25, 57, 59, 60, 198).

[Li+16b] B. Li, M. Khlif-bouassida, and A. Toguyéni. “On-the-fly Diagnosability Analysis of LPN Using Verifier Nets”. In: *3rd International Conference on Control and Fault-Tolerant Systems-SYSTOL'16*. 2016, pp. 305–312 (pages 26, 91, 93).

[Li+17a] B. Li, M. Khlif-bouassida, and A. Toguyéni. “Diagnosability of Labeled Petri Nets using Minimal Explanations and T-invariants”. In: *Discrete Event Dynamic Systems, (second review)* (2017) (page 26).

[Li+17b] B. Li, M. Khlif-bouassida, and A. Toguyéni. "On-the-fly Diagnosability ANnalysis of Bounded and Unbounded LPN using Verifier Nets". In: *International Journal of Applied Mathematics and Computer Science, (submitted)* (2017) (page 26).

[Li+17c] B. Li, J. C. Basilio, M. Khlif-bouassida, and A. Toguyéni. "Polynomial Time Verification of Modular Diagnosability of Discrete Event Systems". In: *IFAC2017 World Congress*. 2017 (page 104).

[Li+17d] B. Li, M. Khlif-bouassida, and A. Toguyéni. "Reduction Rules for Diagnosability Analysis of Complex Systems Modeled by Labeled Petri Nets". In: *IEEE Transactions on Automation Science and Engineering, (submitted)* (2017) (page 25).

[Liu14] B. Liu. "An Efficient Approach for Diagnosability and Diagnosis of DES Based on Labeled Petri Nets - Untimed and Timed Contexts". PhD thesis. Ecole Centrale de Lille, 2014 (pages 46, 168, 169, 174, 205).

[Liu+14] B. Liu, M. Ghazel, and A. Toguyéni. "Toward an Efficient Approach for Diagnosability Analysis of DES Modeled by Labeled Petri Nets". In: *13th European Control Conference - ECC'2014*. 2014, pp. 1293–1298 (pages 5, 37, 45, 47, 56, 63, 69, 72–74, 76, 78, 79, 81, 82, 102, 129, 177, 198).

[Liu+16] B. Liu, M. Ghazel, and A. Toguyéni. "Model-Based Diagnosis of Multi-Track Level Crossing Plants". In: *IEEE Transactions on Intelligent Transportation Systems* 17.2 (2016), pp. 546–556 (pages 4, 7, 123, 149, 169, 174).

[Mad+10] A. Madalinski, F. Nouioua, and P. Dague. "Diagnosability Verification with Petri Net Unfoldings". In: *International Journal of Knowledge-Based and Intelligent Engineering Systems* 2 (2010), pp. 49–55 (pages 56, 198).

[MS82] J. Martínez and M. Silva. "A Simple and Fast Algorithm to Obtain All Invariants of a Generalized Petri Net". In: *Informatik-Fachberichte 52: Application and Theory of Petri Nets*. 1982, pp. 301–310 (page 87).

[Mm+13] J. Medina-marin, J. Seck-tuoh mora, N. Hernandez-romero, J. Quezada-quezada, and P. Soto. "Petri Net Reduction Rules Through Incidence Matrix Operations". In: *Proc. of the European Modeling and Simulation Symposium*. 2013, pp. 496–503 (pages 57, 151, 201, 202).

[Mek+12] A. Mekki, M. Ghazel, and A. Toguyéni. “Validation of a New Functional Design of Automatic Protection Systems at Level Crossings with Model-Checking Techniques”. In: *IEEE Transactions on Intelligent Transportation Systems* 13.2 (2012), pp. 714–723 (page 205).

[Mor+11] M. Moreira, T. Jesus, and J. Basilio. “Polynomial Time Verification of Decentralized Diagnosability of Discrete Event Systems”. In: *IEEE Transactions on Automatic Control* 56.7 (2011), pp. 1679–1684 (pages 33, 36, 109, 198).

[Mor+16] M. Moreira, J. Basilio, and F. Cabral. ““Polynomial Time Verification of Decentralized Diagnosability of Discrete Event Systems” versus “Decentralized Failure Diagnosis of Discrete Event Systems”: A critical Appraisal”. In: *IEEE Transactions on Automatic Control* 61.1 (2016), pp. 178–181 (pages 109, 198).

[Mur89] T. Murata. “Petri Nets: Properties, Analysis and Applications”. In: *Proceedings of the IEEE* 77.4 (1989), pp. 541–580 (pages 56, 57).

[MP13] D. Myadzelets and A. Paoli. “Virtual Modules in Discrete-Event Systems: Achieving Modular Diagnosability”. In: *arXiv preprint arXiv:1311.2850*. 2013. arXiv: 1311.2850 (pages 116, 135, 146, 147, 198).

[NN04] M. Nourelfath and E. Niel. “Modular Supervisory Control of an Experimental Automated Manufacturing System”. In: *Control Engineering Practice* 12.2 (2004), pp. 205–216 (page 4).

[Paq+14] D. Paquereau, L. Pietrac, E. Niel, and L. Bouresche. “Determining of Critical and Dreaded States Achieved during Metro Line Supervision”. In: *22nd Mediterranean Conference on Control and Automation, MED 2014*. 2014, pp. 224–230 (page 4).

[PC02] C. Pecheur and A. Cimatti. “Formal Verification of Diagnosability via Symbolic Model Checking”. In: *Workshop on Model Checking and Artificial Intelligence*. 2002 (pages 36, 198).

[Pen00] Y. Pencolé. “Decentralized Diagnoser Approach : Application to Telecommunication Networks”. In: *International Workshop on Principles of Diagnosis (DX'00)*. 2000, pp. 185–192 (pages 108, 198).

[Pen04] Y. Pencolé. “Diagnosability Analysis of Distributed Discrete Event Systems”. In: *International Workshop on Principles of Diagnosis (DX'04)*. 2004, pp. 173–178 (pages 116, 199).

[Pen+15] Y. Pencolé, R. Pichard, and P. Fernbach. "Modular Fault Diagnosis in Discrete-Event Systems with a CPN Diagnoser". In: *9th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes - SAFEPROCESS'2015*. 2015, pp. 470–475 (pages 116, 123, 199).

[Pet62] C. Petri. "Kommunikation Mit Automaten". PhD thesis. 1962 (page 17).

[QK06] W. Qiu and R. Kumar. "Decentralized Failure Diagnosis of Discrete Event Systems". In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 36.2 (2006), pp. 384–395 (pages 33, 109, 199).

[ST15] R. Saddem and A. Toguyéni. "A Modeling Method for the Diagnosis of Embedded Systems based on a Discrete Behavior". In: *15th IFAC Symposium on Information Control Problems in Manufacturing - INCOM 2015*. Vol. 48. 3. 2015, pp. 755–760 (page 4).

[Sam+95] M. Sampath, R. Sengupta, and S. Lafortune. "Diagnosability of Discrete-Event Systems". In: *IEEE Transactions Automatic Control* 40.9 (1995), pp. 1555–1575 (pages 9, 25–28, 30, 36, 44–46, 50, 56, 63, 68, 69, 72, 102, 116, 119, 120, 129, 146, 150, 154, 155, 199, 215).

[Sam+96] M. Sampath, R. Sengupta, S. Lafortune, and K. Sinnamohideen. "Failure Diagnosis Using Discrete-Event Models". In: *IEEE Transactions Automatic Control* 4.2 (1996), pp. 105–124 (pages 25, 199).

[Sch10] K. Schmidt. "Abstraction-based Failure Diagnosis for Discrete Event Systems". In: *Systems & Control Letters* 59.1 (2010), pp. 42–47 (pages 36, 199).

[Sch13] K. Schmidt. "Verification of Modular Diagnosability with Local Specifications for Discrete-Event Systems". In: *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans* 43.5 (2013), pp. 1130–1140 (pages 116, 146, 147, 199).

[Sch+00] S. Schulz, T. Ewing, and J. Rozenblit. "Discrete Event System Specification (DEVS) and StateMate StateCharts Equivalence for Embedded Systems Modeling". In: *Proceedings Seventh IEEE International Conference and Workshopon the Engineering of Computer Based Systems, 2000. (ECBS 2000)* (2000), pp. 308–316 (page 4).

[SE05] S. Schwoon and J. Esparza. "A Note on On-The-Fly Verification Algorithms". In: *Tools and Algorithms for the Construction and Analysis of Systems* (2005), pp. 174–190 (pages 44, 50, 100).

[Sil12] M. Silva. “50 Years after the PhD Thesis of Carl Adam Petri : A Perspective”. In: *11th IFAC Workshop on Discrete Event Systems*. 2012, pp. 13–20 (page 17).

[SW02] R. Su and W. Wonham. “Distributed Diagnosis for Qualitative Systems”. In: *2002 IFAC International Workshop on Discrete Event Systems, WODES'02*. 2002, pp. 169–174 (pages 121, 199).

[SW04] R. Su and W. Wonham. “Hierarchical Distributed Diagnosis under Global Consistency”. In: *2004 IFAC International Workshop on Discrete Event Systems, WODES'04*. 2004, pp. 157–162 (pages 121, 199).

[Tog+03] A. Toguyéni, P. Berruet, and E. Craye. “A Petri Net based Decentralized Synthesis Approach for the Control of Flexible Manufacturing Systems I-Lb”. In: *International Journal of Flexible Manufacturing Systems* 15.1 (2003), pp. 57–85 (page 4).

[Ush+98] T. Ushio, I. Onishi, and K. Okuda. “Fault Detection Based on Petri Net Models”. In: *Proc. of the 1998 IEEE Conf. on systems, man, and cybernetics. San Diego, CA, USA. October*. 1998, pp. 113–118 (pages 9, 44, 55, 199).

[Ven+03] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri. “A Review of Process Fault Detection and Diagnosis Part I: Quantitative Model-based Methods”. In: *Computers and Chemical Engineering* 27.3 (2003), pp. 293–311 (page 4).

[Wan+07] Y. Wang, T. Yoo, and S. Lafortune. “Diagnosis of Discrete Event Systems using Decentralized Architectures”. In: *Discrete Event Dynamic Systems* 17.2 (2007), pp. 233–263 (pages 109, 199).

[WJ05] Y. Wen and M. Jeng. “Diagnosability Analysis Based on T-invariants of Petri Nets”. In: *Networking, Sensing and Control*. 2005, pp. 371–376 (pages 38, 39, 199).

[Wen+05] Y. Wen, C. Li, and M. Jeng. “A Polynomial Algorithm for Checking Diagnosability of Petri Nets”. In: *Proc. SMC05: IEEE Int. Conf. on systems, Man and Cybernetics*. 2005, pp. 2542–2547 (pages 5, 37–39, 69, 70, 101, 199).

[YL02] T. Yoo and S. Lafortune. “Polynomial-Time Verification of Diagnosability of Partially Observed Discrete-Event Systems”. In: *IEEE Transactions Automatic Control* 47.9 (2002), pp. 1491–1495 (pages 33, 44, 45, 129, 146, 150, 154, 155, 199).

[ZL13] J. Zaytoon and S. Lafortune. “Overview of Fault Diagnosis Methods for Discrete Event Systems”. In: *Annual Reviews in Control* 37.2 (2013), pp. 308–320 (page 14).

[Zho+08] C. Zhou, R. Kumar, and R. Sreenivas. “Decentralized Modular Diagnosis of Concurrent Discrete Event Systems”. In: *9th International Workshop on Discrete Event Systems, WODES 2008*. 2008, pp. 388–393 (pages 116, 199).

[Zho+92] M. Zhou, F. Dicesare, and D. Rudolph. “Design and Implementation of a Petri Net based Supervisor for a Flexible Manufacturing System”. In: *Automatica* 28.6 (1992), pp. 1199–1208 (page 4).

APPENDIX A

LITERATURE REVIEW ON UNTIMED DES-BASED DIAGNOSIS

Reference	Automata	PN	Diagnosability	On-line diagnosis	Monolithic architecture	Decentralized architecture	Modular architecture	Distributed architecture	Diagnoser-based	Verifier-based	ILP	Other techniques
[Bas+12]		•	•		•						•	
[BS02]	•			•		•			•			
[BG15]	•		•		•							•
[Bou+15]	•		•	•	•				•			
[Cab+15a]	•			•	•							•
[Cab+09a]		•	•		•				•			
[Cab+09b]		•		•	•				•			
[Cab+10a]		•		•		•			•			
[Cab+11]		•	•			•				•		
[Cab+12]		•	•		•					•		

Table A.1 continued

Reference	Automata	PN	Diagnosability	On-line diagnosis	Monolithic architecture	Decentralized architecture	Modular architecture	Distributed architecture	Diagnoser-based	Verifier-based	ILP	Other techniques
[Cab+14]		•	•		•				•			
[Cab+15b]	•			•			•					•
[Con+06]	•		•				•		•			
[Dot+09]		•		•	•						•	
[Deb+00]	•			•		•			•			
[Fan+13]		•		•				•			•	
[Gou+14]		•	•		•							•
[Gra09]	•		•		•							•
[GL03]		•		•				•	•			
[GL07]		•		•				•	•			
[Jia+01]	•		•		•					•		
[JB05]		•		•				•				•
[JB10]		•	•		•					•		
[LD07]		•		•	•				•			
[Lef14b]		•		•	•						•	
[Lef14a]		•		•	•						•	
[Lef16]		•	•		•							•
[Li+15c]		•	•		•				•			
[Li+15b]		•	•		•				•			
[Li+16a]		•	•	•	•							•
[Liu+14]		•	•	•	•				•			
[Mad+10]		•	•		•							•
[Mor+16]	•		•			•				•		
[Mor+11]	•		•			•				•		
[MP13]	•		•				•		•			
[PC02]	•		•		•							•
[Pen00]	•			•		•			•			

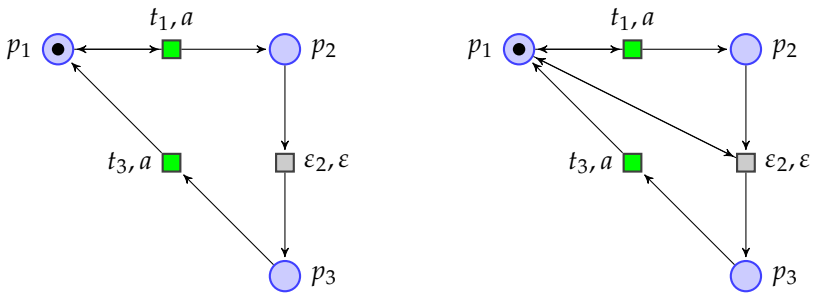
Table A.1 continued

Reference	Automata	PN	Diagnosability	On-line diagnosis	Monolithic architecture	Decentralized architecture	Modular architecture	Distributed architecture	Diagnoser-based	Verifier-based	ILP	Other techniques
[Pen04]	•		•				•			•		
[Pen+15]		•		•			•					•
[QK06]	•		•			•				•		
[Sam+95]	•		•	•	•				•			
[Sam+96]	•		•	•	•				•			
[Sch10]	•		•	•	•							•
[Sch13]	•		•				•					•
[SW02]	•			•				•				•
[SW04]	•			•				•				•
[Ush+98]		•		•	•				•			
[Wen+05]		•	•		•						•	
[WJ05]		•	•		•						•	
[Wan+07]	•		•			•				•		
[YL02]	•		•		•					•		
[Zho+08]	•			•			•					•

APPENDIX B

ALGORITHM FOR REDUCTION RULES

In [Mm+13], the algorithms of the reduction rules used in Section 3.2.1 were proposed. The algorithms are based on the operations of incidence matrix. There exists a problem, while determining whether a transition (or a place) can be reduced or not by using the incidence matrix, because the structure of a PN is consistent with its pre-incidence and post-incidence matrices, but not the incidence matrix. In this section, the reduction rule Fusion of Series Transitions (FST) is used to illustrate this problem.



(1) LPN_1 (2) LPN_2

Figure B.1 – Two examples of LPN

Example 55 *Let us consider the two LPN in Figure B.1. The incident matrix of* LPN_1 *and that of*

LPN_2 are the same:

$$C = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & \varepsilon_2 & t_3 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \end{array};$$

According to the algorithm in [Mm+13], for both LPN models, the transition ε_2 can be suppressed by using the reduction rule FST. However, only the ε_2 in LPN_1 can be suppressed. The ε_2 in LPN_2 cannot be suppressed, because it is the pre- and post-transition of p_1.

Consequently, the determination condition to apply the reduction rules should be provided by using the pre-incidence and post-incidence matrix.

For a transition t_h, which is a regular unobservable transition er an ELOT, the reduction FST can be applied if the pre-incidence and post-incidence matrices are as follows:

$$Pre = \begin{array}{c} \\ p_1 \\ \dots \\ p_i \\ \dots \\ p_j \\ \dots \\ p_m \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & \cdots & t_h & \cdots & t_n \end{array} \\ \begin{bmatrix} \cdots & \cdots & 0 & \cdots & \cdots \\ \cdots & \cdots & 0 & \cdots & \cdots \\ i_1 & \cdots & 1 & \cdots & i_n \\ \cdots & \cdots & 0 & \cdots & \cdots \\ j_1 & \cdots & 0 & \cdots & j_n \\ \cdots & \cdots & 0 & \cdots & \cdots \\ \cdots & \cdots & 0 & \cdots & \cdots \end{bmatrix} \end{array}; Post = \begin{array}{c} \\ p_1 \\ \dots \\ p_i \\ \dots \\ p_j \\ \dots \\ p_m \end{array} \begin{array}{c} \begin{array}{ccccc} t_1 & \cdots & t_h & \cdots & t_n \end{array} \\ \begin{bmatrix} \cdots & \cdots & 0 & \cdots & \cdots \\ \cdots & \cdots & 0 & \cdots & \cdots \\ i_1 & \cdots & 0 & \cdots & i_n \\ \cdots & \cdots & 0 & \cdots & \cdots \\ j_1 & \cdots & 1 & \cdots & j_n \\ \cdots & \cdots & 0 & \cdots & \cdots \\ \cdots & \cdots & 0 & \cdots & \cdots \end{bmatrix} \end{array};$$

The new algorithm for the reduction rule FST is presented in Algorithm 16.

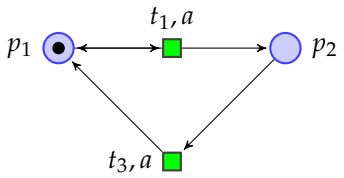


Figure B.2 – Reduced LPN of LPN_1

Example 56 *Let us consider the LPN in Figure B.1(1). The pre-incidence and post-incidence matrices are as follows:*

$$Pre = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & \varepsilon_2 & t_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}; Post = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & \varepsilon_2 & t_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array}.$$

Algorithm 16 Algorithm for the reduction rule FST

Input: pre-incidence matrix Pre and post-incidence matrices $Post$ of LPN that contains m places and n transitions; Output: reduced pre-incidence matrix Pre_r and reduced post-incidence matrices $Post_r$;

1. Step 1: For a transition $t_h \in T$, which is a regular unobservable transition er an ELOT:
2. Step 2: If $(\exists i,\ Pre(i,h) = 1$ and $\forall x \in \{1,\cdots,m\}\backslash\{i\},\ Pre(i,x) = 0) \wedge (\exists j,\ Post(j,h) = 1$ and $\forall y \in \{1,\cdots,m\}\backslash\{j\},\ Pre(j,y) = 0) \wedge (i \neq j)$;
   - Step 2.1: Delete the h column of Pre and $Post$:
     - $A_1 = Pre[1\cdots m, 1\cdots h-1]$, $B_1 = Post[1\cdots m, 1\cdots h-1]$;
     - $A_2 = Pre[1\cdots m, h+1\cdots n]$, $B_1 = Post[1\cdots m, h+1\cdots n]$;
     - $A_d = [A_1\ A_2]$, $B_d = [B_1\ B_2]$;
   - Step 2.2: Sum the row i and the row j:
     - $A_3 = A_d[i, 1\cdots n-1]$, $B_3 = B_d[i, 1\cdots n-1]$;
     - $A_4 = A_d[j, 1\cdots n-1]$, $B_4 = B_d[j, 1\cdots n-1]$;
     - $A_{34} = A_3 + A_4$, $B_{34} = B_3 + B_4$;
   - Step 2.3: Replace the row i by A_{34} and remove the row j from A_d:
     - $A_5 = A_d[1\cdots i-1, 1\cdots n-1]$, $B_5 = B_d[1\cdots i-1, 1\cdots n-1]$;
     - $A_6 = A_d[i+1\cdots j-1, 1\cdots n-1]$, $B_6 = B_d[i+1\cdots j-1, 1\cdots n-1]$;
     - $A_7 = A_d[j-1\cdots m, 1\cdots n-1]$, $B_7 = B_d[j-1\cdots m, 1\cdots n-1]$;
     - $$Pre_r = \begin{bmatrix} A_5 \\ A_{34} \\ A_6 \\ A_7 \end{bmatrix}, Post_r = \begin{bmatrix} B_5 \\ B_{34} \\ B_6 \\ B_7 \end{bmatrix}.$$

According to the condition in Step 2 of Algorithm 16, the transition ε_2 can be suppressed, because

$$Pre(\cdot,\varepsilon_2) = \begin{matrix} \\ p_1 \\ p_2 \\ p_3 \end{matrix} \overset{\varepsilon_2}{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}; Post(\cdot,\varepsilon_2) = \begin{matrix} \\ p_1 \\ p_2 \\ p_3 \end{matrix} \overset{\varepsilon_2}{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}.$$

Using Algorithm 16, the reduced pre-incidence and post-incidence matrices are:

$$Pre_r = \begin{matrix} \\ p_1 \\ p_2 \end{matrix} \begin{matrix} \begin{matrix} t_1 & t_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{matrix}; Post_r = \begin{matrix} \\ p_1 \\ p_2 \end{matrix} \begin{matrix} \begin{matrix} t_1 & t_3 \end{matrix} \\ \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \end{matrix}.$$

The reduced LPN is shown in Figure B.2.

The algorithms for other reduction rules used in Section 3.2.1 are developed similarly, but they are omitted in this thesis.

APPENDIX C

DEVELOPMENT OF THE LC BENCHMARK [LIU14]

A level crossing (LC), is an intersection where a railway line (or multiple railway lines) crosses a road or path at the same level, as opposed to the railway line crossing over or under using a bridge or a tunnel.

C.1 An overview on LC system

In France, there are more than 18,000 LCs. Every day they are traversed by an average of 16,000,000 vehicles and nearly 450,000 closing cycles take place for the passage of trains. LCs are identified as critical safety points in both road and railway infrastructures [Gha09]. On average, 400 people are killed every year in the European Union (EU) [ĆP13]. Therefore, safety of LCs always attracts great attention in railway operation and also in the research area [GE07; KG09; Mek+12].

In this section, we apply our diagnosis techniques to an LC system. We consider a bidirectional multi-track LC (or unidirectional single-track LC for the simple case) and a bidirectional road. Generally, an LC plant is composed of train sensors set on the railway infrastructure, local control system, sound alarm, road lights and barriers, as shown in Figure C.1 [GEK14]. The LC global dynamics can be depicted while considering three subsystems, namely the railway traffic, the LC controller and the barriers, which will be detailed in Appendix C.2.

The logic of a single-track track LC is as follows: when a train approaching the LC is

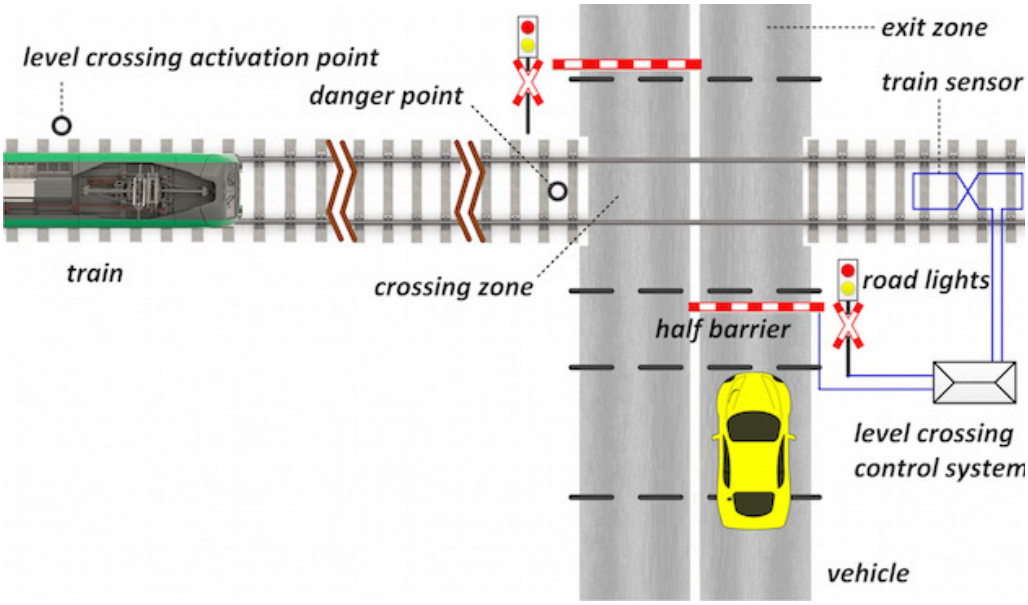


Figure C.1 – The construction of a single-track track LC system

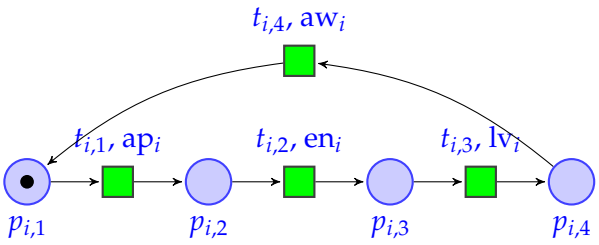
detected by the sensors, the barriers are lowered and the road lights show red. The LC is reopened to road traffic as soon as the train is detected (also by train sensors) out of the crossing zone. As for a multi-track LC, the control on barriers depends on the railway traffic on each line:

- The LC is closed when a train approaching the LC from any line is detected by the train sensors;
- The LC is reopened to road traffic only if no train is still in the crossing zone.

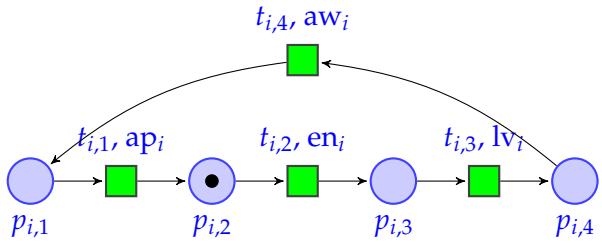
The LC dynamics will be depicted by means of PN models in the next section.

C.2 Modeling of the LC subsystems

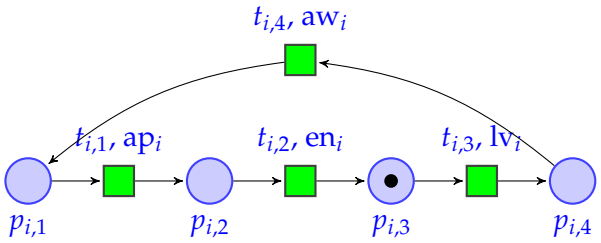
This section presents the modeling of the LC subsystems, namely the railway traffic, the LC controller and the barriers. The n-track LC benchmark will be built based on the single-track LC model [LS85] with some modifications. We will give their corresponding LPN models and operating principles. Note that, as the first step, only the normal behavior will be modeled; some failures will be introduced afterward.



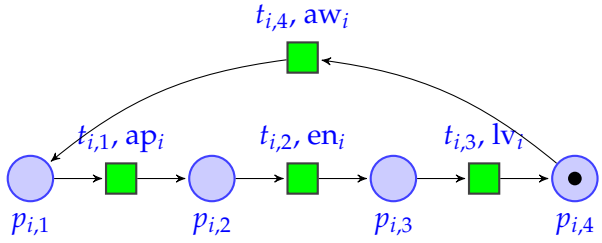
(a) the train is approaching the LC



(b) the train is before the LC



(c) the train is within the LC



(d) the train has left the LC

Figure C.2 – The LPN model for a train passing an LC

C.2.1 Railway traffic

Railway traffic can be modeled as an LPN composed of 4 places and 4 transitions as shown in Figure C.2, where:

- Marked place $p_{i,1}$ (here the subscript i denotes the track index) denotes that a train is approaching the LC, as shown in Figure C.2(a);
- Marked place $p_{i,2}$ denotes that the train has come into the section before the LC, which can be detected by sensor ($t_{i,1}$, ap_i) (here "ap" denotes "approaching"), as shown in Figure C.2(b);
- Marked place $p_{i,3}$ denotes that the train has entered the LC, which can be detected by sensor ($t_{i,2}$, en_i) (here "en" denotes "entering"), as shown in Figure C.2(c);
- Marked place $p_{i,4}$ denotes that the train has left the LC, which can be detected by sensor ($t_{i,3}$, lv_i) (here "lv" denotes "leaving"), as shown in Figure C.2(d). The zone delimited by transition $t_{i,1}$ and $t_{i,3}$ will be called the crossing zone;
- Finally, place $p_{i,4}$ is linked with $p_{i,1}$ through transition ($t_{i,4}$, aw_4) (here "aw" denotes that the train is "away" from the LC), which implies that the next train can approach the LC (when $p_{i,1}$ is marked) only if the previous train has left the LC crossing zone (when $p_{i,4}$ is marked). In other words, there is no overlapping between successive train passages.

C.2.2 LC controller

The LC controller is equipment to collect trains position information from the sensors along the track (in the railway traffic subsystem) and send controlling commands to the barriers and the road lights. The road lights will be omitted in the model as their status can be directly deduced from that of the barriers. It is a processing subsystem between railway traffic and the protection subsystem. The LPN model pertaining to the LC controller is shown in Figure C.3 and the operating principles are explained below:

- When a train enters the LC crossing zone, an alert signal is sent from sensor $t_{i,1}$ to the LC controller (place p_1 will be marked). Then (t_1, cr) (here "cr" denotes "closing request") is fired and a token is added into place p_5, which means that the condition for closing barriers is satisfied. A token is also added to place p_3 upon t_1 firing, to store the information about the train arrival.
- When a train has left the LC crossing zone, its position is detected by sensor $t_{i,3}$ and this information is sent to the LC controller (place p_4 will be marked). Then (t_2, or) (here "or" denotes "open request") can be fired and a token is added into place p_6, which means that the condition for reopening barriers is satisfied.

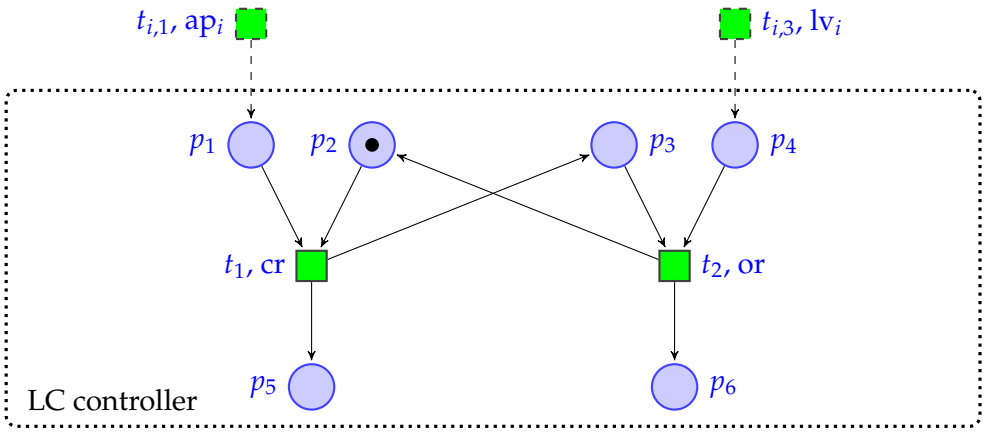
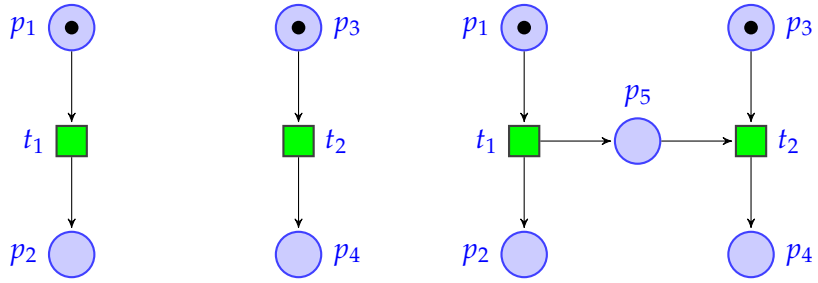


Figure C.3 – The LPN model for LC controller

The LC controller holds, among others, a component called *interlock* [LS85]. An interlock can be a hardware or a software mechanism for ensuring correct sequences of events.

The LPN model for an interlock is shown in Figure C.4. In order to make sure that t_1 has to fire before t_2, a new place p_5 is added as an output place of t_1 and as an input place of t_2, as shown in Figure C.4(b). In other words, the introduction of the interlock (place p_5 and its input/output arcs) ensures that the firing of t_2 is conditioned by the firing of t_1.



(a) without interlock (b) an interlock

Figure C.4 – The PN model for an interlock

In a given system, multiple interlocks may exist for ensuring the order of events in some sequences. For example, two interlocks in the LC controller module exist, as shown in Figure C.3: the one is formed by $t_1 \rightarrow p_3 \rightarrow t_2$ ensuring the firing priority of t_1 over t_2; and the other by $t_2 \rightarrow p_2 \rightarrow t_1$ ensuring the firing priority of t_2 over t_1 after t_1 has been first fired. Such a double-interlock can make sure that t_1 and t_2 fire alternatively. In

practice, this means that the LC may be closed only if it was open and reopened only if it was closed.

C.2.3 Barriers subsystem

The barriers are a subsystem passively responding to the commands from the LC controller. The barrier state switches between "up" (place p_7 is marked) and "down" (place p_8 is marked), i.e., the intermediary positions are ignored. The barriers can be set to "down" (resp. "up") to prevent (resp. permit) vehicles from crossing only if the closing (resp. reopening) condition is satisfied. Here p_7 and p_8 are mutually exclusive, i.e., they cannot be marked at the same time, since a barrier can be only up or down. The LPN model for the barrier system is given in Figure C.5, where the labels of t_7 and t_8 transitions denote "lower" and "raise" respectively.

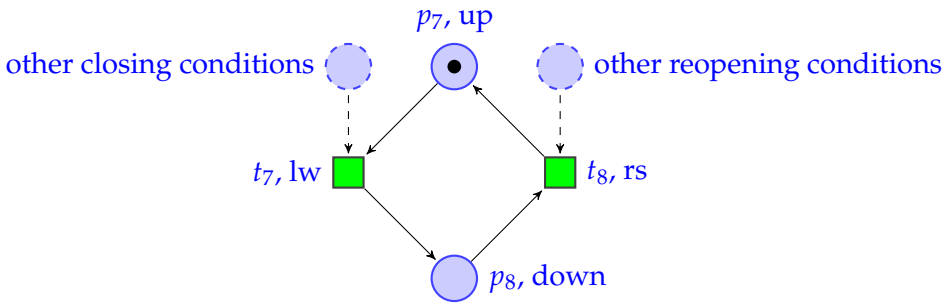


Figure C.5 – The LPN model for a barrier system

C.3 Single-track LC model

After having set up the models for the three LC subsystems, let us now establish the global single-track LC model depicted in Figure C.6.

The railway traffic subsystem "communicates" with the LC controller through the train sensors which send train position information. The LC controller sends "close" or "open" command to switch the "up" and "down" states of the barriers. Place p_9, together with transitions t_4 and $t_{i,2}$, forms an interlock ensuring that normally the barriers must be well lowered (transition t_4 has been fired) before the train enters the LC (transition $t_{i,2}$ is fired).

In the LC, there are two classes of faults which are denoted by red colored transitions in Figure C.7: the first one is modeled by transition ($t_{i,5}$, ig) (here "ig" denotes "ignore")

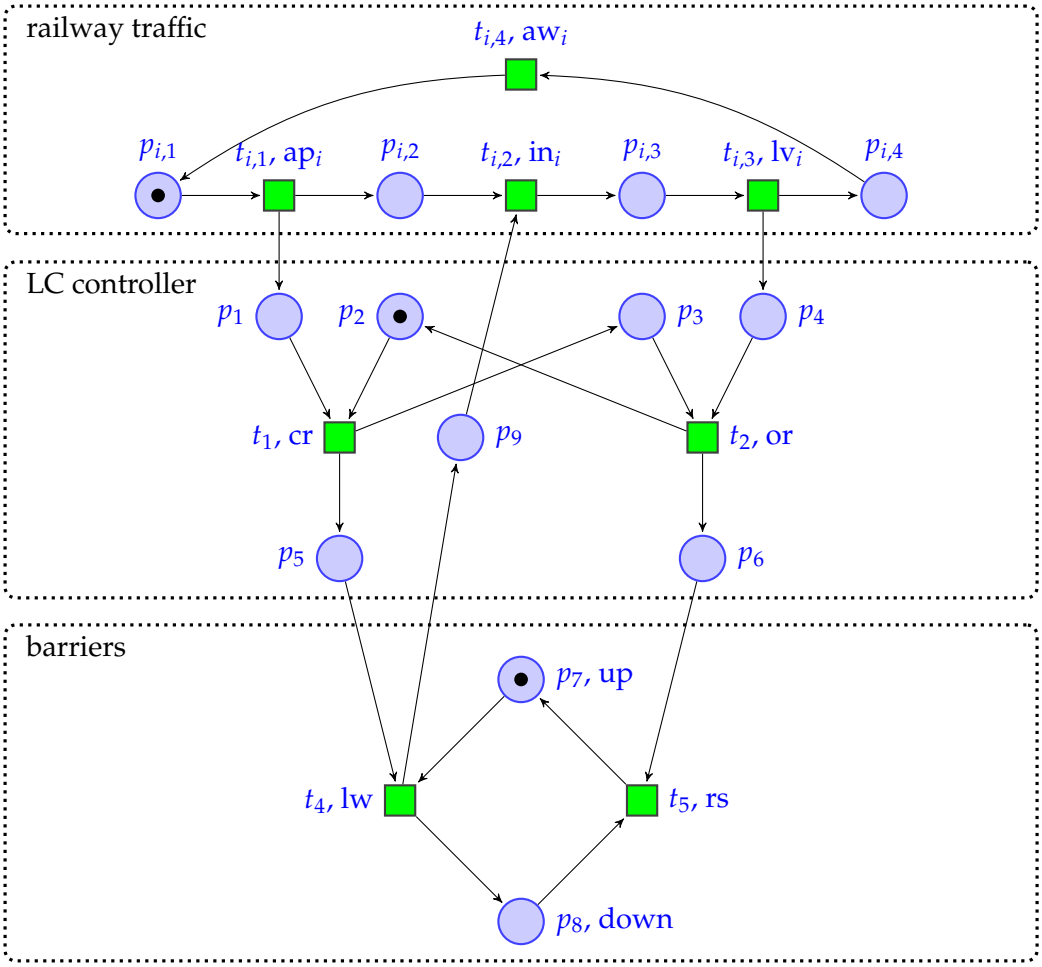


Figure C.6 – A single-track LC

indicating that the train may enter the LC crossing zone before the barriers are ensured to be lowered; the other modeled by transition (t_6, bf) (here "bf" denotes "barrier fault") indicating a barrier failure that results in a premature barrier raising. Each of these two faults can induce a train-car collision.

Note that compared with the model shown in Figure C.3, there are two more arcs into and out of place p_9: the arc from $t_{i,2}$ to p_9 ensures that p_9 is remarked after the firing of $t_{i,2}$; the other arc from p_9 to $t_{i,3}$ takes p_9 as one of the conditions for firing $t_{i,3}$. Both of the two arcs ensure the boundedness of the LPN model. More precisely, the LPN here is 1-bounded (or n-bounded for the n-track LC model afterward).

In the following section, we will introduce a more general LPN model for the LC

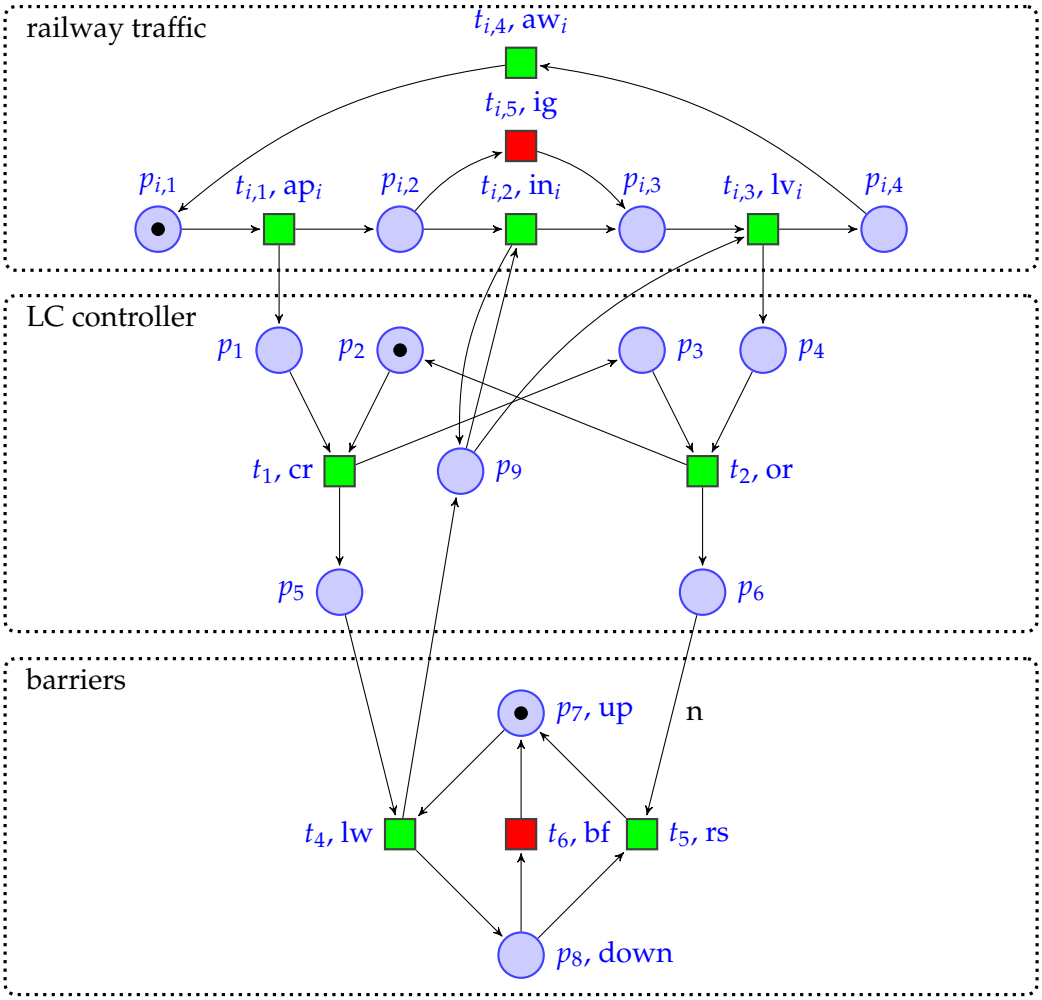


Figure C.7 – A single-track LC with two classes of faults

system, while taking into account n railway lines.

C.4 n-track LC model

Figure C.7 describes a global LPN model for a unidirectional signle-track LC. Based on this model, a more general model is given in Figure C.8 – involving n railway tracks, which can be obtained from the single-track LC model while fulfilling the following controlling rules under a nominal situation:

- The LC must be closed if any approaching train is detected in any line;

- The LC can be reopened if there is no train in the "within" or "before" sections in any line.

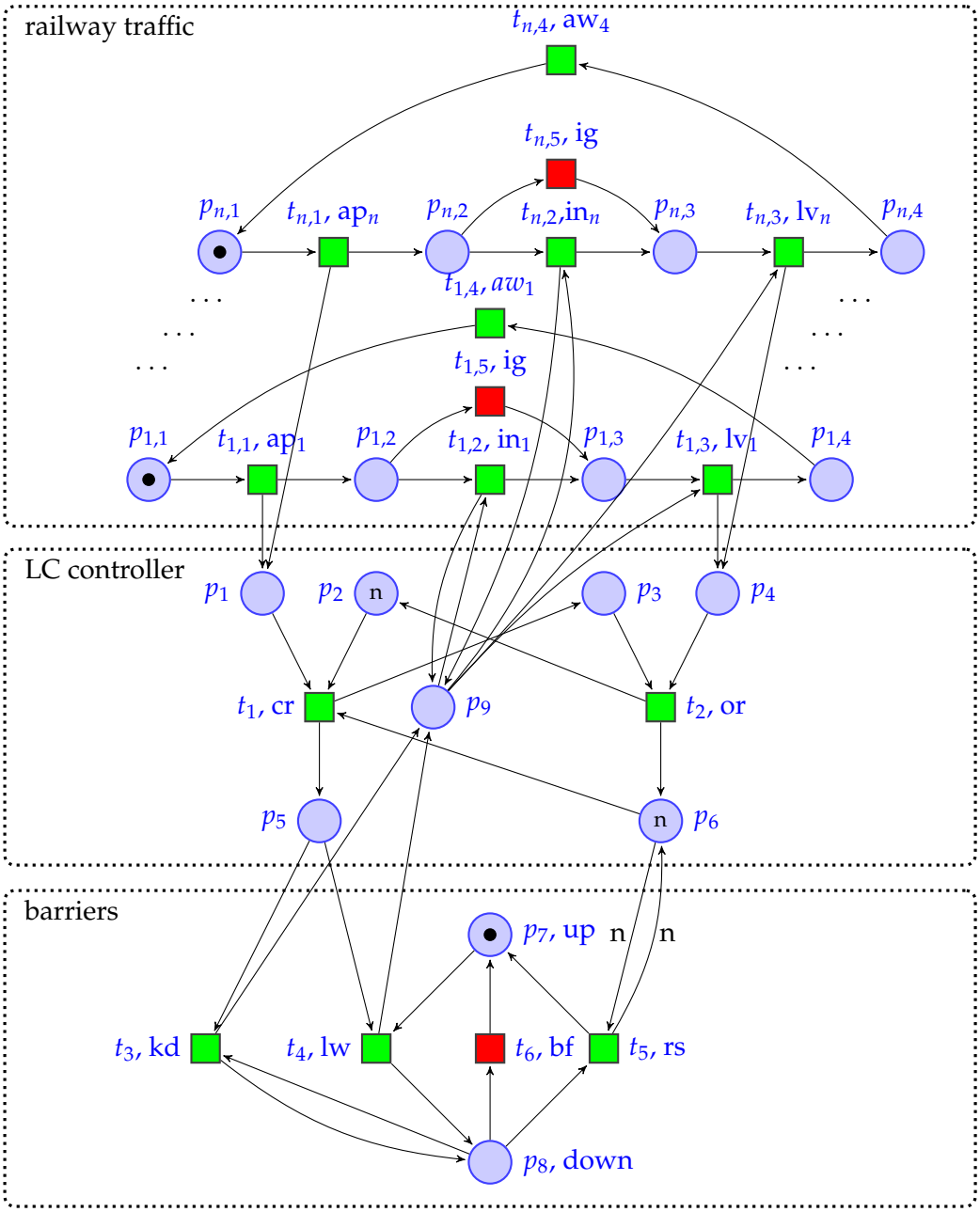


Figure C.8 – n-track LC benchmark

In other terms, the above rules eliminate all the possibilities that the collision between railway and road traffic may take place.

Compared with the single-track LC (cf. Figure C.7), there are several changes when generating the n-track LC model:

- Transition t_3 is newly added. In the n-track LC model, t_3 can be fired if both places p_5 and p_8 are marked. This means that if there is an LC closing request from one of the n lines (place p_5 is marked), whereas the barriers are already in the low position (place p_8 is marked) due to a previous closing command from any other line. Then the barriers shall remain down (transition t_3 fires for clearing the request from marked place p_5 while keeping the token in p_8).
- Place p_2 is marked with n tokens to ensure that at most n closing requests can be proceeded by the LC controller (place p_1 is n-bounded).
- Place p_6 is also marked with n tokens and denotes the reopening condition. Each firing of t_1 removes a token from p_6 and puts a token into p_3, meaning that the LC cannot be reopened when at least one closing request is proceeded by the LC controller. The LC can be reopened only if p_6 is n-marked, i.e., all the trains have passed the LC and their closing LC requests have already been treated by the LC controller.
- The two arcs linking t_5 and p_6 have a weight of n. t_5 can be fired only if p_6 holds the reopening condition (n tokens), i.e., no train is still in the crossing zone. The firing of t_5 also returns n tokens to p_6 to indicate that at most n closing requests can be treated (as no train is still crossing on any of n tracks).
- The arcs linking $t_{i,2}$ to p_9 and p_9 to $t_{i,3}$ ensure that, whether the train passes the LC normally ($t_{i,2}$ is fired) or upon a fault "ig" ($t_{i,5}$ is fired), the token in p_9 will be removed when the train leaves the LC. This ensures the boundedness of the LPN model, since, without these two arcs, the LPN will be unbounded due to the unbounded place p_9.

In order to obtain sufficiently large LC models for analysis, one can add as many "railway traffic" blocks as necessary and connect them with the "LC controller" and "barriers" blocks in the same way.

In this global model, all the transitions are observable, but the faulty transitions, i.e., $T_o = T \backslash T_u$ and $T_u = T_f = \{t_6\} \cup (\cup_i \{t_{i,5}\})$.

The n-track LPN model can be rather big when n takes great values. The state space of the corresponding LPN models for the various values of n can be calculated by the TINA tool [Ber+04], as shown in Table C.1, where:

- n denotes the number of tracks in the n-track LC;
- $|P|$ denotes the number of places in the LPN;
- $|T|$ denotes the number of transitions in the LPN;
- $|A|$ denotes the number of arcs in the reachability graph (RG), i.e., the number of automaton arcs in the diagnoser approach [Sam+95].
- $|R|$ denotes the number of nodes in the RG, i.e., the number of automaton states when analyzing diagnosability with the diagnoser approach;
- $\mathcal{T}_T$ denotes the time used for computing the RG (here the value of $|R|$ and $|A|$) of the PN by means of TINA on an Inter Mac (CPU: 2.8 GHz, RAM: 16 GB).

Table C.1 – Some figures about the state space of the various LC models

n	$\|P\|$	$\|T\|$	$\|A\|$	$\|R\|$	$\mathcal{T}_T$
1	13	11	28	24	<1s
2	17	16	540	216	<1s
3	21	21	6,256	1,632	<1s
4	25	26	56,704	11,008	<1s
5	29	31	442,880	68,608	2s
6	33	36	3,126,272	403,456	11s
7	37	41	20,500,480	2,269,184	140s
8	41	46	127,074,304	12,320,768	29m
9	45	51	o.m.	o.m.	o.m.

Note: o.m. = out of memory

Recall here that not the whole state space will be generated while using our on-the-fly technique. However, the RGs are generated in order to transform them into the input files (language-equivalent automata) for UMDES.

As shown in Table C.1, the size of the RG grows very quickly as n increases, since places p_2 and p_6 can hold as many as n tokens, due to which so many markings exist.

Titre: Diagnostic et Diagnosticabilité des Systèmes à Événements Discrets Complexes Modélisés par des Réseaux de Petri Labellisés

Cette thèse porte sur le diagnostic des systèmes à événements discrets modélisés par des Réseaux de Petri labellisés (RdP-L). Les problèmes de diagnostic monolithique et de diagnostic modulaire sont abordés. Des contributions sont proposées pour résoudre les problèmes d'explosion combinatoire et de complexité de calcul.

Dans le cadre de l'analyse de la diagnosticabilité monolithique, certaines règles de réduction sont proposées comme un complément pour la plupart des techniques existantes de l'analyse de la diagnosticabilité, qui simplifient le modèle RdP-L tout en préservant sa propriété de diagnosticabilité. Pour un RdP-L *sauf* et *vivant*, une nouvelle condition suffisante pour la diagnosticabilité est proposée. Pour un RdR-L *borné* et non bloquant après l'occurrence d'une faute, l'analyse à-la-volée est améliorée en utilisant la notion d'explications minimales qui permettent de compacter l'espace d'état ; et en utilisant des T-semiflots pour trouver rapidement un cycle indéterminé. Une analyse à-la-volée utilisant Verifier Nets (VN) est proposée pour analyser à la fois les RdP-L *bornés* et *non-bornés*, ce qui permet d'obtenir un compromis entre efficacité du calcul et limitation des explosions combinatoires.

Dans le cadre de l'analyse de la diagnosticabilité modulaire, une nouvelle approche est proposée pour les RdP-Ls décomposés. Les règles de réduction, qui préservent la propriété de la diagnosticabilité modulaire, sont appliquées pour simplifier le modèle initial. La diagnosticabilité locale est analysée en construisant le VN et le Graphe d'Accessibilité Modifié (MAG) du modèle local. La diagnosticabilité modulaire est vérifiée en construisant la composition parallèle du MAG et des graphes d'accessibilités d'autres modules du système. La complexité de calcul est inférieure à celles des autre approches dans la littérature. D'autre part, l'explosion combinatoire est également réduite en utilisant la technique de $\varepsilon-$réduction.

Mots clés : Diagnostic de fautes, Systèmes à événements discrets (SED), Réseaux de Petri labellisés (RdP-L), Diagnosticabilité monolithique, Diagnosticabilité modulaire, Diagnostic des SED.

Title: Diagnosis and Diagnosability of Complex Discrete Event Systems Modeled by Labeled Petri Nets

This thesis deals with fault diagnosis of discrete event systems modeled by labeled Petri nets (LPN). The monolithic diagnosability and modular diagnosability issues are addressed. The contributions are proposed to reduce the combinatorial explosion and the computational complexity problems.

Regarding monolithic diagnosability analysis, some reduction rules are proposed as a complement for most diagnosability techniques, which simplify the LPN model and preserve the diagnosability property. For a *safe* and *live* LPN, a new sufficient condition for diagnosability is proposed. For a *bounded* LPN that does not deadlock after a fault, the on-the-fly diagnosability analysis is improved by using minimal explanations to compact the state space; and by using T-invariants, to find quickly an indeterminate cycle. An on-the-fly diagnosability analysis using Verifier Nets (VN) is proposed to analyze both *bounded* and *unbounded* LPN, which achieves a compromise between computation efficiency and combinatorial explosion limitation.

Regarding modular diagnosability analysis, a new approach is proposed for decomposed LPNs model. Reduction rules, that preserve the modular diagnosability property, are applied to simplify the model. The local diagnosability is analyzed by building the VN and the Modified Reachability Graph (MRG) of the local model. The modular diagnosability is verified by building the parallel composition of the MRG and the reachability graphs of other modules of the system. We prove in this study that the computational complexity of our approach is lower than existing approaches of literature. The combinatorial explosion is also reduced by using the $\varepsilon-$reduction technique.

Keywords: Fault diagnosis, Discrete event systems (DES), Labeled Petri nets (LPN), Monolithic diagnosability, Modular diagnosability, DES diagnosis.