Università degli Studi di Cagliari

西安电子科技大学
XIDIAN UNIVERSITY

DOTTORATO DI RICERCA

INGEGNERIA ELETTRONICA E INFORMATICA

XXX Ciclo

TITOLO TESI

Observation Structures and Opacity Problems in Discrete Event Systems

Settore scientifico disciplinari di afferenza

ING-INF/04 AUTOMATICA

Presentata da: Yin TONG

Coordinatore Dottorato: Prof. Fabio ROLI

Tutor: Prof. Alessandro GIUA
Prof. Zhiwu LI

Esame finale anno accademico 2016 – 2017
Tesi discussa nella sessione straordinaria giugno 2017

SOMMARIO

La *sicurezza* è un fondamentale requisito di sistemi quali l'Internet delle cose (Internet of things), le reti di comunicazione e i sistemi informatici. In tali sistemi, alcune informazioni (dette *segreti*) non devono essere corrotte o acquisite da persone non autorizzate (dette *intrusi*). Motivati da questa esigenza di sicurezza delle informazioni, negli ultimi anni numerose nozioni di segretezza sono state formalizzate, quali la *non-inferenza*, l'*anonimia*, l'*opacità*, etc. Tra queste, si è dimostrato che l'opacità è più generale di altre proprietà, consentendo ad esempio di descrivere anche non-inferenza e anonimia. L'opacità implica l'assenza di un particolare flusso di informazioni, o più precisamente, l'impossibilità per un intruso di determinare la veridicità di un predicato che descrive il segreto sulla base delle sue osservazioni. Nei sistemi ad eventi discreti (SED), a seconda della definizione di segreto, l'opacità viene tipicamente classificata in *opacità dello stato corrente* (current-state opacity), *opacità dello stato iniziale* (initial-state opacity), *opacità del linguaggio* (language-based opacity).

L'evoluzione di un sistema non è solitamentedirettamentemisurabile da un osservatore esterno (compreso l'intruso) a causa delle limitazioni insite nei sensori e nei canali di comunicazione, ma può solo essere parzialmente osservata. Nel caso dei sistemi ad eventi discreti sono necessari modelli che consentano di descrivere precisamente diverse modalità di osservazione dei sistemi. Al contempo, a seguito del problema dell'esplosione dello spazio di stato, l'analisi dell'opacità attraverso l'enumerazione di tutti gli stati non è applicabile a sistemi di elevate dimensioni. È necessario sviluppare algoritmi di verifica più efficaci dal punto di vista computazionale. Infine, dato un sistema che non è opaco, un ulteriore problema consiste nel progettare un controllore che ne restringa il comportamento in modo tale che il sistema controllato sia opaco e massimamente permissivo.

Questa tesi si focalizza sulle strutture di osservazione e sui problemi di opacità nei SED. I principali risultati della ricerca sono brevemente riassunti nei seguenti punti.

1. Le reti di Petri sono un formalismo grafico e matematico applicabile a molti sistemi e uno strumento efficace per descrivere e studiare i SED. In questa tesi vengono introdotte due classi di reti di Petri: le reti di Petri etichettate con uscite (labeled Petri nets with outputs) e le reti di Petri etichettate adattative (adaptive labeled Petri nets), al fine di meglio formalizzare e generalizzare le principali strutture di osservazione

delle reti di Petri. Paragonate con altri modelli di reti di Petri, le due classi di generatori hanno generalmente un più alto potere di modellazione. Allo scopo di definire i legami tra i diversi formalismi, noi presentiamo anche delle procedure generali per la trasformazione tra modelli e degli algoritmi che convertano, se possibile, una struttura in un'altra struttura.

2. L'opacità dello stato corrente, l'opacità dello stato iniziale e l'opacità del linguaggio sono definite formalmente nell'ambito delle reti di Petri. Successivamente, vengono studiate le loro differenze rispetto alle proprietà analoghe riferite agli automi. Inoltre, la nozione di opacità del linguaggio viene generalizzata al caso di opacità stretta del linguaggio per affrontare il caso in cui l'intruso sia solo interessato ad un sottoinsieme di transizioni.

3. È noto che la verifica dell'opacità dello stato corrente, dello stato iniziale e del linguaggio sono problemi decidibili nei SED a stati finiti. Tuttavia, la decidibilità di tali problemi nell'ambito delle reti di Petri richiedeva ulteriori indagini. In questa tesi è stato dimostrato che tali probleminon sono decidibili per sistemi di reti di Petri arbitrari. Ossia, non esiste alcun algoritmo generale che fornisca la risposta corretta al problema di stabilire se un dato sistema di rete di Petri è opaco rispetto allo stato corrente (o allo stato iniziale, o al linguaggio). Pertanto, ha senso restringere l'attenzione ai soli sistemi limitati.

4. Per i sistemi limitati, i metodi esistenti di verifica dell'opacità richiedono una enumerazione dell'insieme degli stati consistenti con una data osservazione. Ovviamente, per sistemi di grandi dimensioni il problema dell'esplosione dello stato diventa insormontabile. Traendo beneficio dalla nozione di marcatura di base, in questa tesi viene proposto un metodo praticamente efficiente per la verifica dell'opacità dello stato corrente e dello stato iniziale. Tale approccio consente di evitare l'enumerazione esaustiva dello spazio di stato, limitando l'enumerazione ad un solo sottoinsieme dello spazio di stato, ossia le marcature di base. Tutte le altre marcature possono poi essere descritte a partire dalle marcature di base, attraverso un insieme di vincoli lineari. Risolvendo un certo numero di problemi di programmazione lineare intera, è possibile quindi verificare l'opacità dello stato corrente e dello stato iniziale. Inoltre, la definizione di una struttura, detta *verifier*, consente di analizzare l'opacità stretta del linguaggio. Date le reti che modellano il sistema e il segreto, il verifier sincronizza il sistema e il segreto tenendo traccia di sia delle sequenze che appartengono al segreto

che di quelle che non vi appartengono. In particolare, grazie alla nozione di spiegazioni minime, non vi è necessità di enumerare le sequenze segrete e non. Pertanto l'approccio proposto si rivela più efficace di altri metodi in letteratura.

5. Infine, dato un sistema modellato mediante un automa a stati finiti non opaco rispetto allo stato corrente e ad un dato segreto, viene proposto un approccio per progettare un supervisore ottimo che restringe il comportamento del sistema al fine di assicurare l'opacità del sistema controllato rispetto allo stato corrente. In altri articoli in letteratura viene assunto che l'insieme degli eventi osservabili dall'intruso sia incluso nell'insieme degli eventi osservabili dal supervisore, o viceversa. Chiaramente tale relazione di inclusione può non valere in generale. Pertanto, in questa tesi si considera il caso in cui i due insiemi non sono necessariamente comparabili. Una nuova struttura della *I-osservatore aumentato* (*augmented I-observer*) viene definita per calcolare il sottolinguaggio supremo generato dal sistema che non lascia trapelare il segreto. Sulla base dell'I-osservatore aumentato, è possibile sintetizzare un insieme di supervisori localmente ottimi che garantiscono che il sistema controllato sia massimamente permissivo e opaco rispetto allo stato corrente e ad un dato segreto.

Infine, i contributi della tesi sono ricapitolati e sono indicate le linee di ricerca futura relative alle strutture di osservazione delle reti di Petri e ai problemi di opacità nei SED.

Parole Chiave: Sistema a eventi discreti, reti di Petri, opacità, sicurezza dell'informazione, automa, controllo supervisivo

ABSTRACT

Security is one of the most important properties of systems such as Internet of things, communication networks, and computer systems. In these systems, some information (called secret) should not be corrupted or acquired by unauthorized people (called intruders). Motivated by the concern of information security, in recent years many notions of secrecy have been formulated, such as non-inference, anonymity, and opacity, etc. Among them, opacity is proven capable of formulating some other security properties: for instance, non-inference and anonymity. Opacity formalizes the absence of information flow, or more precisely, the impossibility for an intruder to infer the truth of the secret based on its observation. In discrete event systems (DESs), depending on the definition of secret, opacity is usually categorized as *current-state opacity*, *initial-state opacity*, and *language-based opacity*.

Limited by the sensors or communication, the evolution of a system may not be completely detected by outsiders, including the intruder. Thus, it is important to have powerful DES models that are capable of describing different observation structures of systems. Meanwhile, due to the state explosion problem, opacity analysis through enumerating all states is not applicable to large scale systems. It is necessary to develop more effective verification algorithms. Finally, given a system that is not opaque, another problem is how to design controllers restricting the behavior of the system such that the controlled system is opaque and maximally permissive.

This thesis focuses on the observation structures and opacity problems in DESs. The main results of this research are briefed as follows.

1. Petri nets are a graphical and mathematical tool applicable to many systems and a promising tool for describing and studying DESs. We propose two more general classes of Petri nets: labeled Petri nets with outputs (LPNOs) and adaptive labeled Petri nets (ALPNs), to better formalize and generalize current Petri net observation structures. Compared with other Petri net models, the two classes of generators generally have the highest modeling power. Looking for bridges between the different formalisms, we also present general procedures for the transformation between models and algorithms to convert one structure into another one if possible.

2. Current-state opacity, initial-state opacity and language-based opacity are formally defined in the framework of Petri nets. Then we study their differences between the

opacity properties in automata. Moreover, we generalize the notion of language opacity to strict language opacity to deal with the case where the intruder is only interested in a subset of transitions.

3. It has been shown that current-state, initial-state and language opacity verification problems are decidable in finite systems. However, the decidability of current-state, initial-state and language opacity verification problems in Petri net systems still requires further investigation. In the thesis, we prove that current-state, initial-state and language opacity verification problems in Petri net systems are undecidable. Namely, there is no general algorithm that gives a correct answer to the question whether a given Petri net system is current-state (or initial-state, or language) opaque with respect to a given secret or not. Therefore, one can focus on the opacity verification problems in bounded Petri net systems.

4. For bounded systems, the existing methods of verifying opacity require enumeration of the set of states consistent with an observation. Obviously, for large scale systems the state explosion problem is unavoidable. Based on the notion of basis markings, we propose a practically efficient approach to verifying current-state opacity and initial-state opacity. Using the proposed approach, the exhaustive enumeration of the state space can be avoided but only a small number of basis markings. All other markings can be represented by a set of linear systems. By solving the integer linear programming problems, current-state and initial-state opacity can be verified. Moreover, a structure, called *verifier*, is constructed to analyze strict language opacity. Given the nets modeling the plant and the secret, the verifier synchronizes the plant and the secret keeping tracks of both sequences belonging to and not belonging to the secret. In particular, thanks to the notion of minimal explanations there is no need to enumerate all the secret/non-secret sequences. Therefore, the proposed approach is more efficient than other methods.

5. Finally, given a system modeled with a finite automaton that is not current-state opaque with respect to a given secret, we propose an approach to designing an optimal supervisor that restricts the behavior of the system to ensure current-state opacity of the controlled system. In other works, it is usually assumed that the set of events observable by the intruder is included in the set of events observable by the supervisor, or vice versa. Clearly, such an inclusion relation may not hold in general. Therefore, in the thesis, we consider the case where the two sets of events are incomparable. A new

structure, called *augmented I-observer*, is defined to compute the supremal sublanguage generated by the system that will not leak the secret. Based on the augmented I-observer, a set of locally optimal supervisors can be synthesized such that the controlled system is maximally permissive and current-state opaque with respect to the given secret.

Finally, contributions of the thesis are summarized as conclusions, and future research on observation structures of Petri nets and opacity problems in DESs is prospected.

Keywords: Discrete event system, Petri net, opacity, information security, automaton, supervisory control

摘要

物联网、通信网络、计算机网络等系统都具有信息的传递与共享，保证这些信息不被非法获取是系统安全性的重要组成。近年来，人们对信息安全的关注促进了系统安全性形式化方法的研究，研究者们提出了不干涉性、匿名性、隐蔽性等不同概念。其中隐蔽性这一性质被证明更具一般性，例如不干涉性、匿名性等均可规约为隐蔽性。隐蔽性描述了系统信息流的缺失。具体而言，系统具有隐蔽性意味着无论入侵者的观测到什么都无法推知出系统的敏感信息（称为“秘密”）。在离散事件系统中，根据秘密的不同定义，隐蔽性主要可分为当前状态隐蔽性、初始状态隐蔽性、语言隐蔽性等。

受限于传感器功能、信息传递方式等约束，系统的行为一般无法完全被外界所感知，包括试图非法获得系统敏感信息的入侵者。要对离散事件系统进行建模和分析首先要有强有力的离散事件系统建模工具能描述外界对系统的不同观测行为。其次，由于状态爆炸问题，通过穷举所有可能状态的隐蔽性分析方法对大规模系统并不适用。设计更为高效、可实现的系统安全性分析方法尤为重要。最后，对于不具有隐蔽性的系统，如何通过控制系统行为避免秘密的泄露也是隐蔽性研究的一个重要课题。因此，本论文主要研究了离散事件系统模型的观测结构以及基于离散事件系统模型的系统隐蔽性分析与控制，主要研究成果如下：

1. Petri网是系统建模的有效数学和图形工具。为了拓展现有Petri网观测结构，在本论文中我们提出了具有输出的标记Petri网（LPNOs）和动态标记Petri 网（ALPNs）两种Petri网模型，并对其建模能力进行了研究。我们证明了相比其它Petri网模型而言，LPNOs与ALPNs具有最高的建模能力。因此，这两种新模型的提出大大扩展了Petri网的应用范围。为了研究具有不同观测结构的Petri模型间的相互关系，我们进一步提出了观测等价的概念，研究了不同模型间的等价转换，并提出了相应的转换算法。

2. 本论文给出了Petri网模型下当前状态隐蔽性、初始状态隐蔽性、语言隐蔽性的严格定义。然后，详细比较了这三种隐蔽性在Petri网模型中与自动机模型中的区别。对于语言隐蔽性，我们提出了严格语言隐蔽性的概念，这一概念既能形式化原有语言隐蔽性概念又能更灵活地描述当入侵者只关注部分事件时的系统隐蔽性。

3. 研究者们已经证明了有界Petri网系统的隐蔽性验证是可判断的，但是对于无界网系统并没有严格的结论。本论文证明了在Petri网模型下，当前状态隐蔽性、

初始状态隐蔽性、语言隐蔽性的验证是不可判定的，也就是说并不存在一般性算法能对任意Petri网系统的隐蔽性进行验证。因此对于Petri网系统隐蔽性验证的研究可集中于有界网系统。

4. 离散事件系统模型下，已有系统当前状态隐蔽性和初始状态隐蔽性验证的方法都必须列举系统所有可能的状态，因此对于大规模系统不可避免地会遇到状态爆炸问题。基于基本标识图，本论文对有界Petri网系统提出了更为有效的当前状态和初始状态隐蔽性的验证方法。所提方法只需要列举基本标识，即很小一部分可达标识，其他标识均可用一组线性等式表示。最后通过解整数线性规划可以有效地判定系统的当前状态和初始状态隐蔽性。从而有效地避免了状态爆炸问题，对于验证大规模系统的隐蔽性也是可行的。对于严格语言隐蔽性，我们提出了验证器这一新的结构进行判定。验证器采用系统与秘密并行运行的思路，无需对非秘密变迁序列进行验证。同时，验证器的构造是基于最小释义序列的，因此避免了列举所有的变迁发射序列，算法效率得到进一步提高。

5. 如果系统不具有当前状态隐蔽性，一个重要问题是如何设计控制器以保障系统的隐蔽性。针对这一问题研究者们已经提出了一些有效的控制器设计算法，但前提是必须假设控制器与入侵者的可观事件集之间存在包含或者被包含的关系。显然，在大多数实际情况下这样的包含关系并不一定成立。基于有限状态自动机模型，我们提出了一种被称为“扩展入侵者观测器”的新结构，利用该结构可以有效计算出系统所有具有隐蔽性的行为。基于该结构和监督控制理论，在本论文中我们提出了新的控制器设计算法。该算法移除了控制器与入侵者可观事件集之间需具有一定包含关系的约束，在保障系统当前状态隐蔽性的同时保证了受控系统的行为是最大许可的。

最后，在总结本论文工作的基础上，对离散事件系统观测结构与隐蔽性研究的未来工作进行了展望。

关键词：离散事件系统， Petri网， 隐蔽性， 信息安全， 自动机， 监督控制理论

List of Figures

List of Tables

List of Symbols

$\mathbb{N}$	The set of non-negative integers
$\mathbb{Z}$	The set of integers
$\mathbb{R}$	The set of real numbers
$\mathbb{R}_{\geq 0}$	The set of nonnegative real numbers
$\vec{0}$	A vector whose elements are zero
$\mathcal{A}$	An automaton $\mathcal{A} = (X, E, \delta, x_0, X_m)$
$\mathcal{A}^r$	The reverse automaton of $\mathcal{A}$
$\mathcal{A}^e$	The initial-state estimator of $\mathcal{A}$ wrt E', $\mathcal{A}^e = (\overline{X}, E', \delta_e, \overline{x}_0)$
$L(\mathcal{A})$	The generated language of $\mathcal{A}$
$L_m(\mathcal{A})$	The marked language of $\mathcal{A}$
$L(\mathcal{A}, x)$	The language generated from x in $\mathcal{A}$
$P_{E'}$	The natural projection from E to E'
$P_{E'}^{-1}$	The inverse projection of $P_{E'}$
$R_\varepsilon(x)$	The set of states reached from x by empty word
$R_e(x)$	The set of states reached from x by strings starting with e and containing only e and empty word
N	A Petri net $N = (P, T, Pre, Post)$
P	The set of places of a Petri net
T	The set of transitions of a Petri net
T_e	The set of transitions labeled with $e \in \Sigma \cup \{\varepsilon\}$
$T(M)$	The set of transitions enabled at marking M
$T_e(M)$	The set of transitions enabled at marking M and labeled with e
Pre	The input matrix of a Petri net
$Post$	The out matrix of a Petri net
C	The incidence matrix of a Petri net
T_u	The set of unobservable transitions
C_u	The incidence matrix of the T_u-induced subnet of a Petri net
σ	An event sequence $\sigma \in E^*$ or $\sigma \in T^*$
M	A marking of a Petri net
M_0	The initial marking of a Petri net system
$\mathcal{M}_0$	A set of initial markings
$\langle N, M_0 \rangle$	A Petri net system
$R(N, M_0)$	The reachability set of a Petri net $\langle N, M_0 \rangle$
$L(N, M)$	The set of transition sequences enabled at M
G	A labeled Petri net, $G = (N, M_0, \Sigma, \ell)$
$\mathcal{L}(G, M)$	The language generated from M of an LPN system G
G_O	A labeled Petri net with outputs, $G_O = (N, M_0, \Sigma, \ell, f)$
G_P	A partially observed Petri net, $G_P = (N, M_0, \Sigma, \ell, V)$
G_A	An adaptive labeled Petri net, $G_A = (N, M_0, \Sigma_A, \ell_A)$
w	A word $w \in E^*$ or $w \in \Sigma^*$
$\mathcal{U}(M)$	The unobservable reach of marking M
$\mathcal{C}(w)$	The set of markings consistent with w
$\mathcal{I}(w)$	The set of markings generating w

$\mathcal{G}_A$	An agreement graph
$\hat{\mathcal{G}}_A$	A conflict graph
$\mathcal{M}_B$	The set of basis markings
$\mathcal{B}$	A basis reachability graph
$\mathcal{B}_c$	The basis reachability graph for current-state opacity
$\mathcal{M}_b(w)$	The set of basis markings consistent with w
$\mathcal{I}_b(w)$	The set of basis markings generating w
S	A secret, $S \subseteq R(N, M_0)$ or $S \subseteq L(N, R(N, M_0))$
$ex(S)$	The set of exposable markings
$wex(S)$	The set of weakly exposable markings
G^S	The labeled Petri net system describing the secret
$\mathcal{V}$	A verifier for strict language opacity
E_I	The set of events observable by the intruder
E_S	The set of events observable by the supervisor
E_C	The set of events controllable by the supervisor
Sup	A supervisor $Sup = (Y, E_S, \delta_s, y_0, \Psi)$
$\mathcal{A}_g$	An augmented I-observer

List of Abbreviations

DES	Discrete Event System
NFA	Nondeterministic Finite Automaton
DFA	Deterministic Finite Automaton
PN	Petri Net
LPN	Labeled Petri Net
RG	Reachability Graph
BRG	Basis Reachability Graph
POPN	Partially Observed Petri Net
LPNAF	Labeled Petri Net with Affine Output Funtion
LPNO	Labeled Petri Net with Outputs
ALPN	Adaptive Labeled Petri Net
CSO	Current-State Opacity/Opaque
ISO	Initial-State Opacity/Opaque
LO	Language Opacity/Opaque
SLO	Strict Language Opacity/Opaque
GMEC	Generalized Mutual Exclusion Constraint
ILPP	Integer Linear Programming Problem
LPP	Linear Programming Problem
SCOP	Supervisory Control and Observation Problem
CSOEP	Current State Opacity Enforcement Problem

Content

Chapter 1 Introduction

Cyberinfrastructures, ranging from Internet and mobile communication networks to national defense and health service systems, are integrated into every aspect of our lives. Motivated by the concern about cybersecurity, various notions of secrecy have been formulated, such as noninterference [1, 2], anonymity [3] and opacity [4]. The notion of opacity introduced in [5] for discrete event systems (DESs) formalizes the absence of information flow, or more precisely, the impossibility for an intruder to infer the truth of the secret information. Besides describing secrecy of a system, opacity is proven capable of formulating observability, diagnosability and detectability of a system [6]. Thus, opacity has a wide range of application. Over the last decade, opacity problems in DESs have become a very fertile field of research [7].

A discrete event system (or event-driven system), different from time-driven systems, is a dynamic system with a discrete state space and piecewise constant state trajectories that evolve in accordance with the abrupt occurrence, at possibly unknown irregular intervals, of physical events that determine a state transition. Many systems, particularly technological ones (e.g., queueing systems, computer systems), are in fact discrete state systems. Note that even if this is not the case, for many applications of interest a discrete-state view of a complex system may be necessary.

Different models, like finite automata and Petri nets, are used for specification, verification, synthesis as well as for analysis and evaluation of different qualitative and quantitative properties of existing physical systems. In particular, Petri nets (PNs) are a graphical and mathematical tool applicable to many systems and a promising tool for describing and studying systems (e.g., information processing systems and automated manufacturing systems) that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. The issue of representing behavior of DESs using appropriate modeling formalisms is a key issue for performing analysis and control of DESs.

In this thesis, we study, first the modeling of DESs using Petri nets. In particular, we consider DESs where some events or states are unobservable. Then we study opacity problems in DESs. More precisely, opacity verification and enforcement problems are considered. Given a DES that is modeled by a Petri net, the opacity verification problem consists in determining whether the system is opaque with respect to a given secret or not. On the other hand, the opacity enforcement problem consists in turning a system that is not

opaque into opaque. In this chapter, we first present an overview on these two topics and then we give a motivation for our studies. Finally, we describe the structure of the thesis and its contributions.

1.1 Observation Structures for Petri Net Generators

The behavior of a *logical*, i.e., *untimed*, DESs can be described in terms of sequences that specify the order of event occurrences. Such behavior can be represented by means of a *formal language* whose alphabet is the event set of a DES and the event sequences are words in that language. The issue of representing languages using appropriate modeling formalisms is a key issue for performing analysis and control of DESs [8].

It is often assumed that the initial state of a system is known but the system dynamics is not perfectly known due to partial observations provided by sensors. The set of events is partitioned into two disjoint sets: observable events whose occurrences can be detected by sensors and unobservable ones whose occurrences cannot be detected. In this thesis, Petri net generators are considered, where the state is given by token distribution on places, and events are represented by transitions. A classical Petri net generator to model systems with the aforementioned observation structure is the so called *labeled Petri net* (LPN). LPNs have been adopted by many researchers to analyze and control a DES [5, 9–12]. In [13–17] a more general model where state information may also be provided by sensors is considered: in particular they assume that some places of a Petri net may be observable, i.e., the number of tokens that they contain can be measured. In this case, there are two types of observations: labels of transitions and components of markings. Such a class of Petri net generators is usually called *partially observed Petri nets* (POPNs) [16, 17]. This class of generators is extended in [18] considering observations that are linear functions of the marking and thus can model sensors that are not able to provide precise measurements of the state components but only information such as the total amount of available resources regardless of their distribution. However, this type of observation cannot describe affine or general nonlinear functions of the marking.

To cover all cases of practical interest where complex observations are possible, it is meaningful to better formalize and generalize current Petri net observation structures. Moreover, Ru and Hadjicostis [16] show that for any POPN there exists an observation equivalent LPN. We believe that it is needed to study the conversion procedure among different classes of Petri net generators for following reasons: finding a conversion procedure between two different formalisms has a theoretical interest *per se* and in the literature several approach-

es of this type have been proposed for models of concurrent systems (e.g., communicating sequential processes, place/transition nets, process algebra) or performance models (e.g., stochastic Petri nets, queueing networks).

1.2 Opacity Problems in DESs

Opacity is a general information flow property that characterizes whether a given "secret" behavior of a system cannot be inferred by an outsider, further called the intruder. It is assumed that the intruder knows the structure of the system but only has partial observability. Therefore, based on its observation, the intruder can estimate the behavior of the system. The system is opaque with respect to a given secret if the estimate of the intruder never falls in the secret. In other words, no matter what observing, the intruder always has uncertainty about whether a secret behavior has occurred. In DESs, the secret can be a set of states or a language. Depending on the definition of secret, opacity properties can be categorized into two main classes: *state-based opacity* and *language-based opacity*.

Except the earlier work of Bryans, *et al*. [5] that is carried out in the framework of Petri nets, most work related to opacity in DESs is studies in the framework of finite automata. In automata, different notions of opacity have been defined, such as current-state opacity, initial-state opacity, k-step opacity, language opacity, strong language opacity, weak language opacity, etc. [6, 19–22]. In the framework of automata two types of observation masks have been investigated in the literature: static and dynamic [6, 23]. A mask is static if the set of events that the intruder can observe is fixed. It is dynamic if the set of observable events changes with the state or the trace of the system. Obviously, the dynamic mask is a generalization of the static one. In [22], it is proven that in automata current-state opacity, initial-state opacity, initial-and-final-state opacity, and language opacity are transformable in polynomial time. In the thesis, we mainly study three opacity properties: *current-state opacity* (CSO), *initial-state opacity* (ISO), and *language opacity* (LO).

- The secret is defined as a set of states. A system is *current-state opaque* if the intruder is never able to establish if the current state of the system is within the set of secret states.

- The secret is also defined as a set of states. A system is *initial-state opaque* if the intruder cannot establish if the evolution of the system has started from a secret state.

- In the case of language opacity, the secret is defined as a language, i.e., a set of event

sequences. A system is *language opaque* if the intruder cannot establish if the evolution of the system belongs to the secret.

Few similar opacity notions are defined in Petri nets [5], which extend the modeling power of finite automata and provide structural and mathematical advantages of modeling and analyzing the system. It is worthwhile to formalize the notions of opacity that have been widely studied in automata in Petri nets, and to further investigate the opacity problems.

Opacity problems consist in two aspects: *opacity verification* and *opacity enforcement*. In the remainder of this section, we provide a comprehensive review on work considering DESs and relating to the opacity problems.

1.2.1 Opacity Verification

The opacity verification problem is of determining whether a system is opaque with respect to a given secret or not. It has been shown that current-state, initial-state and language opacity verification problems are decidable in finite automata [24]. Nonetheless, the current-state opacity verification problem in probabilistic finite automata and language opacity in timed automata are undecidable [25, 26]. Bryans *et al.* [5] prove that for *bounded* Petri nets current-state, initial-state opacity and language opacity verification problems are decidable. Moreover, some opacity verification problems in transition systems are undecidable, as well as the B-initial-state opacity problem in Petri nets [24]. Decidability of opacity verification problems in different systems is surveyed in [27]. However, the decidability of current-state, initial-state and language opacity verification problems in Petri nets still require further investigation.

Methods for verifying opacity are proposed by many researchers in the area of DESs [5, 19, 22, 25, 28–30]. In a system modeled by a nondeterministic finite automaton (NFA), the verification of both current-state opacity and initial-state opacity is PSPACE-complete [7, 28, 31], with respect to the number n of states in the NFA. To verify current-state opacity one needs to convert the NFA into an equivalent deterministic finite automaton (DFA), which has a complexity of $\mathcal{O}(2^n)$ [19, 22]. Initial-state opacity in NFA can be verified by the method proposed by Saboori and Hadjicostis [28]. In their approach a DFA called the *trellis-based initial-state estimator* is constructed with a complexity of $\mathcal{O}(2^{n^2})$. A state of the estimator reached from the initial state following a word w includes all pairs (initial state, current state) of the NFA such that the current state may be reached from the corresponding initial state observing w. As long as an initial-state estimator is built, there is no need to reconstruct it when the secret changes. In their improved method [28], *verifiers* are introduced to study

initial-state opacity. The verifier does not precisely estimate the initial state but only records the possible current states and if such states are reachable from secret/non-secret states, and hence the complexity is reduced to $\mathcal{O}(4^n)$. Furthermore, Wu and Lafortune [22] show that the observer of the corresponding reverse automaton can be used to estimate the initial state, which further reduces the complexity of verifying initial-state opacity to $\mathcal{O}(2^n)$. In [6], the notions of strong language opacity and weak language opacity are defined in automata. It is shown that strong language opacity can be verified in complexity of $\mathcal{O}(4^n)$. Polynomial algorithms to check weak language opacity of an automaton are proposed in [32]. On the contrary, providing an efficient algorithm to check other language-based opacity is still an open problem.

Petri nets have been proposed as a fundamental model for DESs in a wide variety of applications. Many problems such as supervisory control, fault diagnosis, etc. have been solved using Petri net models, providing efficient and well founded approaches [33–35]. Nevertheless, few works use this model to deal with opacity. Apart from the earlier work of Bryans *et al.* [5, 24], so far no efficient opacity analysis method has been reported yet. For bounded Petri nets we may construct its reachability graph (RG) that is an NFA, so that the aforementioned approaches could be applied. However, constructing the RG will inevitably suffer from the state explosion that also characterizes automaton models. The verification of language-based opacity in Petri nets has never been discussed before, the only way to solve the problem in bounded Petri nets was to construct the RG and apply the approach in [6]. Moreover, there exist many practical cases where the intruder only cares about a subset of transitions, which cannot be described by any existing language-based opacity directly. As an example, in a defence system, the intruder may only care about the order and the occurrence of some events rather than all the events. Therefore, it is meaningful to develop a new notion of language-based opacity.

1.2.2 Opacity Enforcement

Given a system that is not opaque, the opacity enforcement problem consists in turning the system into an opaque one. Approaches to opacity enforcement may rely on supervisory control [36–40], dynamically restraining the observability of events [23], inserting additional events in the output behavior of the system [41, 42] and the runtime validation technique [30].

The authors of [23, 31] investigate the problem of synthesizing a dynamic observation mask under which the system is current-state opaque (or language opaque). It is assumed

that the intruder knows not only the structure of the system but also the synthesized dynamic observation mask. By reducing the problem to a safety two-player game problem, it shows that the dynamic observation masks enforcing opacity can be computed in EXPTIME and they can be finitely presented.

In [41], Wu and Lafortune propose the method of using insertion function. The insertion function is a monitoring interface placed at the output of the system. It monitors the system's output behavior and inserts fictitious observable events to the output without interacting with the system. To avoid causing suspicion, the modified output has to be consistent with an existing behavior that does not reveal the secret. A finite structure called *all insertion structure* is proposed to synthesize insertion functions enforcing current-state opacity. In [42], assuming the intruder knows the implementation of the insertion function, the authors further study the problem and propose an algorithm to compute the insertion function that can be publicly known.

The aforementioned methods share the same idea of restricting the observability of the intruder so that it is confused and the system is opaque. Therefore, the behavior of the system would never be restricted. However, imposing dynamic observation functions or the insertion functions may require extra costs as mentioned in [23, 43], or even impossible. This leads us to supervisory control. Given a system that is not opaque with respect to a given secret, the problem of opacity enforcement using supervisory control is to design an optimal supervisor that restricts the behavior of the system such that the controlled system is opaque and maximally permissive.

There is some related work on the design of supervisors to enforce opacity properties. In [29], the authors consider enforcing language opacity and a set of intruders having different observations. They assume that all events are observable and controllable to the supervisor, and showed that an optimal supervisor always exists. Considering the same language opacity enforcement problem but with only one intruder, Dubreil *et al.* [37, 44] study a more general case where the supervisor may observe a set of events different from the one observed by the intruder in the presence of uncontrollable events. The authors of [39] propose methods for designing optimal supervisors to enforce two different opacity properties: initial-state opacity and infinite-step opacity, with the assumption that the supervisor can observe all events. More recently, the common assumption that all controllable events are also observable [29, 37, 39, 44] is relaxed in [40] to enforce current-state opacity. However, all these work [29, 37, 39, 40] assumes that the set of events observable by the intruder is included in the set of events observable by the supervisor, or vise versa. Clearly, in practice,

it is not always the case.

1.3 Organization and Contribution of the Thesis

The work of the thesis can be partitioned into two parts: Chapter 3, and Chapters 4 to 7, which relate to modeling of DESs and opacity problems in DESs, respectively, and the two parts are not closely connected. The organization and main contributions of the thesis are summarized as follows:

- Chapter 2. This chapter presents some basics on automata, Petri nets, and theory of computation.

- Chapter 3. We consider in this chapter more general observation structures, by correspondingly defining two new classes of Petri net generators: labeled Petri nets with outputs (LPNOs) and adaptive labeled Petri nets (ALPNs). To compare the modeling power of different Petri net generators, the notion of observation equivalence is proposed. Compared with other classes of Petri nets generators, ALPNs are shown to be the class of bounded generators possessing the highest modeling power. Looking for bridges between the different formalisms, we first present a general procedure to convert a bounded LPNO into an equivalent ALPN or even into an equivalent labeled Petri net (if any exists). Finally, we discuss the possibility of converting an unbounded LPNO into an equivalent ALPN.

- Chapter 4. Definitions of current-state opacity, initial-state opacity, and language opacity in automata have been recalled in this chapter. Furthermore, we formalize the aforementioned opacity properties in labeled Petri nets (LPNs) and compare their differences with ones in [5]. Finally, we define a new notion of language-based opacity in Petri nets, called *strict language opacity*, which describes the scenario of the intruder cares only a subset of transitions. Its relation with language opacity is also studied.

- Chapter 5. The main contribution of this chapter consists in proving that current-state, initial-state and language opacity verification problems are undecidable. All proofs are carried out using reduction.

- Chapter 6. Novel approaches based on the analysis of basis reachability graph (BRG), which is typically greatly smaller than the RG, are proposed to verify current-state

opacity, initial-state opacity in LPNs. Moreover, if the secret is defined as the intersection of a series of generalized mutual exclusion constraints (GMECs), then current-state opacity can be verified by solving a set of integer linear programming problems (ILPPs) instead of exhaustively enumerating the unobservable reach of basis markings. If the incidence matrix is totally unimodular, then these ILPPs can be relaxed to linear programming problems (LPPs). We show that if a certain assumption is satisfied, initial-state opacity can be efficiently verified using the BRG. Otherwise, we propose a modified BRG (MBRG) to verify initial-state opacity. A finite structure called *verifier* is proposed to verify strict language opacity in LPNs under the assumption that the intruder is interested in only observable transitions and the secret is described by the generated language of a Petri net. Compared with other language opacity verification approach, the proposed approach is of lower complexity. Finally, we develop a MATLAB toolbox to implement most of the proposed algorithms.

- Chapter 7. We tackle the current-state opacity enforcement problem in the framework of finite automata. No containment relation is assumed between the sets of events observable by the intruder and by the supervisor. We call this general setting *incomparable observations*. Furthermore, we relax the assumption that all controllable events are observable. A structure called *augmented I-observer* is constructed, and it is shown that based on the augmented I-observer, the current-state opacity enforcement problem with incomparable observations can be reduced to the problem of supervisory control under partial observation. Thus, a set of locally optimal supervisors enforcing current-state opacity can be synthesized.

- Chapter 8. We conclude the thesis and discuss the potential directions for future work.

Chapter 2 Background

In this chapter we recall the formalism used in the thesis, namely automata and Petri nets.

2.1 Automata

2.1.1 Automaton Models

Definition 2.1. A *nondeterministic finite automaton* (NFA) is a 5-tuple $\mathcal{A} = (X, E, \delta, x_0, X_m)$, where X is a finite set of *states*, E is a set of *events*, $\delta : X \times E \cup \{\varepsilon\} \to 2^X$ is a (partial) *transition relation*, $x_0 \in X$ is the *initial state*, and $X_m \subseteq X$ is a set of *marked states*. ◇

An automaton can be described by a graph in which each state corresponds to a node, represented by a circle. The initial state is denoted by an arrow and a final state by a double circle. We call ε the *empty word*. To be more general, there could be a set of initial states $X_0 \subseteq X$ in $\mathcal{A}$. Set E^* is the set of finite strings composed of elements of E, including the empty word. The transition relation δ can be extended to[1] $\delta : X \times E^* \to 2^X$. We denote by $\delta(x, \sigma)!$ the fact that $\sigma \in E^*$ is defined at x in $\mathcal{A}$. If the transition relation in Definition 2.1 is a one to one function $\delta : X \times E \to X$, then the automaton is called a *deterministic finite automaton* (DFA). The transition function of a DFA can be extended to $\delta : X \times E^* \to 2^X$ simply as: for all $x \in X$, $\delta(x, \varepsilon) = x$ and $\delta(x, \sigma e) = \delta(\delta(x, \sigma), e)$ for $\sigma \in E^*$ and $e \in E$. When there are no marked states, we write $\mathcal{A} = (X, E, \delta, x_0)$; when there is no initial state is specified either, we write $\mathcal{A} = (X, E, \delta)$. Throughout the thesis, we assume that all states in an automaton are reachable from the initial state.

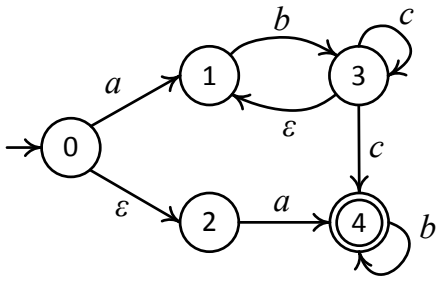

Fig. 2.1 Nondeterministic finite automaton.

Table 2.1 Transition relation of the NFA in Fig. 2.1.

δ	ε	a	b	c
0	$\{0, 2\}$	$\{1\}$	-	-
1	$\{1\}$	-	$\{3\}$	-
2	$\{2\}$	$\{4\}$	-	-
3	$\{1, 3\}$	-	-	$\{3, 4\}$
4	$\{4\}$	-	$\{4\}$	-

[1] For sake of simplicity, this procedure is not recalled here and we refer to [45] for details.

Example 2.1. Fig. 2.1 shows the graphical structure of a NFA $\mathcal{A} = (X, E, \delta, x_0, X_m)$ with $X = \{0, 1, 2, 3, 4\}$, $E = \{a, b, c\}$, $x_0 = 0$, $X_m = \{4\}$. Its transition relation is given by Table 2.1. Fig. 2.2 shows a DFA $\mathcal{A} = (X, E, \delta, x_0)$ with $X = \{0, 1\}$, $E = \{a, b\}$, and $x_0 = 0$. Its transition function is given by Table 2.2. ◇

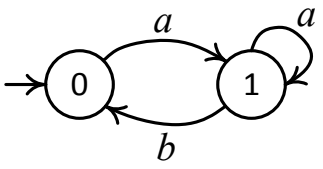

Fig. 2.2 Deterministic finite automaton.

Table 2.2 Transition function of the DFA in Fig. 2.2.

δ	a	b
0	1	-
1	1	0

Definition 2.2. Given an NFA $\mathcal{A} = (X, E, \delta, x_0, X_m)$, its *generated language*[2] is defined as

$$L(\mathcal{A}) = \{\sigma \in E^* | \delta(x_0, \sigma)!\}.$$

Its *marked language* is defined as

$$L_m(\mathcal{A}) = \{\sigma \in E^* | \delta(x_0, \sigma) \cap X_m \neq \emptyset\}.$$ ◇

Note that if $\mathcal{A}$ is a DFA, then $L_m(\mathcal{A}) = \{\sigma \in E^* | \delta(x_0, \sigma) \in X_m\}$. The language generated from a given state $x \in X$ of an automaton $\mathcal{A} = (X, E, \delta, x_0, X_m)$ is defined as $L(\mathcal{A}, x) = \{\sigma \in E^* | \delta(x, \sigma)!\}$. Generally, given a set of states $Y \subseteq X$, we define $L(\mathcal{A}, Y) = \bigcup_{x \in Y} L(\mathcal{A}, x)$ the language generated from states in Y.

The *prefix closure* of a language $L \subseteq E^*$ is defined to be the language

$$\overline{L} = \{\sigma' \in E^* | \sigma'\sigma'' \in L \text{ for some } \sigma'' \in E^*\}.$$

If $L = \overline{L}$, we say L is *prefix-closed*. Clearly, the generated language of $\mathcal{A}$ is always prefix-closed.

Example 2.2. Consider the NFA in Fig. 2.1. Its generated language is $L(\mathcal{A}) = \{\varepsilon, a, ab, abc, abb, \ldots\}$, which is prefix-closed. The marked language of $\mathcal{A}$ is $L_m(\mathcal{A}) = L(\mathcal{A}) \setminus \{\varepsilon\}$. Consider the DFA in Fig. 2.2. Its generated language is $L(\mathcal{A}) = \{\varepsilon, a, aa, ab, aab, \ldots\}$.

Let $E = \{a, b, c\}$ and $L = \{abb, abc\}$. The prefix closure of L is $\overline{L} = \{\varepsilon, a, ab, abb, abc\}$. Clearly, $L \neq \overline{L}$, and thus L is not prefix-closed. ◇

Let $E' \subseteq E$ be a subset of E. The *natural projection* of E on E' is $P_{E'} : E \to E'$, defined as:

$$P_{E'}(e) = \begin{cases} \varepsilon, & \text{if } e \in \{\varepsilon\} \cup (E \setminus E'); \\ e, & \text{if } e \in E'; \end{cases}$$

[2]Thus $\mathcal{A}$ is also called a *generator*.

and it can be extended to $P_{E'} : E^* \to E'^*$ in a recursive manner: $P_{E'}(\sigma e) = P_{E'}(\sigma)P_{E'}(e)$, where $\sigma \in E^*$ and $e \in E$. Given a language $L \subseteq E^*$, $P_{E'}(L) = \bigcup_{\sigma \in L} P_{E'}(\sigma)$. The *inverse projection* of $P_{E'}$ is denoted by $P_{E'}^{-1} : E'^* \to E^*$ and is defined as $P_{E'}^{-1}(w) = \{\sigma \in E^* | P_{E'}(\sigma) = w\}$. Thus, the projection operator $P_{E'}$ removes all the events that are not in E' from a string, while its inverse projection associates to a sequence w the set of strings whose projection on E' is w.

Projection $P_{E'}$ projects all events in $E \setminus E'$ to the empty word, which models that the occurrence of events in E' cannot be detected. Thus events in E' (resp., $E \setminus E'$) are called *observable* (resp., *unobservable*) events, $P_{E'}$ is also called an *observation mask*, and $w = P_{E'}(\sigma)$ is called an *observation* of $\sigma \in E^*$.

Example 2.3. Let $E = \{a, b, c\}$, $E' = \{a, b\}$ and $\sigma = acbcbbc$. The projection of σ on E' is $P_{E'}(\sigma) = abbb$. Let $w = ab \in E'^*$. Then $P_{E'}^{-1}(w) = \{c^i a c^j b c^k\}$ with $i, j, k \in \mathbb{N} = \{0, 1, 2, \ldots\}$. ◇

2.1.2 Operations on Automata

In this section, we introduce three operations on automaton used in the thesis: *reverse automata*, *observer automata*, and *parallel composition*. By operating on the automaton/automata one can do some language operations or compose two or more automata, etc.

Reverse Automata

Given an automaton $\mathcal{A} = (X, E, \delta)$, its *reverse automaton* $\mathcal{A}^r = (X, E, \delta^r)$ is the automaton obtained by reversing all arcs in $\mathcal{A}$. Namely, given a state x and an event $e \in E$, if $x' \in \delta(x, e)$ holds in $\mathcal{A}$ then $x \in \delta^r(x', e)$ holds in $\mathcal{A}^r$. Furthermore, $x' \in \delta(x, \sigma)$ holds in $\mathcal{A}$, if and only if $x \in \delta^r(x', \sigma')$ holds in $\mathcal{A}^r$, where $x, x' \in X$, $\sigma \in E^*$ and σ' is its reverse.

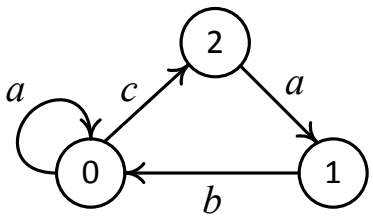

Fig. 2.3 DFA without specifying the initial state.

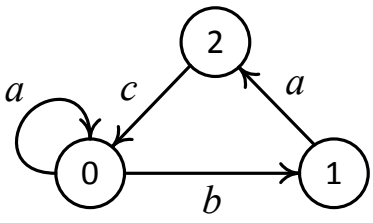

Fig. 2.4 Reverse automaton of the DFA in Fig. 2.3.

Example 2.4. Consider the DFA $\mathcal{A} = (X, E, \delta)$ in Fig. 2.3. Its reverse automaton $\mathcal{A}^r = (X, E, \delta^r)$ is shown in Fig. 2.4. We can see that in $\mathcal{A}^r$, $\delta^r(1, acba) = 2$, and $\delta(2, abca) = 1$ in $\mathcal{A}$. ◇

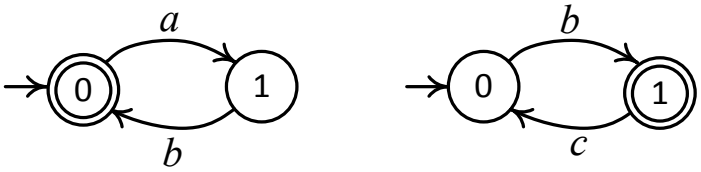

Fig. 2.5 Two DFAs $\mathcal{A}_1$ (left) and $\mathcal{A}_2$ (right) in Example 2.5.

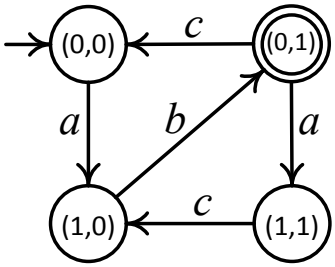

Fig. 2.6 Parallel Composition between $\mathcal{A}_1$ and $\mathcal{A}_2$ in Fig. 2.5.

Parallel Composition

Parallel composition synthesizes several subsystems to an overall system. In this way, the interaction between the various subsystems is modeled. For sake of simplicity, consider two DFAs $\mathcal{A}_1 = (X_1, E_1, \delta_1, x_{01}, X_{m1})$ and $\mathcal{A}_2 = (X_2, E_2, \delta_2, x_{02}, X_{m2})$. The parallel composition of $\mathcal{A}_1$ and $\mathcal{A}_2$ is a DFA $\mathcal{A}_1||\mathcal{A}_2 = (X_1 \times X_2, E_1 \cup E_2, \delta_{12}, (x_{01}, x_{02}), X_{m1} \times X_{m2})$, where for $(x_1, x_2) \in X_1 \times X_2$ and $e \in E_1 \cup E_2$

$$\delta_{12}((x_1, x_2), e) = \begin{cases} (\delta_1(x_1, e), x_2), & \text{if } e \in E_1 \setminus E_2 \text{ and } \delta_1(x_1, e)! \\ (\delta_1(x_1, e), \delta_2(x_2, e)), & \text{if } e \in E_1 \cap E_2, \delta_1(x_1, e)! \text{ and } \delta_2(x_2, e)! \\ (x_1, \delta_2(x_2, e)), & \text{if } e \in E_2 \setminus E_1 \text{ and } \delta_2(x_2, e)! \end{cases}$$

Example 2.5. Consider the two DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ in Fig. 2.5. Their parallel composition $\mathcal{A}_1||\mathcal{A}_2$ is shown in Fig. 2.6. ◇

Parallel composition $\mathcal{A}_1||\mathcal{A}_2$ has the following properties:

1. For $i = 1, 2$, $P_{E_i}(L(\mathcal{A}_1||\mathcal{A}_2)) = L(\mathcal{A}_i)$, and

2. $P_{E_i}(L_m(\mathcal{A}_1||\mathcal{A}_2)) = L_m(\mathcal{A}_i)$,

where P_{E_i} is the natural projection from $E_1 \cup E_2$ to E_i. We can see that the maximal number of states of $\mathcal{A}_1||\mathcal{A}_2$ is $|X_1| \times |X_2|$.

Observer Automata

Observer automata is an important tool in the study of opacity problems and we will use them frequently in the rest of the thesis. An *observer automaton* is a deterministic version

of a given NFA such that the generated and marked languages are correspondingly equivalent. Thus, the observer automaton is also called the equivalent DFA, and the procedure of computing the observer automaton is called *determinization*.

Definition 2.3. Given an NFA $\mathcal{A} = (X, E, \delta, x_0, X_m)$, a state $x \in X$ and an event $e \in E$, the *unobservable reach* (or ε-reach) of x is

$$R_\varepsilon(x) = \{x' \in X | x' \in \delta(x, \varepsilon)\},$$

the set of states that can be reached from x by empty word. The e-reach of x is

$$R_e(x) = \bigcup_{x' \in \delta(x,e)} R_\varepsilon(x')$$

the set of states that can reach from x by strings starting with e and containing only one e and empty word. ◇

Example 2.6. Consider the NFA in Fig. 2.1. $R_\varepsilon(3) = \{1, 3\}$ and $R_\varepsilon(4) = \{4\}$. $R_c(3) = \bigcup_{x \in \delta(3,c)} R_\varepsilon(x) = R_\varepsilon(3) \cup R_\varepsilon(4) = \{1, 3, 4\}$. ◇

Given a NFA $\mathcal{A} = (X, E, \delta, x_0, X_m)$ and a subset $E' \subseteq E$ of events, its observer automaton $\mathcal{A}' = (X', E', \delta', x'_0, X'_m)$ with respect to E' can be built following Algorithm 1.

We can see that each state of the observer automaton is a set of states in $\mathcal{A}$, i.e., a subset of X. Therefore, in the worst case the number of states of $\mathcal{A}'$ is $2^{|X|}$. The important properties of $\mathcal{A}'$ are that:

1. $\mathcal{A}'$ is a DFA.
2. $P_{E'}(L(\mathcal{A})) = L(\mathcal{A}')$.
3. $P_{E'}(L_m(\mathcal{A})) = L_m(\mathcal{A}')$.
4. $x \in \delta(x_0, \sigma) \Leftrightarrow x \in \delta'(x'_0, \sigma')$, where $\sigma \in E^*$ and $\sigma' = P_{E'}(\sigma)$.

Example 2.7. Consider the NFA in Fig. 2.1. Let $E' = \{a, b\}$. First, replace event c with ε. $x'_0 = R_\varepsilon(0) = \{0, 2\} \cap X_m = \emptyset$, and $X' = \{x'_0\}$. Therefore, $x'_0 \notin X'_m$. At Step 8 of Algorithm 1:

- Select $B = x'_0$ and $e = a$. $B' = R_a(0) \cup R_a(2) = \{1, 4\} \cap X_m \neq \emptyset$. Therefore, $\delta'(x'_0, a) = \{1, 4\} = x'_1$, $X'_m = \{x'_1\}$ and $X' = \{x'_0(\text{old}), x'_1\}$.

Algorithm 1 Construction of the observer automaton

Input: A NFA $\mathcal{A} = (X, E, \delta, x_0, X_m)$ and a set $E' \subseteq E$
Output: The observer automaton $\mathcal{A}' = (X', E', \delta', x'_0, X'_m)$ wrt E'

1: Replace all events not in E' with ε;
2: $x'_0 := R_\varepsilon(x_0)$;
3: $X' := \{x'_0\}$;
4: **if** $x'_0 \cap X_m \neq \emptyset$, **then**
5: $\quad X'_m := \{x'_0\}$;
6: **else**
7: $\quad X'_m := \emptyset$;
8: **end if**
9: **while** $\exists B \in X'$ with no tag, **do**
10: $\quad$ **for all** $e \in E' : \exists x \in B, \delta(x, e)!$, **do**
11: $\quad\quad B' := \bigcup_{x \in B} R_e(x)$;
12: $\quad\quad$ **if** $B' \notin X'$, **then**
13: $\quad\quad\quad X' := X' \cup \{B'\}$;
14: $\quad\quad\quad$ **if** $B' \cap X_m \neq \emptyset$, **then**
15: $\quad\quad\quad\quad X'_m := X'_m \cup \{B'\}$;
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end if**
18: $\quad\quad \delta'(B, e) = B'$;
19: $\quad$ **end for**
20: $\quad$ Tag B "old";
21: **end while**
22: Remove all tags.

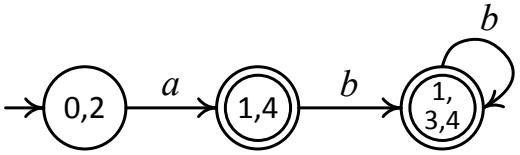

Fig. 2.7 Observer automaton of the NFA wrt $E' = \{a, b\}$ in Fig. 2.1.

- Select $B = x'_1$ and $e = b$. $B' = R_b(1) \cup R_b(4) = \{1, 3, 4\} \cap X_m \neq \emptyset$. Therefore, $\delta'(x'_1, b) = \{1, 3, 4\} = x'_2$, $X'_m = \{x'_1, x'_2\}$ and $X' = \{x'_0(\text{old}), x'_1(\text{old}), x'_2\}$.
- Select $B = x'_2$. For $e = b$, $B' = R_b(1) \cup R_b(3) \cup R_b(4) = x'_2$. Therefore, $\delta'(x'_2, b) = x'_2$. We can go to Step 21 as all states in X' are tagged as "old".

The obtained observer automaton $\mathcal{A}'$ wrt E' is shown in Fig. 2.7. Clearly, $\mathcal{A}'$ is a DFA, $P_{E'}(L(\mathcal{A})) = L(\mathcal{A}')$ and $P_{E'}(L_m(\mathcal{A})) = L_m(\mathcal{A}')$. Let $\sigma = abc$. Then $\sigma' = P_{E'}(\sigma) = ab$. In $\mathcal{A}'$, $\delta'(x'_0, ab) = \{1, 3, 4\}$. Namely, the state reached by abc in $\mathcal{A}$ could be 1, 3, or 4. $\diamond$

In [22] a special observer automaton is proposed: *initial-state estimator*. Given a NFA $\mathcal{A} = (X, E, \delta)$, its initial-state estimator $\mathcal{A}^e = (\overline{X}, E', \delta_e, \overline{x}_0)$ wrt $E' \subseteq E$ is the observer

automaton wrt E' of its reverse automaton $\mathcal{A}^r$, where $\overline{X} \subseteq X$ and $\overline{x}_0 = X$.

Let $\mathcal{A} = (X, E, \delta)$ be an NFA, $\mathcal{A}^e = (\overline{X}, E', \delta_e, \overline{x}_0)$ be its initial-state estimator wrt E', and $w \in E'^*$. There exists $x \in X$ and $\sigma \in E^*$ such that $\delta(x, \sigma)!$ and $P_{E'}(\sigma) = w$ iff $\delta(\overline{x}_0, w') = \overline{x}$ is defined in $\mathcal{A}^e$ such that $x \in \overline{x}$, where w' is the reverse of w.

In words, the state reached by a word w' in $\mathcal{A}^e$ is the set of states from which the word w can be generated in $\mathcal{A}$, where w' is the reverse of w. This is the reason why $\mathcal{A}^e$ is called the initial-state estimator of $\mathcal{A}$.

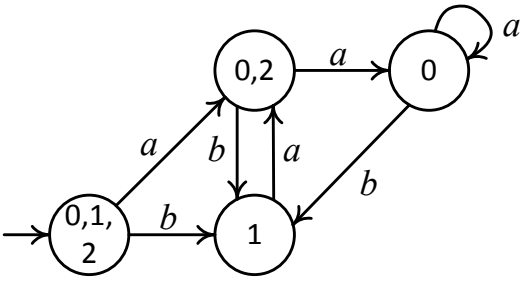

Fig. 2.8 Initial-state estimator $\mathcal{A}^e$ wrt $E' = \{a, b\}$.

Example 2.8. Consider the DFA in Fig. 2.3 and let $E' = \{a, b\}$. Its initial-state estimator wrt E', i.e., the observer of $\mathcal{A}^r$ wrt E', is shown in Fig. 2.8. Suppose $w' = aab$. Then $\delta_e(\overline{x}_0, w') = \{1\}$. Therefore, the word baa is generated from state 1 in $\mathcal{A}$. ◇

2.2 Petri Nets

2.2.1 Petri Net Models

Definition 2.4. A *Petri net* (PN) is a 4-tuple $N = (P, T, Pre, Post)$, where P is a set of *places*, graphically represented by circles; T is a set of *transitions*, graphically represented by bars; $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the *pre-* and *post-incidence functions* that specify the weighted arcs directed from places to transitions and from transitions to places, respectively. We denote them as $m \times n$-dimensional matrices, where $m = |P|$ and $n = |T|$, and for instance $Pre(p, t) = k$ denotes the arc with weight k from place p to t. The incidence matrix of a net is denoted by $C = Post - Pre$. ◇

The *input* and *output* sets of a node $x \in P \cup T$ are denoted by ${}^\bullet x$ and $x^\bullet$, respectively. A Petri net $N = (P, T, Pre, Post)$ is a *state machine* (resp., *marked graph*) if $\forall t \in T$, $|{}^\bullet t| = |t^\bullet| \leq 1$ (resp., $\forall p \in P$, $|{}^\bullet p| = |p^\bullet| \leq 1$). A Petri net is said to be *acyclic* if there are no oriented cycles.

A *marking* is a function $M : P \to \mathbb{N}$ that assigns to each place a nonnegative integer number of tokens, graphically represented by black dots. We denote a marking as a vector $M \in \mathbb{N}^m$, and the number of tokens in place p is denoted by $M(p)$. A marking is also presented as $M = \sum_{p \in P} M(p) \cdot p$. A *Petri net system* $\langle N, M_0 \rangle$ is a net N with *initial marking* M_0.

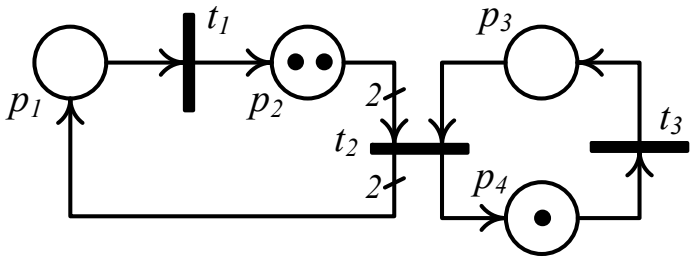

Fig. 2.9 Petri net system.

Example 2.9. Fig. 2.9 shows a PN system $\langle N, M_0 \rangle$ with $M_0 = 2p_2 + p_4$ (or denoted as $M_0 = [0\ 2\ 0\ 1]^T$), $P = \{p_1, p_2, p_3, p_4\}$, $T = \{t_1, t_2, t_3\}$,

$$Pre = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ and } Post = \begin{bmatrix} 0 & 2 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Its incidence matrix is

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

Consider p_2 and t_2. The set of input transitions of p_2 is ${}^\bullet p_2 = \{t_1\}$, and the set of output transitions of p_2 is $p_2^\bullet = \{t_2\}$. The input places of t_2 are ${}^\bullet t_2 = \{p_2, p_3\}$, and the output places of t_2 are $t_2^\bullet = \{p_1, p_4\}$. The Petri net system is not acyclic because there are oriented cycles, e.g., $p_1 t_1 p_2 t_2 p_1$. ◇

A transition t is *enabled* at marking M if $M \geq Pre(\cdot, t)$ and may fire yielding a new marking[3] $M' = M + C(\cdot, t)$. We write $M[\sigma\rangle$ to denote that the sequence of transitions $\sigma = t_{j1} \cdots t_{jk}$ is enabled at M, and $M[\sigma\rangle M'$ to denote that the firing of σ yields M'. A marking M is *reachable* in $\langle N, M_0 \rangle$ if there exists a sequence σ such that $M_0[\sigma\rangle M$. Given a sequence $\sigma \in T^*$, the function $\pi : T^* \to \mathbb{N}^n$ associates with σ the Parikh vector $y = \pi(\sigma) \in \mathbb{N}^n$, i.e., $y(t) = k$ if transition t appears k times in σ. If σ is enabled at M_0, the reachable marking of firing σ can be computed by $M = M_0 + C \cdot \pi(\sigma)$.

[3] Given a matrix A, we use $A(\cdot, t)$ to denote the column corresponding to transition t.

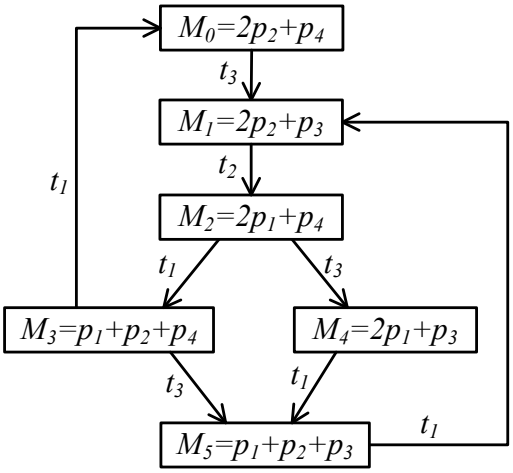

Fig. 2.10 RG of the PN system in Fig. 2.9.

The set of all markings reachable from M_0 defines the *reachability set* of $\langle N, M_0\rangle$, denoted by $R(N, M_0)$. A Petri net system is *bounded* if there exists a non-negative integer $k \in \mathbb{N}$ such that for any place $p \in P$ and any reachable marking $M \in R(N, M_0)$, $M(p) \leq k$ holds.

Given a bounded PN, its reachability set $R(N, M_0)$ can be graphically represented by the *reachability graph* (RG) that is a directed graph whose nodes are reachable markings and arcs are tagged by transitions in T. If $M_1[t\rangle M_2$ and $M_1, M_2 \in R(N, M_0)$, then M_1 and M_2 are two nodes in the RG and there is an arc from M_1 to M_2 tagged with t.

Theorem 2.1. [46] Let $\langle N, M_0\rangle$ be a PN system where N is an acyclic PN.

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

(ii) A marking M is reachable from M_0 iff there exists a nonnegative integer solution y satisfying the state equation $M = M_0 + C \cdot y$.

Example 2.10. Consider the PN system in Fig. 2.9, which is not acyclic. At M_0 only t_3 is enabled and the firing of t_3 yields $M_1 = [0\ 2\ 0\ 1]^T + [0\ 0\ 1\ -1]^T = [0\ 2\ 1\ 0]^T$. Let $\sigma = t_3t_2t_3$, which is enabled at M_0. Then $y = \pi(\sigma) = [0\ 1\ 2]^T$ and the reachable marking is $M = M_0 + C \cdot y = [2\ 0\ 1\ 0]^T$. The maximal number of tokens in a place is 2 and thus the PN system is bounded. The RG of the PN system is shown in Fig. 2.10. ◇

Definition 2.5. Given a PN system $\langle N, M_0\rangle$ and a marking M, we define the set of transition sequences enabled at M as

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

Furthermore, given a set of markings $Y \subseteq R(N, M_0)$, we define

$$L(N, Y) = \bigcup_{M \in Y} L(N, M)$$

the set of transition sequences enabled at markings in Y. ◇

Example 2.11. Consider the PN system in Fig. 2.9. $L(N, M_0) = \{t_3, t_3t_2, t_3t_2t_1, \ldots\}$. Let $M = p_1+p_2+p_4$ and $Y = \{M_0, M\}$. Then $L(N, M) = \{t_1, t_1t_3, t_1t_3t_2, \ldots\}$ and $L(N, Y) = L(N, M_0) \cup L(N, M)$. ◇

2.2.2 Labeled Petri Nets

Definition 2.6. A *labeled Petri net* (LPN) system is a 4-tuple $G = (N, M_0, \Sigma, \ell)$, where $\langle N, M_0\rangle$ is a Petri net system, Σ is an *alphabet* (a set of labels), and $\ell : T \to \Sigma \cup \{\varepsilon\}$ is a *labeling function* that assigns to each transition $t \in T$ either a symbol from Σ or the empty word ε. ◇

The set of transitions in an LPN system can be partitioned into two disjoint sets $T = T_o \cup T_u$, where $T_o = \{t \in T | \ell(t) \in \Sigma\}$ is the set of *observable* transitions, whose occurrence can be detected, and $T_u = T \setminus T_o = \{t \in T | \ell(t) = \varepsilon\}$ is the set of *unobservable/silent* transitions, whose occurrence cannot be detected.

Definition 2.7. Given a net $N = (P, T, Pre, Post)$, and the set of unobservable transitions $T_u \subseteq T$, the T_u-induced (or unobservable) subnet of N is the new net $N' = (P, T_u, Pre', Post')$, where Pre' and $Post'$ are the restriction of Pre and $Post$ to T_u. The net N' can be thought as obtained from N removing all transitions in $T \setminus T_u$. The incidence matrix of N' is denoted as $C_u = Post' - Pre'$. ◇

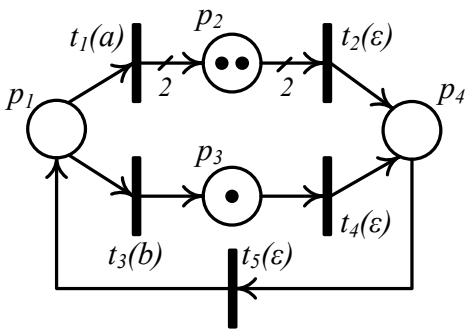

Fig. 2.11 Labeled Petri net system.

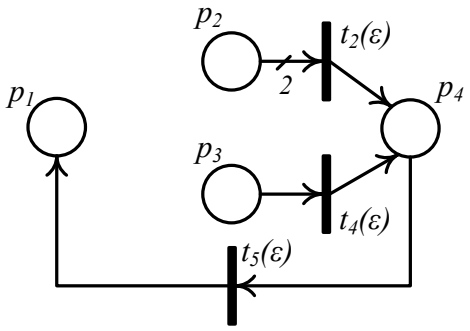

Fig. 2.12 T_u-induced subnet of the LPN in Fig. 2.11.

Example 2.12. An LPN system $G = (N, M_0, \Sigma, \ell)$ is shown in Fig. 2.11 with $\Sigma = \{a, b\}$, $\ell(t_1) = a$, $\ell(t_3) = b$, and $\ell(t_2) = \ell(t_4) = \ell(t_5) = \varepsilon$. Thus $T_o = \{t_1, t_3\}$ and $T_u = \{t_2, t_4, t_5\}$. Its T_u-induced subnet N' is shown in Fig. 2.12. The incidence matrix of N' is

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

Since there is no cycle in N', the T_u-induced subnet is acyclic. $\diamond$

The labeling function can be extended to firing sequences $\ell : T^* \to \Sigma^*$, i.e., $\ell(\sigma t) = \ell(\sigma)\ell(t)$ with $\sigma \in T^*$ and $t \in T$. The labeling function of an LPN system $G = (N, M_0, \Sigma, \ell)$ can be classified as follows

- Free: if all transitions are labeled distinctly, namely a different label is associated to each transition, and no transition is labeled with the empty word.
- Deterministic: if no transition is labeled with the empty word, and the following condition holds: for all $t, t' \in T$, with $t \neq t'$, and for all $M \in R(N, M_0) : M[t\rangle \wedge M[t'\rangle \Rightarrow [\ell(t) \neq \ell(t') \vee [C(\cdot, t) = C(\cdot, t')]$, i.e., two transitions simultaneously enabled may not share the same label or the two markings reached from M by firing t and t' are the same. This ensures that the knowledge of the firing label $\ell(t)$ is sufficient to reconstruct the marking that the firing of t yields.
- λ-free[4]: if no transition is labeled with the empty word.
- Arbitrary: if there is no restriction posed on the labeling function.

Note that all types of labeling only depend on the structure of the net, i.e., not related to M_0, but for deterministic labeling that depends both on the structure and behavior of the net. Clearly, PN systems can be regarded as a particular case of LPN systems, where the labeling function is free and $\Sigma = T$. If it is not specified, all labeling functions considered in the thesis are arbitrary.

Definition 2.8. Given an LPN system $G = (N, M_0, \Sigma, \ell)$ and a marking $M \in R(N, M_0)$, we define the *language generated* from M as

$$\mathcal{L}(G, M) = \{w \in \Sigma^* | \exists \sigma \in L(N, M) \text{ and } \ell(\sigma) = w\}.$$

[4]In the PN literature, the empty word is denoted λ, while in the formal language literature it is denoted ε. In the thesis we denote the empty word ε but, to keep consistent with the PN literature, we still use the term *λ-free* for a non erasing labeling function $\ell : T \to \Sigma$.

In particular, $\mathcal{L}(G, M_0)$ is called the *the generated language*[5] *of* G. Furthermore, given a set of markings $Y \subseteq R(N, M_0)$ of G, we define

$$\mathcal{L}(G, Y) = \bigcup_{M \in Y} \mathcal{L}(N, M)$$

the language generated from markings in Y. ◇

Given a language, if there exists a free- (resp., deterministic-, λ-free-, arbitrary-) labeled Petri net system that can generate the language, then the language is called a *free* (resp., *deterministic, λ-free, arbitrary*) Petri net language.

Given an LPN system $G = (N, M_0, \Sigma, \ell)$, and a word $w \in \Sigma^*$, the following definition defines the set of transition sequences that may produce w, the set of markings reached after w generated, and the set of markings from which w could be generated, respectively.

Definition 2.9. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system, $w \in \Sigma^*$ a word. We define

$$\mathcal{S}(w) = \{\sigma \in L(N, M_0) | \ell(\sigma) = w\}$$

as the *set of firing sequences consistent with* w,

$$\mathcal{C}(w) = \{M \in \mathbb{N}^m | \exists \sigma \in \mathcal{S}(w) : M_0[\sigma\rangle M\}$$

as the *set of markings consistent with* w, and

$$\mathcal{I}(w) = \{M \in R(N, M_0) | \exists \sigma \in T^* : M[\sigma\rangle \text{ and } \ell(\sigma) = w\}$$

as *the set of markings generating* w. ◇

Example 2.13. Consider the LPN system in Fig. 2.11. Let $w = ab$. Then $\mathcal{S}(w) = \{t_2t_4t_5t_5t_1t_3, t_4t_2t_5t_5t_1t_3, t_2t_5t_4t_5t_1t_3, t_4t_5t_2t_5t_1t_3, t_2t_5t_1t_4t_5t_3, \ldots\}$, $\mathcal{C}(w) = \{2p_2 + p_3, 2p_3, p_3 + p_4, p_3 + p_1, 2p_2 + p_4, 2p_2 + p_1, 2p_4, p_1 + p_4, 2p_1\}$, and $\mathcal{I}(w) = R(N, M_0)$. ◇

2.3 Theory of Computation

We briefly review some concepts and results from the theory of computation that are needed in the thesis. We refer the reader to [47] and [48] for more details.

[5] Thus G is called a *Petri net generator*.

2.3.1 Decidability

A *problem* is described by giving i) a general description of its parameters, and ii) a statement of what the solution is required to satisfy. An *instance* of a problem is obtained by specifying particular values for all the problem parameters. In particular, a *decision problem* is a problem whose solution is either "yes" or "no". In other words, it is any arbitrary yes-or-no question on an infinite set of inputs.

Algorithms are general, step-by-step procedures for solving problems. An algorithm is said to *solve* a problem if that algorithm can be applied to any instance of the problem and is guaranteed to output a solution for that instance. The Church-Turing thesis provides a precise definition of algorithms, which also capture the computability of a function.

Definition 2.10. A problem is said to be *decidable* if it is possible to construct an algorithm that always leads to a correct yes-or-no answer. ◇

In other words, for undecidable problems there is no algorithm that can be given at all to solve them.

2.3.2 Problem Reduction and Complexity Classes

A *reduction* is a way of converting one problem (say Problem A) to another problem (say Problem B) such that a solution to Problem B can be used to solve Problem A. Clearly, if Problem A is reducible to Problem B[6], then solving Problem A cannot be harder than solving Problem B.

Problem reduction plays an important role in determining whether a problem is decidable or not and later in classifying problems in complexity. In terms of decidability, supposing Problem A is reducible to Problem B, we have the following conclusions:

- Problem A is not decidable ⇒ Problem B is not decidable.
  (Equivalently, Problem B is decidable ⇒ Problem A is decidable.)

- Problem A is decidable,we cannot say anything about Problem B's decidability.

To study computational problems, a powerful model, called Turing machine, was proposed by Alan Turing in 1936. A Turing machine is a mathematical model of a general computing machine. The Church-Turing thesis states that if a problem can be solved by an algorithm, there exists a Turing machine that solves the problem.

[6]The complexity of reducing Problem A to Problem B should not be higher than that of solving Problem B.

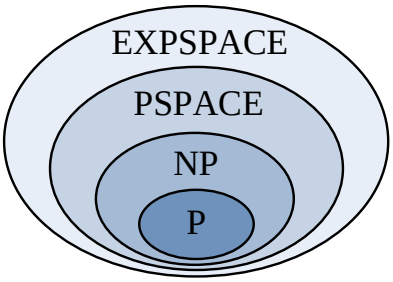

Fig. 2.13 Relation among complexity classes.

A complexity class is a set of problems of related complexity. Many complexity classes can be defined by bounding the time or space used by the algorithm.

Definition 2.11. A problem is in

- class P, if it can be solved in polynomial time by a deterministic Turing machine;
- class NP, if it can be solved in polynomial time by a non-deterministic Turing machine;
- class PSPACE, if it can be solved by a deterministic Turing machine using polynomial space;
- class EXPSPACE, if it can be solved by a deterministic Turing machine using exponential space; ◇

The relation among complexity classes is shown in Fig. 2.13.

Definition 2.12. A problem is said to be *NP-complete* if i) it is in NP, and ii) any problem in NP can be reduced to this problem in polynomial time. A problem is said to be *NP-hard* if there exists a NP-complete problem that can be reduced to it in polynomial time. ◇

NP-complete or PSPACE-complete problems are considered to be intractable, i.e., there is no known algorithm that can solve the problem efficiently.

Chapter 3 Generalized Observation Structures for Petri Net Generators

3.1 Introduction

In this chapter we study observation structures considered for Petri net generators. It is known that labeled Petri nets describe the observation structure in which only the occurrence of some transitions is observable, i.e., transitions in T_o. Two more general classes of Petri net generators are considered: *labeled Petri nets with outputs* (LPNOs) and *adaptive labeled Petri nets* (ALPNs).

- An LPNO can be thought of as a labeled Petri net endowed with additional state sensors: an *output function* provides an observation that is an arbitrary function of the current net marking. Therefore, in an LPNO there are two types of observations: event observations generated by the labeling function and state observations generated by the output function.

- An ALPN can be regarded as a labeled Petri net whose labeling function depends on the current marking, i.e., the observation produced by a transition firing may change as the net evolves.

We believe that each of these two classes of generators represents a useful modeling formalism in the system design stage providing an intuitive way to capture the logical observation structure (in terms of event and state sensors) needed to solve a control or optimization problem.

To compare the modeling power among different classes of Petri net generators, the notion of *observation equivalence* is proposed. Two Petri net generators are said to be observation equivalent if i) they have the same net structure and, ii) for an arbitrary firing sequence that occurs in the two generators, the state and sequence estimations reconstructed the two observations are identical. We point out that the notion of observation equivalence proposed in this chapter is not related to *what the observer sees* (e.g., the observed language) but rather to *what the observer can infer about the system's dynamical evolution*. Thus the results presented herein are relevant to addressing a wide range of problems that are currently investigated in the DES literature, such as state estimation,fault diagnosis or opacity [5, 11, 49].

A class of Petri net generators (say Class A) is not observation equivalent to but strictly observation weaker than another class (say Class B) means that there is no Petri net generator in Class A that is equivalent to a Petri net generator in Class B. In other words, with the same net structure the observation structure of generators in Class B can model sensors that provide more accurate measures. Therefore, we say generators in Class B have higher modeling power than those in Class A.

Ru and Hadjicostis [16] showed that for any partially observed Petri net there exists an observation equivalent LPN. In the chapter, we generalize this result to the larger class of LPNOs whose output function is an affine function, called *labeled Petri nets with an affine output function* (LPNAFs). We also show that LPNOs and ALPNs have higher modeling power than other classes of Petri net generators: LPNs and POPNs, and are not comparable between them.

Finally, we restrict our attention to bounded Petri net generators that describe systems with a finite state space. In this case we prove that any bounded LPNO can be converted into an observation equivalent ALPN. This implies that ALPNs are the class of bounded Petri net generators with higher modeling power. This motivates us to study the conversion from bounded LPNOs into ALPNs. In particular, an algorithm to convert a bounded LPNO into an observation equivalent ALPN is proposed. The algorithm relies on the *vertex coloring* of a special graph and can be used to determine the ALPN with a minimal alphabet or, if it exists, an LPN observation equivalent to the given bounded LPNO. A sufficient and necessary condition for the existence of an LPN equivalent to a given bounded LPNO is also developed. Finally, we show that in some cases the conversion is applicable to unbounded LPNOs. Such a conversion procedure has its merit: if an automatic conversion procedure from LPNOs to ALPNs is available, it is sufficient to derive approaches for analysis and design of the most general class ALPNs rather than for each model. As a particular case, in this conversion an LPN may be obtained and several results concerning this model have already been presented in the literature [5, 9–12].

This chapter is organized as follows. Formal definitions of labeled Petri nets with outputs and adaptive labeled Petri nets are presented in Section 3.2. In Section 3.3 we formally state the notion of observation equivalence based on which the modeling power between different classes of Petri net generators is compared. An algorithm that converts a bounded LPNO into an observation equivalent ALPN is developed in Section 3.4 where the number of labels of the observation equivalent ALPN is also discussed. Then in Section 3.5, a sufficient and necessary condition for the existence of the observation equivalent LPN

to a bounded LPNO is reported, and the corresponding conversion algorithm is presented. In Section 3.6 the conversion of unbounded LPNOs is studied. Finally, conclusions are presented.

3.2 Petri Net Generators

3.2.1 Labeled Petri Nets (LPNs)

We consider the case in which an external agent (e.g., the supervisor in a supervisory control problem, or the intruder in an opacity problem) that knows the initial marking and the structure of the PN but observes the firing of transitions through a mask. A common assumption is that of considering the mask as a projection from the set of transitions T to an alphabet Σ which represents available sensors readings[11, 49, 50]. Such a mechanism is illustrated in Fig. 3.1. The mask is possibly evasive, i.e., the output label assigned to a transition may either be a symbol from the alphabet or the empty string ε to denote that the firing of the transition does not produce an observable reading. A transition of the latter type is said to be *unobservable* (or *silent*). Such an observation structure can be formalized as *labeled Petri nets* (LPN), which has been defined in Definition 2.6. To be more clear, we still recall its definition here and it is denoted as G_L.

Definition 3.1. A *labeled Petri net* (LPN) is a generator $G_L = (N, M_0, \Sigma, \ell)$, where $\langle N, M_0\rangle$ is a Petri net system, Σ is an alphabet, and $\ell : T \to \Sigma \cup \{\varepsilon\}$ is a labeling function that assigns to each transition $t \in T$ either a symbol from Σ or the empty word ε. $\diamond$

Given a firing sequence σ, the corresponding observation generated by the observation function is formally defined as follows.

Definition 3.2. Given an LPN $G_L = (N, M_0, \Sigma, \ell)$, the *observation function* of G_L is defined as a mapping $L_L : T^* \to \Sigma^*$ that associates a firing sequence $\sigma = t_1 t_2 \cdots t_k$ with the observation $w = L_L(\sigma) = \ell(t_1)\ell(t_2)\cdots\ell(t_k)$. $\diamond$

For LPNs, its observation function is identical to the labeling function. Since we consider more than one observation structures in this chapter, with a little abuse of notation, we still define the observation function for LPNs.

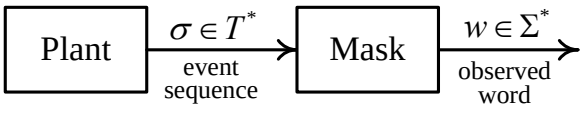

Fig. 3.1 Observation mask.

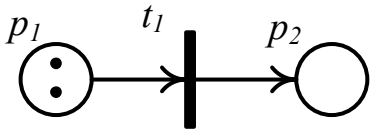

Fig. 3.2 Petri net system.

Example 3.1. Consider an LPN $G_L = (N, M_0, \Sigma, \ell)$, where $\langle N, M_0\rangle$ is shown in Fig. 3.2, $\Sigma = \{a\}$ and $\ell(t_1) = a$. Let $\sigma = t_1 t_1$. Then the observation produced is $w = L_L(\sigma) = aa$.$\diamond$

3.2.2 Partially Observed Petri Nets (POPNs)

In addition to sensors that detect the firing of transitions, it is also possible to have sensors that provide information on the markings of a net. Several researchers studied Petri nets where some places are observable, i.e., their token content [13–16], or even more general, a linear combination of their token contents [18] is known. Such observation structures can be formalized as follows.

Definition 3.3. A *partially observed Petri net* (POPN) is a generator $G_P = (N, M_0, \Sigma, \ell, V)$, where (N, M_0, Σ, ℓ) is an LPN, $P_o \subset P$ is a set of *observable places* and $V \in \mathbb{R}^{|P_O|\times|P|}$ is a *place sensor configuration*, where $\mathbb{R}$ denotes the set of real numbers. $\diamond$

The observations in a POPN are strings of triple (observation of the start state, label of the transition, observation of the reached state).

Definition 3.4. Let $G_P = (N, M_0, \Sigma, \ell, V)$ be a POPN and $\sigma = t_1 \cdots t_k$ be a firing sequence with $M_0[t_1\rangle M_1 \cdots M_{k-1}[t_k\rangle M_k$. The *observation function* of G_P is defined as a mapping $L_P : T^* \to (\mathbb{R}^{|P_o|} \times \Sigma \times \mathbb{R}^{|P_o|})^*$ that associates sequence σ with the observation

$$s_P = L_P(\sigma) = (M_{V0}, \ell(t_1), M_{V1}) \cdots (M_{Vk-1}, \ell(t_k), M_{Vk}),$$

where $M_{Vi} = V \cdot M_i$ and $i \in \{0, 1, 2, \cdots, k\}$.

As a particular case, $(M_{Vi}, \ell(t), M_{Vj})$ is defined as the null observation, if $\ell(t) = \varepsilon$ and $M_{Vi} = M_{Vj}$. $\diamond$

Example 3.2. Consider a POPN $G_P = (N, M_0, \Sigma, \ell, V)$, where $\langle N, M_0\rangle$ is shown in Fig. 3.2, $\Sigma = \emptyset$, $\ell(t_1) = \varepsilon$, and $V = [0\ 1]$. Let $\sigma = t_1 t_1$. Then we have $M_0[t_1\rangle M_1[t_1\rangle M_2$, where $M_1 = [1\ 1]^T$ and $M_2 = [0\ 2]^T$. Therefore, $M_{V0} = V \cdot M_0 = 0$, $M_{V1} = V \cdot M_1 = 1$ and $M_{V2} = V \cdot M_2 = 2$. The corresponding observation is $s_P = L_P(\sigma) = (0, \varepsilon, 1)(1, \varepsilon, 2)$. $\diamond$

3.2.3 Labeled Petri Nets with Outputs (LPNOs)

Partially observed Petri nets consider a very particular class of state observations where the exact marking of some places, or a linear combination of the markings, is measured. However, in many cases a sensor may provide more general information about the state: consider, as an example, the case of a buffer where a sensor only detects if the buffer is empty or not. This motivated us to define a class of labeled Petri nets endowed with a general observation function associated to state sensors.

Definition 3.5. A *labeled Petri net with outputs* (LPNO) is a generator $G_O = (N, M_0, \Sigma, \ell, f)$, where (N, M_0, Σ, ℓ) is an LPN and $f : R(N, M_0) \to \mathbb{R}^k$ is an *output function* associated with $k \in \mathbb{N}$ state sensors. $\diamond$

In an LPNO there are two types of observations: transition labels and marking information.

Definition 3.6. Given an LPNO $G_O = (N, M_0, \Sigma, \ell, f)$, let $\sigma = t_1 \cdots t_k$ be a firing sequence producing the trajectory $M_0[t_1\rangle M_1 \cdots M_{k-1}[t_k\rangle M_k$. The *observation function* of G_O is defined as a mapping $L_O : T^* \to (\Sigma \times \mathbb{R}^k)^*$ that associates σ with the observation

$$s = L_O(\sigma) = (\ell(t_1), \Delta f(M_0, t_1)) \cdots (\ell(t_k), \Delta f(M_{k-1}, t_k)),$$

where $\Delta f(M_{i-1}, t_i) = f(M_i) - f(M_{i-1}) \in \mathbb{R}^k$, $i = 1, 2, \cdots, k$.

If $\ell(t_i) = \varepsilon$ and $\Delta f(M_{i-1}, t_i) = 0$, the observation $(\varepsilon, 0)$ is the *null observation* as no transition firing is detected. $\diamond$

Remark: since the initial marking is assumed to be known, the initial observation $f(M_0)$ provides no additional information. In this case the two sequences $f(M_0), f(M_1), \ldots$ and $\Delta f(M_0, t_1), \Delta f(M_1, t_2), \ldots$ contain the same information. This also implies that the observation s_P in a POPN (see Definition 3.4) contains the same information as the observation $(\ell(t_1), V \cdot M_1 - V \cdot M_0) \cdots (\ell(t_k), V \cdot M_k - V \cdot M_{k-1})$ and we can conclude that POPNs are a special subclass of LPNOs whose output function is $f(M) = V \cdot M$.

Example 3.3. Consider an LPNO $G_O = (N, M_0, \Sigma, \ell, f)$, where $\langle N, M_0\rangle$ is shown in Fig. 3.2, $\Sigma = \emptyset$, $\ell(t_1) = \varepsilon$, and the output function is $f(M) = min\{M(p_2), 1\}$. Let the firing sequence be $\sigma = t_1 t_1$. Based on the result in Example 3.2, we have $f(M_0) = 0$, $f(M_1) = 1$ and $f(M_2) = 1$. Therefore, $\Delta f(M_0, t_1) = 1$ and $\Delta f(M_1, t_1) = 0$. The corresponding observation would be $s = (\varepsilon, 1)$, since the second firing of t_1 produces the null observation $(\varepsilon, 0)$. $\diamond$

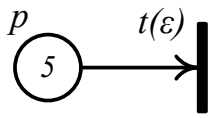

Fig. 3.3 LPNO model of the manufacturing cell in Example 3.4.

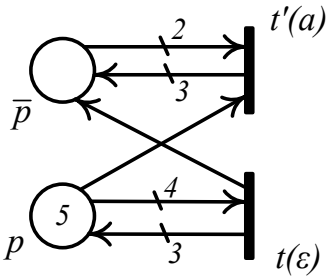

Fig. 3.4 LPN model of the manufacturing cell in Example 3.4.

The following example shows that LPNOs provide an intuitive way to model systems with arbitrary state sensors.

Example 3.4. Consider the net in Fig. 3.3 describing a manufacturing cell: there is a buffer modeled by places p, and a robot modeled by transition t that moves products. The action of the robot is not detectable, i.e., transitions t is labeled with the empty string. However, on the buffer there is a counter whose measuring range is from 0 to 3: if the content is lower than three, the device counts the products in p; otherwise a saturation will be reached. Therefore, the output function is

$$f(M) = \begin{cases} 3, & \text{if } M(p) \geq 3; \\ M(p), & \text{otherwise.} \end{cases}$$

We note that it may also be possible to use LPNs to describe this system since place p is 5-bounded, i.e., for all reachable markings M it holds $M(p) \leq 5$. The corresponding LPN is the much less intuitive net shown in Fig. 3.4. Here place $\overline{p}$ is the complementary place of p (i.e., $M(p) + M(\overline{p}) = 5$) and t' is a duplicate of t. If $M(p) \geq 3$, t is activated; otherwise, t' is activated. The LPN has a larger size and, moreover, if the bound of p or the range of the counter changes, the LPN structure has to be changed. In addition, if p is unbounded, no LPN can model the system, since no complementary place can be defined. ◇

We next define a particular subclass of LPNOs called labeled Petri nets with an affine output function.

Definition 3.7. A *labeled Petri net with an affine output function* (LPNAF) is an LPNO $G_O = (N, M_0, \Sigma, \ell, f)$ whose output function is an affine function $f(M) = A \cdot M + B$ with constant matrices $A \in \mathbb{R}^{k \times m}$ and $B \in \mathbb{R}^k$. ◇

Note that the POPNs considered in [13–16, 18] are all subclasses of LPNAF where matrix $B = 0$.

Example 3.5. Consider an LPNAF $G_O = (N, M_0, \Sigma, \ell, f)$, where $\langle N, M_0 \rangle$ is shown in Fig. 3.2, $\Sigma = \emptyset$, $\ell(t_1) = \varepsilon$ and $f(M) = A \cdot M + B$ with $A = [1 \ -1.5]$ and $B = 2$. Let $\sigma = t_1 t_1$. The corresponding observation would be $s = (\varepsilon, -2.5)(\varepsilon, -2.5)$. ◇

3.2.4 Adaptive Labeled Petri Nets (ALPNs)

In the framework of DESs with partial event observations, it is usually assumed that the observation corresponding to an event is static, i.e., it does not change as the system evolves. However, there are some situations where the observation produced by the occurrence of an event also depends on the states. Let us consider, as an example, the case of a sensor that may be turned off in some states or where communication failures change the observation. Some studies have considered this paradigm in DESs modeled by automata [51–54]. To the best of our knowledge, it has never been defined in the framework of Petri nets, which motivated us to define a Petri net generator where the labeling function depends on the state: we call it an adaptive labeled Petri net.

Definition 3.8. An *adaptive labeled Petri net* (ALPN) is a generator $G_A = (N, M_0, \Sigma_A, \ell_A)$, where $\langle N, M_0 \rangle$ is a Petri net system, Σ_A is an alphabet and $\ell_A : R(N, M_0) \times T \to \Sigma_A \cup \{\varepsilon\}$ is an *adaptive labeling function*. ◇

According to the definition of ALPNs, the label assigned to a transition need not be fixed but may depend on the states. However, the observation corresponding to a firing sequence is a string of labels the same as the one in an LPN.

Definition 3.9. Given an ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$, let $\sigma = t_1 \cdots t_k$ be a firing sequence producing the trajectory $M_0[t_1\rangle M_1 \cdots M_{k-1}[t_k\rangle M_k$. The *observation function* of G_A is defined as a mapping $L_A : T^* \to \Sigma_A^*$ that associates sequence σ with the observation

$$w_A = L_A(\sigma) = \ell_A(M_0, t_1) \cdots \ell_A(M_{k-1}, t_k).$$ ◇

Example 3.6. Consider an ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$, where $\langle N, M_0 \rangle$ is shown in Fig. 3.2, $\Sigma_A = \{a\}$, and the adaptive labeling function is $\ell_A(M_0, t_1) = a$ and $\forall M \in \{[1\ 1]^T, [0\ 2]^T\}$, $\ell_A(M, t_1) = \varepsilon$. Let the firing sequence still be $\sigma = t_1 t_1$. The observation would be $w_A = a$. ◇

The following example shows that ALPNs provide an intuitive way to model systems with state dependent event labels.

Example 3.7. Consider the manufacturing cell modeled by the ALPN in Fig. 3.5, where transition t_1 represents the operation of a robot moving products from an upstream buffer

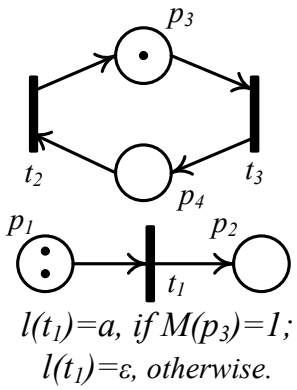

Fig. 3.5 ALPN model of the system with sensors switched on/off.

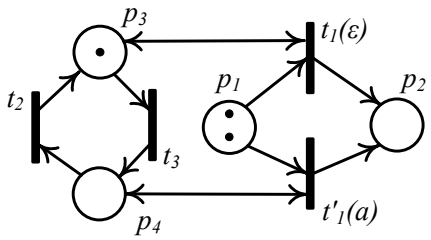

Fig. 3.6 LPN model of the system with sensors switched on/off.

(p_1) to a downstream buffer (p_2). A sensor on the robot may be turned on (place p_3 is marked) and off (place p_3 is empty) by suitable commands (transitions t_2 and t_3). When the sensor is on, the operation of the robot is detected, otherwise it is unobservable. We model this with a state dependent label

$$\ell(t_1) = \begin{cases} a, & \text{if } M(p_3) \geq 0; \\ \varepsilon, & \text{otherwise.} \end{cases}$$

Note that in this particular case the system can be modeled by the LPN in Fig. 3.6 as well, where t_1' is a duplicate of t_1. However, such an LPN model has a larger size and is less intuitive. ◇

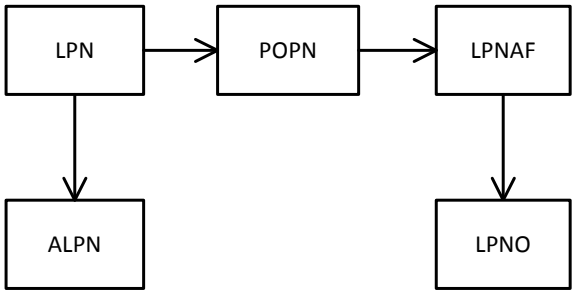

Fig. 3.7 Structural relationships between generators.

From a structural point of view, the relationships between the classes of generators previously defined can be summarized in Fig. 3.7. For each arc, the class corresponding to

the head node is more general than that corresponding to the tail node.

3.3 Structural Relationships and Observation Equivalence

In the previous section we have compared the different generators introduced in this chapter in terms of structural relationships. Here we address the problem of comparing them in terms of modeling power by introducing an appropriate notion of observation equivalence.

We point out a fact: if a model is structurally more general than another, it does not necessarily mean that it has greater modeling power. As an example, it is well known that NFAs are a generalization of DFAs but as far as the languages are concerned, the two models have the same power. In Chapter 2, we have presented the well known procedure to convert an NFA into an equivalent deterministic one that accepts the same language.

We assume that the purpose of observing a system is that of reconstructing both the sequence of events that has occurred and the current state of the system. To this end, we propose a notion of observation equivalence that applies to generators having the same underlying net structure but a different observation structure: two generators are observation equivalent if their observation structures provide the same information on the transition firings and on the markings.

In the following let

$$\mathcal{J} = \{LPN, POPN, LPNAF, LPNO, ALPN\}$$

denote the set of all these classes of generators.

Definition 3.10. Consider a generator G in class $\mathcal{X} \in \mathcal{J}$, whose underlying net system $\langle N, M_0 \rangle$ is assumed to be known. Let L be its observation function, and x an observation. We define:

- the set of *firing sequences consistent with* x as

$$\mathcal{S}(x) = \{\sigma \in L(N, M_0) \mid L(\sigma) = x\};$$

- the set of *markings consistent with* x as

$$\mathcal{C}(x) = \{M \in \mathbb{N}^m \mid \exists \sigma \in \mathcal{S}(x) : M_0[\sigma\rangle M\}. \qquad \diamond$$

Using these sets we define the notion of observation equivalence between generators.

Table 3.1 Firing estimates in G_O and G_A

		$\sigma = \varepsilon$	$\sigma = t_1$	$\sigma = t_1 t_1$
G_O	s	ε	$(\varepsilon, 1)$	$(\varepsilon, 1)$
	$\mathcal{S}(s)$	$\{\varepsilon\}$	$\{t_1, t_1 t_1\}$	$\{t_1, t_1 t_1\}$
G_A	w_A	ε	a	a
	$\mathcal{S}(w_A)$	$\{\varepsilon\}$	$\{t_1, t_1 t_1\}$	$\{t_1, t_1 t_1\}$

Definition 3.11. A generator G in class $\mathcal{X}$ is said to be *observation equivalent* to a generator G' in class $\mathcal{X}'$ if the following two conditions hold:

i) G and G' have the same net system $\langle N, M_0 \rangle$,

ii) for any sequence $\sigma \in L(N, M_0)$ that produces an observation x in G and an observation x' in G', $\mathcal{S}(x) = \mathcal{S}(x')$ holds. $\diamond$

Note that in Definition 3.11, $\mathcal{S}(x) = \mathcal{S}(x')$, together with condition i), implies $\mathcal{C}(x) = \mathcal{C}(x')$. In this chapter, "equivalence" always refers to "observation equivalence". The notion of observation equivalence between generators induces a meaningful relationship between classes of generators.

Example 3.8. Consider the LPNO G_O in Example 3.3 and the ALPN G_A in Example 3.6. These two generators are observation equivalent, since they have the same net system and according to Table 3.1, for all $\sigma \in T^*$ it holds $\mathcal{S}(L_O(\sigma)) = \mathcal{S}(L_A(\sigma))$. $\diamond$

Definition 3.12. Given two classes of Petri net generators $\mathcal{X}, \mathcal{X}' \in \mathcal{J}$, class $\mathcal{X}$ is said to be *observation weaker* than $\mathcal{X}'$ if for any generator G in class $\mathcal{X}$ there exists an observation equivalent generator G' in class $\mathcal{X}'$. This relation is denoted by

$$\mathcal{X} \preccurlyeq \mathcal{X}'.$$

We also write:

- $\mathcal{X} \approx \mathcal{X}'$ if $\mathcal{X} \preccurlyeq \mathcal{X}'$ and $\mathcal{X}' \preccurlyeq \mathcal{X}$: in this case we say that the two classes are *observation equivalent*;
- $\mathcal{X} \precneqq \mathcal{X}'$ if $\mathcal{X} \preccurlyeq \mathcal{X}'$ and $\mathcal{X}' \not\preccurlyeq \mathcal{X}$ hold[1]: in this case we say that class $\mathcal{X}$ is *strictly observation weaker* than $\mathcal{X}'$;
- $\mathcal{X} \nsim \mathcal{X}'$ if $\mathcal{X} \not\preccurlyeq \mathcal{X}'$ and $\mathcal{X}' \not\preccurlyeq \mathcal{X}$ hold: in this case we say that the two classes are *not observation comparable*. $\diamond$

[1]Here $\mathcal{X}' \not\preccurlyeq \mathcal{X}$ denotes that the relation $\mathcal{X}' \preccurlyeq \mathcal{X}$ does not hold, i.e., there exists at least one generator in $\mathcal{X}'$ such that there is no generator in $\mathcal{X}$ observation equivalent to it.

Obviously if class $\mathcal{X}'$ is structurally more general than $\mathcal{X}$ (see Fig. 3.7), then $\mathcal{X} \preccurlyeq \mathcal{X}'$ holds; here we complete the analysis by discussing when two classes are observation equivalent or not comparable.

3.3.1 LPNs, POPNs and LPNAFs

In this section we show that LPNs, POPNs and LPNAFs are observation equivalent. This generalizes a result by Ru and Hadjicostis [16] who proved the equivalence between LPNs and POPNs.

Proposition 3.1. LPNs, POPNs and LPNAFs are observation equivalent, i.e., $LPN \approx POPN \approx LPNAF$.

Proof: The relationship $LPN \preccurlyeq POPN \preccurlyeq LPNAF$ immediately follows from the structural relationship in Fig. 3.7. We now complete the proof by showing that $LPNAF \preccurlyeq LPN$. To do this we provide a constructive procedure that, given an arbitrary LPNAF $G_O = (N, M_0, \Sigma, \ell, f)$ with $f = A \cdot M + B$, determines an equivalent LPN $G_L = (N, M_0, \Sigma', \ell')$.

Given an LPNAF G_O, let $T_e = \{t \in T | \ell(t) = e\}$ with $e \in \Sigma \cup \{\varepsilon\}$ be the set of transitions that have the same label e and C_e be the incidence matrix restricted to T_e. For any $e \in \Sigma$, set T_e is further divided into $T_e = T_{e1} \cup \cdots \cup T_{el}$ such that $\forall t \in T_{ei}$ ($i \in \{1, 2, \cdots, l\}$) the corresponding columns $C_e^A(\cdot, t)$ of matrix $C_e^A = A \cdot C_e$ are identical. For $e = \varepsilon$, set T_ε is divided into $T_\varepsilon = T_{\varepsilon 0} \cup T_{\varepsilon 1} \cup \cdots \cup T_{\varepsilon l}$ such that $\forall t \in T_{\varepsilon 0}$, the corresponding columns $C_\varepsilon^A(\cdot, t) = \vec{0}$ and $\forall t \in T_{ei}$ ($i \in \{1, 2, \cdots, l\}$), the corresponding columns $C_\varepsilon^A(\cdot, t)$ are identical. Then the equivalent LPN $G_L = (N, M_0, \Sigma', \ell')$ has labeling: $\forall e \in \Sigma \cup \{\varepsilon\}$, $\forall t \in T_{ei}$ with $i \in \{1, \cdots, l\}$, $\ell'(t) = ei$ and $\forall t \in T_{\varepsilon 0}$, $\ell'(t) = \varepsilon$. In the following, we prove that G_L is equivalent to G_O.

Let transition $t' \in T$ fire at marking $M \in R(N, M_0)$ with $M[t'\rangle M'$. The corresponding observation in G_O would be $s = (\ell(t'), \Delta f)$, where $\ell(t') = e$ and

$$\begin{aligned}
\Delta f &= f(M') - f(M) \\
&= A \cdot M' + B - (A \cdot M + B) \\
&= A \cdot (M' - M) \\
&= A \cdot C(\cdot, t') \\
&= C_e^A(\cdot, t').
\end{aligned}$$

Therefore, for G_O the set of firing transitions consistent with s from marking M is $\mathcal{S}(s) = \{t \in T_e | M[t\rangle \wedge C_e^A(\cdot, t) = \Delta f\}$. Assume that the observation in G_L is $w = ej$.

For G_L the set of firing transitions consistent with w from marking M is $\mathcal{S}(w) = \{t \in T_{ej} | M[t\rangle\}$. According to the definition of T_{ej}, we have $\forall t \in \mathcal{S}(w), C_e^A(\cdot, t) = C_e^A(\cdot, t') = \Delta f$. Namely, $\mathcal{S}(w) = \mathcal{S}(s)$. Furthermore, it also indicates that, given a transition t, at every marking where transition t is enabled, the firing of t will cause the same observation $(\ell(t), f(M') - f(M))$. Thus the proof can be easily extended to firing sequences. □

Example 3.9. Consider the LPNAF $G_O = (N, M_0, \Sigma, \ell, f)$ in Example 3.5, whose incidence matrix is $C = [-1\ 1]^T$. Based on the constructive procedure in the proof of Proposition 3.1, for transition t_1, we have that $\Delta f = A \cdot C(\cdot, t_1) = -2.5$, different from 0. Therefore, the equivalent LPN is $G_L = (N, M_0, \Sigma', \ell')$, where $\ell'(t) = a$ and $\Sigma' = \{a\}$. ◇

3.3.2 LPNs and LPNOs

In this section we discuss the observation relationship between LPNs and LPNOs.

Proposition 3.2. LPNs are strictly observation weaker than LPNOs, i.e., $LPN \precneqq LPNO$.

Proof: Fig. 3.7 shows that LPNOs are structurally more general than LPNs, which implies $LPN \preccurlyeq LPNO$. According to Definition 3.12, it is sufficient to prove $LPNO \npreccurlyeq LPN$ by giving an LPNO whose equivalent LPN does not exist.

Consider the LPNO G_O in Example 3.3. Assume that there is an LPN $G_L = (N, M_0, \Sigma', \ell')$ equivalent to G_O. Since the labeling function in G_L is static, the labeling function only could be $\ell'(t_1) = \varepsilon$ or $\ell'(t_1) = a$, i.e., transition t_1 in G_L is either observable or not.

- Assume that the labeling function in G_L is $\ell'(t_1) = \varepsilon$. At the initial marking, the firing of t_1 will produce the observation $w = \varepsilon$ in G_L. The set of firing sequences consistent with w is $\mathcal{S}(w) = \{\varepsilon, t_1, t_1t_1\}$. On the other hand, in G_O the corresponding observation is $s = (\varepsilon, 1)$, and thus the set of possible firing sequences is $\mathcal{S}(s) = \{t_1, t_1t_1\}$. According to Definition 3.11, these two generators are not equivalent.

- Assume the labeling function in G_L is $\ell'(t_1) = a$. At the initial marking, the firing of t_1 will produce the observation $w = a$ in G_L and $\mathcal{S}(w) = \{t_1\}$, while in G_O, the observation is $s = (\varepsilon, 1)$ and $\mathcal{S}(s) = \{t_1, t_1t_1\}$. Therefore, G_O and G_L are still not equivalent. In conclusion, there is no LPN equivalent to G_O.

□

From the equivalence between LPNs, POPNs and LPNAFs, a result also follows.

Corollary 3.1. POPNs and LPNAFs are strictly observation weaker than LPNOs.

3.3.3 LPNs and ALPNs

Now we consider the observation relation between LPNs and ALPNs, two classes of generators where only event occurrences are observed.

Proposition 3.3. LPNs are strictly observation weaker than ALPNs, i.e., $LPN \precneqq ALPN$.

Proof: The relationship $LPN \preccurlyeq ALPN$ trivially follows from the structural relationship in Fig. 3.7. Now we prove $ALPN \not\preccurlyeq LPN$ by giving an ALPN whose equivalent LPN does not exist.

Consider the ALPN G_A in Example 3.6. Assume that there is an LPN $G_L = (N, M_0, \Sigma, \ell)$ equivalent to G_A. G_L has the same net system $\langle N, M_0 \rangle$ with G_A. The possible labeling function of G_L is either $\ell(t_1) = \varepsilon$ or $\ell(t_1) = b \in \Sigma$. Namely, in G_L transition t_1 is either unobservable or observable. Let $\ell(t_1) = b$ (the case that transition t_1 is unobservable can be proved in the same way) and a firing sequence be $\sigma = t_1 t_1$. Then, the corresponding observations in G_A and G_L are $w_A = a$ and $w = bb$, respectively. Therefore, in G_A, the set of firing sequences consistent with w_A is $\mathcal{S}(w_A) = \{t_1, t_1 t_1\}$; in G_L, the set of firing sequences consistent with w is $\mathcal{S}(w) = \{t_1 t_1\}$, i.e., $\mathcal{S}(w_A) \neq \mathcal{S}(w)$. We conclude that G_L is not equivalent to G_A. There is no LPN equivalent to G_O. □

From the equivalence between LPNs, POPNs and LPNAFs, the following result is drived.

Corollary 3.2. POPNs and LPNAFs are strictly observation weaker than ALPNs.

3.3.4 LPNOs and ALPNs

Fig. 3.7 shows that there is no specific structural relation between LPNOs and ALPNs. In this section, we will show that these classes are not comparable either with respect to the observation equivalence relation.

Proposition 3.4. ALPNs and LPNOs are not observation comparable, i.e., $ALPN \nsim LPNO$.

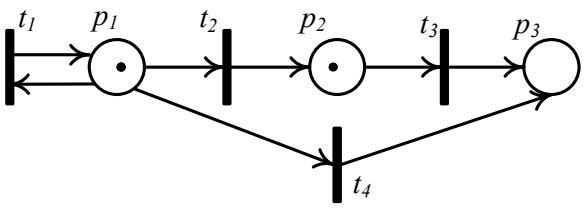

Fig. 3.8 ALPN that cannot be converted into an LPNO.

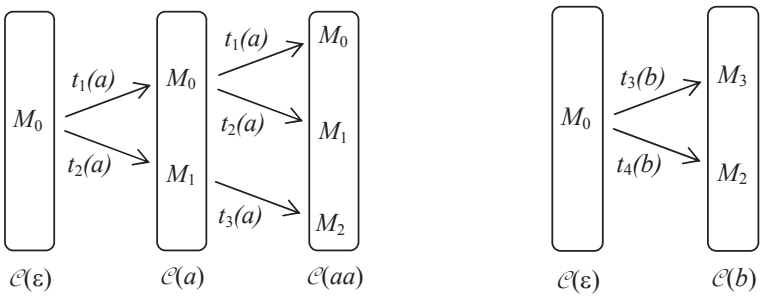

Fig. 3.9 Computation of sets $\mathcal{C}(aa)$ and $\mathcal{C}(b)$ for the ALPN.

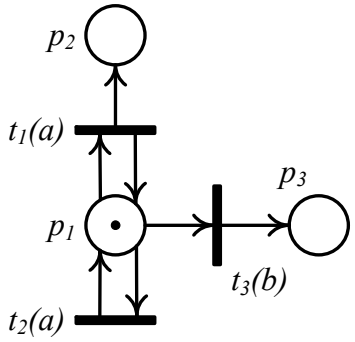

Fig. 3.10 LPNO whose equivalent ALPN has an infinite number of labels.

Proof: a) First, we prove that $ALPN \not\preccurlyeq LPNO$ is true by means of an example. Let us consider the ALPN in Fig. 3.8 with initial marking $M_0 = [1\ 1\ 0]^T$ and the adaptive labeling function given by Table 3.2.

Table 3.2 Adaptive labeling function of the ALPN

$\ell_A(M,t)$	t_1	t_2	t_3	t_4
M_0	a	a	b	b
$M \neq M_0$	b	b	a	a

For observed words $w_A = aa$ and $w_A = b$, the sets $\mathcal{S}(aa)$ and $\mathcal{S}(b)$ can be iteratively computed as shown in Fig. 3.9, where $M_1 = [0\ 2\ 0]^T$, $M_2 = [0\ 1\ 1]^T$ and $M_3 = [1\ 0\ 1]^T$.

We claim that there does not exist an LPNO equivalent to this ALPN. We prove this by contradiction. If we assume that such a generator exists, then its output function necessarily satisfies $f(M_0) = f(M_1) = f(M_2) = f(M_3)$ since otherwise we would be able to distinguish between the three firing sequences $\sigma_1 = t_1t_1, \sigma_2 = t_1t_2$ and $\sigma_3 = t_2t_3$ after the firing of two a's or between the two firing sequences $\sigma_4 = t_3$ and $\sigma_5 = t_4$ after the firing of b. In addition, all transitions necessarily have the same label, say a. However such an LPNO after a would produce a set of consistent firing sequences $\mathcal{S}((a,0)) = \{t_1, t_2, t_3, t_4\} \neq \mathcal{S}(a) = \{t_1, t_2\}$, which contradicts the assumption.

b) Now we show that $LPNO \not\preccurlyeq ALPN$ is true by means of another example. Consider the LPNO in Fig. 3.10, where the output function is

$$f(M) \begin{cases} 0, & \text{if } M(p_3) = 0; \\ M(p_2), & \text{otherwise.} \end{cases}$$

If the observation is $s = \underbrace{(a,0)(a,0)\cdots(a,0)}_{k}(b,x)$, x could be any number from 0 to k. To find the equivalent ALPN we have to assign infinite labels to transition t_3: $\ell_A(M_i, t_3) = [b, i]$, where $M_i = [0\ 1\ i]^T$ for $i \in \mathbb{N}$. In that case the equivalent ALPN needs an infinite alphabet, a condition that is not consistent with Definition 3.8. □

Even though LPNOs and ALPNs are not observation comparable, when generators whose underlying net system is bounded are considered, bounded LPNOs are strictly observation weaker than bounded ALPNs.

Proposition 3.5. Bounded LPNOs ($LPNO_{bound}$) are strictly observation weaker than bounded ALPNs ($ALPN_{bound}$), i.e., $LPNO_{bound} \precneqq ALPN_{bound}$.

Proof: The relation $ALPN_{bound} \not\preccurlyeq LPNO_{bound}$ follows from part a) of the proof of Proposition 3.4. Thus we are left to prove $LPNO_{bound} \preccurlyeq ALPN_{bound}$. To show this, we present a brute force approach that determines the equivalent ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$ of a bounded LPNO $G_O = (N, M_0, \Sigma, \ell, f)$. Given a bounded G_O, the adaptive labeling function of its equivalent ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$ can be determined by the following rule: for any $t \in T$ and $M \in R(N, M_0)$ with $M[t\rangle M'$, $\ell_A(t) = [\ell(t), f(M') - f(M)]$, i.e., the corresponding observation in G_O is assigned as a label to the transition in G_A. The alphabet of G_A is a finite set $\Sigma_A = \{[\ell(t), f(M') - f(M)] | t \in T, M \in R(N, M_0), M[t\rangle M'\}$, since G_O is bounded. Once the transition fires in G_A, the new label exactly describes the observation of G_O and the sets of firing sequences consistent with the observations in G_A and G_O must be identical. Thus the two generators are equivalent. □

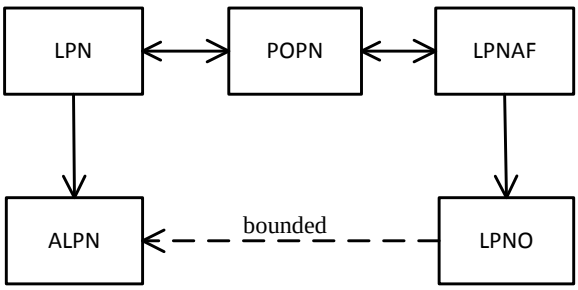

Fig. 3.11 Observation relationships between generators.

In conclusion, equivalence relations between all classes of Petri net generators discussed in this chapter are illustrated in Fig. 3.11. A double arrowed arc $\leftrightarrow$ connects two

classes that are observation equivalent while an arrow $\rightarrow$ denotes that the class at the tail is strictly observation weaker than the one at the head. The arrow tagged "bounded" denotes that bounded LPNOs are strictly observation weaker than bounded ALPNs.

3.4 Conversion of Bounded LPNOs into ALPNs

As mentioned in the introduction, bridges between different formalisms have both theoretical significance and practical relevance. The conversion between LPNs, POPNs, and LPNAFs was discussed in the proof of Proposition 3.1. According to the structural relations shown in Fig. 3.7, POPNs and LPNAFs are both subclasses of LPNOs, and LPNs is a subclass of both ALPNs and LPNOs. Therefore, the procedure to convert LPNOs to ALPNs can be also applied to convert generators of all those subclasses to an equivalent ALPNs. Moreover, ALPNs are the class that has the highest modeling power among bounded Petri net generators. For this reason, in the rest of this chapter we focus on the conversion from LPNOs to ALPNs.

In this section we present an algorithm to convert a bounded LPNO into an equivalent ALPN with a minimal number of labels. The interest for finding a minimal alphabet relies on the following observations: i) applying the brute force approach (in Proposition 3.5) may introduce unnecessary labels; if we consider the cardinality of the alphabet corresponding to the number of event sensors in the system, reducing the cardinality of the alphabet leads to a cost reduction in the implementation of an observation structure; ii) it may allow us to determine an equivalent net with a finite alphabet even when the brute-force procedure generates an infinite number of labels (we will give such an example in Section 3.6); and iii) this procedure may allow us to verify that a given LPNO cannot be converted into an LPN, which will be discussed in Section 3.5.

The proposed conversion algorithm reduces the computation of the adaptive labeling function of the equivalent ALPN to solving the *vertex coloring problem* [55] of a graph called a *conflict graph*. A running example illustrates the algorithm. We assume that LPNOs discussed in this section and the following two are bounded.

3.4.1 Problem Reduction

According to Definition 3.11, two equivalent generators have the same net system. Thus, given an LPNO $G_O = (N, M_0, \Sigma, \ell, f)$, to compute its equivalent ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$, we only need to determine the adaptive labeling function. We show that this issue can be reduced to solving a *vertex coloring*. The proposed procedure requires three

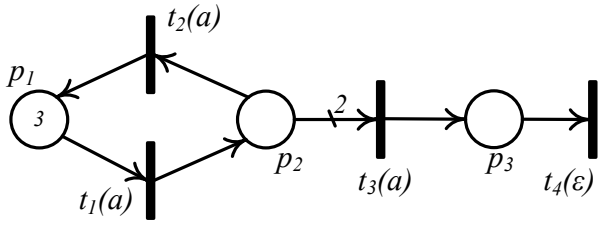

Fig. 3.12 LPNO with a nonlinear output function.

main steps.

Step 1 Since observation equivalence requires the set of consistent markings of the two generators to be identical for all observations, we first determine which pairs of markings are *confusable*, i.e., belong to the same consistent set for some observations.

Step 2 Using this information, we determine which pairs $[M, t] \in R(N, M_0) \times T$ should have the same label in the ALPN constructing the *agreement graph* $\mathcal{G}_A$. We also determine which pairs $[M, t]$ should have a different label in the ALPN constructing the *conflict graph* $\hat{\mathcal{G}}_A$.

Step 3 Finally, the problem of finding the label assignment that determines the equivalent ALPN is reduced to solving the vertex coloring of graph $\hat{\mathcal{G}}_A$.

3.4.2 Computation of the Confusion Relation

Given an observation in an LPNO, there may be more than one marking consistent with the observation. First, we define the *confusable relation* between two markings.

Definition 3.13. Given an LPNO G_O, a marking M is said to be *confusable* with M', denoted by $M \sim M'$, if there exists an observation $s \in \mathcal{L}(N, M_0)$ s.t. $M, M' \in \mathcal{C}(s)$. $\diamond$

One can readily verify that $M \sim M'$ is a symmetric, reflexive but not transitive relation. An intuitive way to compute the confusion relation among all markings is to construct an *observer* (see Section 2.1.2). First, since the net is bounded, its reachability graph (RG) can be constructed. This is a graph where each node is a marking M and each arc corresponds to a transition t. We tag each arc t exiting node M with the label $(\ell(t), \Delta f(M, t))$, thus constructing an NFA. Then, the corresponding observer, i.e., the equivalent DFA, can be constructed. Each state of the DFA corresponds to a set $\mathcal{C}(s)$ and all markings in $\mathcal{C}(s)$ are confusable with each other.

Example 3.10. Let us consider the LPNO G_O in Fig. 3.12, where $M_0 = [3\ 0]^T$ and the output function is

$$f(M) = \begin{cases} 1, & \text{if } M(p_2) \text{ is an even number;} \\ -1, & \text{otherwise.} \end{cases}$$

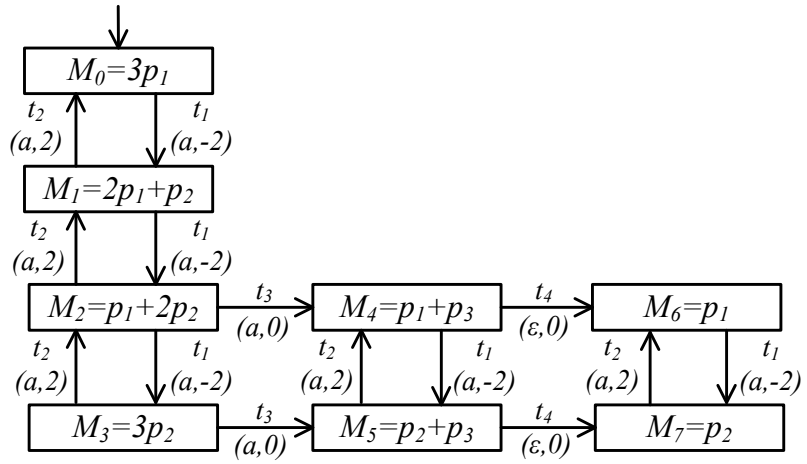

Fig. 3.13 RG of the LPNO in Fig. 3.12.

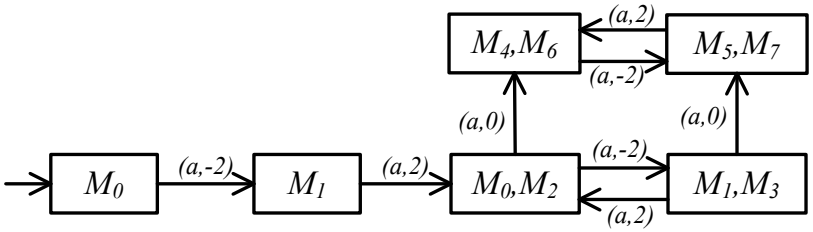

Fig. 3.14 Observer of the RG in Fig. 3.13.

Its RG[2] and the observer are shown in Figs. 3.13 and 3.14, repsectively. Hence, the confusion relations between reachable markings are: $M_0 \sim M_2$, $M_1 \sim M_3$, $M_4 \sim M_6$ and $M_5 \sim M_7$. ◇

It is known that the worst-case complexity of computing a DFA equivalent to an NFA is exponential with respect to the number of states of the NFA. Therefore, the complexity to determine the confusion relation is exponential with respect to the number of markings.

Remark: there may exist more efficient ways to determine the confusion relation. Such a case is discussed in Section 3.6.

3.4.3 Construction of the Agreement and Conflict Graph

If two transitions t and t' of an LPNO may fire at two confusable markings M and M', respectively, and produce the same non-null observation $(e, \Delta f)$, then the two labels $\ell_A(M,t)$ and $\ell_A(M',t')$ must coincide in the equivalent ALPN. Furthermore, any transition t that may fire at a marking M producing the null observation $(\varepsilon, 0)$ should receive a label $\ell_A(M,t) = \varepsilon$ in the equivalent ALPN. These two types of constraints can be captured by a

[2]For clarity, the corresponding transition is also labeled on the arcs.

graph whose nodes are marking-transition pairs $[M, t]$ and whose edges connect nodes that should have the same label in the equivalent ALPN.

Definition 3.14. Given an LPNO G_O, its *agreement graph* is an undirected graph $\mathcal{G}_A = (V, E)$ whose set of vertexes is $V = \{[M, t] \in R(N, M_0) \times T | M[t\rangle\}$ and whose set of edges is $E = E' \cup E''$ where

$$\begin{aligned} E' = &\{([M, t], [M', t']) \in V \times V | [M, t] \neq [M', t'], \\ &(\ell(t), \Delta f(M, t)) = (\ell(t'), \Delta f(M', t')) = (\varepsilon, 0)\} \end{aligned}$$

and

$$\begin{aligned} E'' = &\{([M, t], [M', t']) \in V \times V | \\ &[M, t] \neq [M', t'], M \sim M', \\ &(\ell(t), \Delta f(M, t)) = (\ell(t'), \Delta f(M', t')) \neq (\varepsilon, 0)\}. \end{aligned}$$

$\diamond$

In an agreement graph there are two types of arcs E' and E''. Arcs in E' connect all pairs $[M, t]$ that produce the null observation; arcs in E'' connect pairs $[M, t]$ where markings are confusable and the firings of transitions produce the same non-null observation. Note that there is no self-loop in an agreement graph. After the confusion relation has been determined, the complexity of constructing the agreement graph is $\mathcal{O}(|V|^2)$, since in the worst case, computing the set of edges requires checking $|V|^2$ pairs of nodes $[M, t]$ and $[M't']$.

Example 3.11. Consider Example 3.10 again. In order to clearly illustrate all possible observations, Table 3.3 is built. Based on the confusion relations obtained in Example 3.10 and Table 3.3, the agreement graph in Fig. 3.15 is constructed. To give an example of its construction consider M_0 and M_2. From Example 3.10, M_0 and M_2 are confusable. From Table 3.3, $[M_0, t_1], [M_2, t_1]$ and $[M_2, t_2]$ produce the same observation. Therefore, by Definition 3.14, these three nodes in the agreement graph are connected by arcs in E''. Markings M_4 and M_5 are not confusable, however, nodes $[M_4, t_4]$ and $[M_5, t_4]$ are connected by arcs in E' since they produce the null observation $(\varepsilon, 0)$. $\diamond$

We now consider the connected components of the agreement graph and partition its set of nodes as

$$V = \hat{v}_0 \dot{\cup} \hat{v}_1 \dot{\cup} \hat{v}_2 \dot{\cup} \cdots \dot{\cup} \hat{v}_l$$

Table 3.3 All possible observations at each marking

(e, Δ)	$\{[M,t] \mid \ell(t) = e, \Delta f(M,t) = \Delta\}$
$(a, -2)$	$[M_0, t_1], [M_2, t_1], [M_2, t_2], [M_4, t_1], [M_6, t_1]$
$(a, 2)$	$[M_1, t_1], [M_1, t_2], [M_3, t_2], [M_5, t_2], [M_7, t_2]$
$(a, 0)$	$[M_2, t_3], [M_3, t_3]$
$(\varepsilon, 0)$	$[M_4, t_4], [M_5, t_4]$

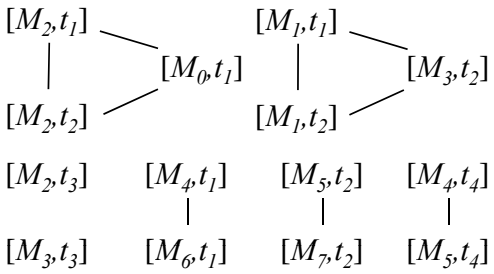

Fig. 3.15 Agreement graph $\mathcal{G}_A$ in Example 3.11.

where for $i \in \{0, 1, 2, \cdots, l\}$, the $\hat{v}_i$-induced subgraph is a component of $\mathcal{G}_A$ and in particular

$$\hat{v}_0 = \{[M,t] \in V \mid \ell(t) = \varepsilon, \Delta f(M,t) = 0\}$$

is the (possibly empty) set of pairs $[M,t]$ that produce the null observation. Correspondingly we define the partition

$$\hat{V} = \{\hat{v}_0, \hat{v}_1, \hat{v}_2, \cdots, \hat{v}_l\}. \tag{3-1}$$

Example 3.12. Consider Example 3.10 again. Based on the agreement graph, we have $\hat{V} = \{\hat{v}_0, \hat{v}_1, \hat{v}_2, \hat{v}_3, \hat{v}_4, \hat{v}_5, \hat{v}_6\}$, where $\hat{v}_0 = \{[M_4, t_4], [M_5, t_4]\}$, $\hat{v}_1 = \{[M_0, t_1], [M_2, t_1], [M_2, t_2]\}$, $\hat{v}_2 = \{[M_1, t_1], [M_1, t_2], [M_3, t_2]\}$, $\hat{v}_3 = \{[M_2, t_3]\}$, $\hat{v}_4 = \{[M_3, t_3]\}$, $\hat{v}_5 = \{[M_4, t_1], [M_6, t_1]\}$, and $\hat{v}_6 = \{[M_5, t_2], [M_7, t_2]\}$. ◇

By means of the agreement graph, we have determined the classes of pairs $[M,t]$ that produce the same observation. We now determine, by means of the conflict graph, which classes must be assigned a different label in the ALPN.

Definition 3.15. Given an LPNO G_O, the *conflict graph* $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$ is an undirected graph whose set of vertexes is $\hat{V}$ as defined in Eq. (3-1) and whose set of edges is $\hat{E} = \hat{E}' \cup \hat{E}''$ where

$$\hat{E}' = \{(\hat{v}_0, \hat{v}_i) \mid \hat{v}_i \in \hat{V}, i \in \{1, 2, \cdots, l\}\}$$

and

$$\begin{aligned}\hat{E}'' =&\{(\hat{v}_i, \hat{v}_j) \in \hat{V} \times \hat{V} | i, j \in \{1, 2, \cdots, l\},\\ &\exists [M, t] \in \hat{v}_i, \exists [M', t'] \in \hat{v}_j :\\ &M \sim M', (\ell(t), \Delta f(M, t)) \neq (\ell(t'), \Delta f(M', t'))\}\end{aligned}$$

◇

Note that $\hat{v}_0$ may not exist, i.e., $\hat{v}_0 = \emptyset$. In this case, $\hat{E}' = \emptyset$ and $\hat{E} = \hat{E}''$. The nodes of graph $\hat{\mathcal{G}}_A$ are classes of nodes $[M, t] \in V$ that produce the same observation. There are also two types of arcs in a conflict graph: $\hat{E}'$ and $\hat{E}''$. Since pairs $[M, t] \in \hat{v}_0$ must be assigned the empty word different from any label from the alphabet, arcs from $\hat{E}'$ connect node $\hat{v}_0$ with every other node; if there exist $[M, t] \in \hat{v}_i$ and $[M', t'] \in \hat{v}_j$ such that M and M' are confusable but t and t' will produce different observations $(e, \Delta f(M, t))$, $(e', \Delta f(M', t'))$, then in the ALPN, different labels must be assigned to them, i.e., $\ell_A(M, t) \neq \ell_A(M', t')$. Thus arcs from $\hat{E}''$ connect such two nodes $\hat{v}_i$ and $\hat{v}_j$.

The complexity of computing the connected components of a graph is known to be linear with respect to the number of edges of a graph using either breadth-first search (BFS) or depth-first search (DFS), i.e., the computation of $\hat{V}$ is $\mathcal{O}(|E|)$. In the worst case, computing the set of edges requires checking $|\hat{V}|^2$ pairs of nodes $\hat{v}_i$ and $\hat{v}_j$. Therefore, based on the agreement graph, the complexity of constructing the conflict graph is $\mathcal{O}(|\hat{V}|^2)$.

3.4.4 Solving the Vertex Coloring Problem

The conflict graph $\hat{\mathcal{G}}_A$ of an LPNO exactly describes the relabeling rule following which an equivalent ALPN can be obtained. We will show that given a bounded LPNO G_O, a vertex coloring of its conflict graph determines an equivalent ALPN and vice versa. Let us first formally define the notion of a vertex coloring.

Definition 3.16. Given a graph $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$, a *vertex coloring* is a pair $(\Sigma_{col}, \ell_{col})$ where Σ_{col} is a finite set of colors and $\ell_{col} : \hat{V} \to \Sigma_{col}$ is a coloring function that assigns to each vertex a color and satisfies the constraint that if $(\hat{v}, \hat{v}') \in \hat{E}$ then $\ell_{col}(\hat{v}) \neq \ell_{col}(\hat{v}')$, i.e., two adjacent vertexes cannot be assigned the same color.

The *vertex coloring problem* is the problem of finding a vertex coloring with a *minimal* number of colors, which is called the *chromatic number* of $\hat{\mathcal{G}}_A$, denoted by $\chi(\hat{\mathcal{G}}_A)$. A graph is called k-*chromatic*, if its chromatic number is k. ◇

Proposition 3.6. Let $G_O = (N, M_0, \Sigma, \ell, f)$ be a bounded LPNO with conflict graph $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$. An ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$ is equivalent to G_O if and only if there exists a vertex coloring $(\Sigma_{col}, \ell_{col})$ of $\hat{\mathcal{G}}_A$ such that $\Sigma_A = \Sigma_{col} \setminus \{\varepsilon\}$ and $[M, t] \in \hat{v}$ with $\hat{v} \in \hat{V} \Rightarrow \ell_A(M, t) = \ell_{col}(\hat{v})$ holds.

Proof: ($\Rightarrow$) To prove the sufficiency of the statement, we show that an ALPN G_A whose adaptive labeling function is defined by a vertex coloring of $\hat{\mathcal{G}}_A$ is equivalent to G_O, namely $\forall \sigma \in L(N, M_0)$, $\mathcal{S}(L_O(\sigma)) = \mathcal{S}(L_A(\sigma))$. This is done by induction on the length of firing sequences.

(Basis step) For any $\sigma \in L(N, M_0)$ of length 0, observations in G_O and G_A are $s = L_O(\sigma) = (\varepsilon, 0)$ and $w_A = L_A(\sigma) = \varepsilon$, respectively. Let $\sigma' = t_1 t_2 \cdots t_k$ with $M_0[t_1\rangle M_1[t_2\rangle M_2 \cdots M_{k-1}[t_k\rangle M_k$.

- First we prove $\mathcal{S}(s) \subseteq \mathcal{S}(w_A)$. Assume $\sigma' \in \mathcal{S}(s)$. It satisfies $\ell(t_i) = \varepsilon$ and $f(M_i) = f(M_0)$, $i = 1, 2, \cdots, k$. According to the definition of $\hat{v}_0$ and the obtained vertex coloring, we have $[M_{i-1}, t_i] \in \hat{v}_0$ and $\ell_{col}(\hat{v}_0) = \varepsilon$, i.e., $\ell_A(M_{i-1}, t_i) = \varepsilon$. Thus, $L_A(\sigma') = \varepsilon$, i.e., $\sigma' \in \mathcal{S}(\varepsilon)$, and $\mathcal{S}(s) \subseteq \mathcal{S}(w_A)$.

- Next we prove $\mathcal{S}(w_A) \subseteq \mathcal{S}(s)$. Let $\sigma' \in \mathcal{S}(w_A)$. Then we have $\ell_A(M_{i-1}, t_i) = \varepsilon$ and $[M_{i-1}, t_i] \in \hat{v}_0$. Therefore, in G_O, $\ell(t_i) = \varepsilon$ and $\Delta f(M_{i-1}, t_i) = 0$ that implies $L_O(\sigma') = (\varepsilon, 0)$, i.e., $\sigma' \in \mathcal{S}(s)$.

As a result, $\mathcal{S}(s) = \mathcal{S}(w_A)$.

(Inductive step) Assume that for any $\sigma \in L(N, M_0)$ of length k, $\mathcal{S}(L_O(\sigma)) = \mathcal{S}(L_A(\sigma))$ holds. In the following, we prove that this is also true for firing sequences of length $k + 1$.

Let $\sigma = \sigma_0 t$ with $M_0[\sigma_0\rangle M_1[t\rangle M_2$, where $|\sigma_0| = k$, $s = L_O(\sigma) = L_O(\sigma_0 t) = s_0(e_1, \Delta)$ and $w_A = L_A(\sigma) = L_A(\sigma_0 t) = w_0 e_2$. In other words, $L_O(\sigma_0) = s_0$, $\ell(t) = e_1$, $\Delta f(M_1, t) = \Delta$, $L_A(\sigma_0) = w_0$ and $\ell_A(M_1, t) = e_2$. Let $\sigma' = \sigma'_0 \sigma'_1$ with $\sigma'_1 = t'_1 t'_2 \cdots t'_k$ and $M_0[\sigma'_0\rangle M'_0[t'_1\rangle M'_1 \cdots M'_{k-1}[t'_k\rangle M'_k$.

- Assume $\sigma' \in \mathcal{S}(s)$ and $\sigma'_0 \in \mathcal{S}(s_0)$. Then there exists $j \in \{1, 2, \cdots, k\}$ such that $\ell(t'_j) = e_1$ and $\Delta f(M'_{j-1}, t'_j) = \Delta$. However, $\forall i \in \{1, 2, \cdots, k\}$ with $i \neq j$, $\ell(t_i) = \varepsilon$ and $\Delta(M'_{i-1}, t'_i) = 0$. According to the definition of $\hat{v}_0$ and the coloring rule, $[M'_{i-1}, t'_i] \in \hat{v}_0$ and in the obtained ALPN $\ell_A(M'_{i-1}, t'_i) = \varepsilon$. Since $\sigma_0, \sigma'_0 t'_1 t'_2 \cdots t'_{j-1} \in \mathcal{S}(s_0)$, M_1 and M'_{j-1} are confusable, i.e., $M_1 \sim M'_{j-1}$. Meanwhile, $(\ell(t), \Delta f(M_1, t)) = (\ell(t'_j), \Delta f(M'_{j-1}, t'_j)) = (e_1, \Delta)$ and hence $[M_1, t]$ and $[M'_{j-1}, t'_j]$ are in a same node of the conflict graph of G_O that indicates in the obtained

ALPN $\ell_A(M'_{j-1}, t'_j) = \ell_A(M_1, t) = e_2$. By induction, $\sigma'_0 \in \mathcal{S}(w_0)$, and therefore, $L_A(\sigma'_0\sigma'_1) = w_0e_2$ and $\sigma' \in \mathcal{S}(w_A)$, i.e., $\mathcal{S}(s) \subseteq \mathcal{S}(w_A)$.

- Analogously, it can be proved $\mathcal{S}(w_A) \subseteq \mathcal{S}(s)$. Assume $\sigma' \in \mathcal{S}(w_A)$ and $\sigma'_0 \in \mathcal{S}(w_0)$. Then there exists $j \in \{1, 2, \cdots, k\}$ such that $\ell_A(M'_{j-1}, t'_j) = e_2$ and $\forall i \in \{1, 2, \cdots, k\}$ with $i \neq j$, $\ell_A(M'_{i-1}, t'_i) = \varepsilon$. Based on the vertex coloring, in the LPNO we have $\ell(t_i) = \varepsilon$ and $\Delta f(M'_{i-1}, t'_i) = 0$. Since $\ell_A(M'_{j-1}, t'_j) = e_2$, by induction $\sigma'_0 \in \mathcal{S}(s_0)$ which means that $\sigma_0, \sigma'_0t'_1t'_2 \cdots t'_{j-1} \in \mathcal{S}(s_0)$ and $M_1 \sim M'_{j-1}$, $[M_1, t]$ and $[M'_{j-1}, t'_j]$ are in a same node of the conflict graph of G_O. Therefore, $\ell(t'_j) = \ell(t) = e_1$, $\Delta f(M'_{j-1}, t'_j) = \Delta f(M_1, t) = \Delta$ and $L_O(\sigma'_0\sigma'_1) = L_O(\sigma') = s_0(e_1, \Delta)$, i.e., $\sigma' \in \mathcal{S}(s)$.

The result follows by induction.

($\Leftarrow$) We prove by contradiction the necessity of the statement. Let $G_A = (N, M_0, \Sigma_A, \ell_A)$ be an ALPN equivalent to G_O. Assume that the adaptive labeling function of G_A is not defined by a vertex coloring to $\hat{\mathcal{G}}_A$, i.e., there exist $[M, t] \in \hat{v}_i$ and $[M', t'] \in \hat{v}_j$ such that $\ell_A(M, t) = \ell_A(M', t')$ and $(\hat{v}_i, \hat{v}_j) \in \hat{E}$. Since $\hat{v}_i$ and $\hat{v}_j$ are adjacent, according to the definition of conflict graphs, there are two possibilities in G_O: i) $M \sim M'$ and $(\ell(t), \Delta f(M, t)) \neq (\ell(t'), \Delta f(M', t'))$; and ii) $(\ell(t), \Delta f(M, t)) = (\varepsilon, 0)$ and $(\ell(t'), \Delta f(M', t')) \neq (\varepsilon, 0)$ (or $(\ell(t'), \Delta f(M', t')) = (\varepsilon, 0)$ and $(\ell(t), \Delta f(M, t)) \neq (\varepsilon, 0)$). For case i), since M and M' are confusable, there exist firing sequences σ and σ' such that $M_0[\sigma\rangle M$, $M_0[\sigma'\rangle M'$ and $L_O(\sigma) = L_O(\sigma') = s$. Therefore, we have $\sigma t \in \mathcal{S}(L_O(\sigma t))$ but $\sigma t \notin \mathcal{S}(L_O(\sigma' t'))$. Assume that the corresponding observation of σ in G_A is w_A. Since G_A is equivalent to G_O, $\sigma t \in \mathcal{S}(L_A(\sigma' t'))$ holds, which implies that, however, $\mathcal{S}(L_A(\sigma' t')) \neq \mathcal{S}(L_O(\sigma' t'))$ and G_A is not equivalent to G_O. Then, we reach a contradiction. Case ii) can be proved analogously. □

Based on the previous results, the ALPN with a minimal alphabet equivalent to a given LPNO can be obtained by solving a vertex coloring problem, i.e, finding a vertex coloring such that the number of colors is minimal. The general procedure to convert a bounded LPNO into an equivalent ALPN with a minimal alphabet is summarized in Algorithm 2.

Since Steps 2 and 3 have polynomial complexity $\mathcal{O}(|V|^2)$ and $\mathcal{O}(|\hat{V}|^2)$, respectively, as we have discussed in the previous sections, the complexity to convert a bounded LPNO into an equivalent ALPN with a minimal alphabet mainly depends on the computation of the confusion relation and on solving the vertex coloring problem, which is known to be in general NP-complete. In the worst case, the RG and the corresponding observer have to

Algorithm 2 Conversion of a bounded LPNO into an equivalent ALPN with a minimal alphabet

Input: a bounded LPNO $G_O = (N, M_0, \Sigma, \ell, f)$
Output: an equivalent ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$
1: Compute the confusion relation.
2: Construct $\mathcal{G}_A$ according to Definition 3.14.
3: Construct $\hat{\mathcal{G}}_A$ according to Definition 3.15.
4: Solve the vertex coloring problem of $\hat{\mathcal{G}}_A$.
5: $\Sigma_A := \Sigma_{col} \setminus \{\varepsilon\}$, $\ell_A := \ell_{col}$.
6: Output G_A.

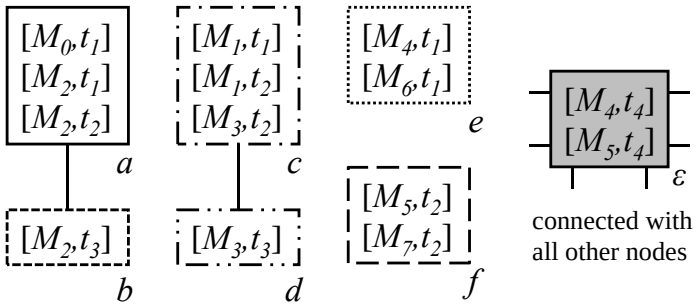

Fig. 3.16 Colored conflict graph $\hat{\mathcal{G}}_A$.

be constructed. Note that in general there is no obvious relation between the size of a net (i.e., the number of places, transitions and tokens that the initial marking assigned to the places) and that of its RG. Therefore, the size of the RG cannot be a priori determined based on the structure of the net. However, in Section 3.6 we show that in some cases without computing the RG the conflict graph can be constructed by just characterizing the output function. Meanwhile, for some special classes of graphs, for example, perfect graphs, the vertex coloring problem can be solved in polynomial time with respect to the number of nodes of the graph (see more results in [56]). Solving the vertex coloring is needed only if one aims to find an equivalent ALPN with a minimal alphabet. On the contrary, the computation of a vertex coloring — not necessarily minimal — is polynomial: one trivial solution is to color every vertex of the conflict graph in different colors and there exist suboptimal solutions with polynomial complexity [56], the greedy algorithm for instance.

Example 3.13. The colored conflict graph of the LPNO in Example 3.10 is shown in Fig. 3.16 (different colors are denoted by different boxes around the nodes), which is a trivial way to color the graph. The equivalent ALPN is $\ell_A(M_0, t_1) = \ell_A(M_2, t_1) = \ell_A(M_2, t_2) = a$, $\ell_A(M_2, t_3) = b$, $\ell_A(M_1, t_1) = \ell_A(M_1, t_2) = \ell_A(M_3, t_2) = c$, $\ell_A(M_3, t_3) = d$, $\ell_A(M_4, t_1) = \ell_A(M_6, t_1) = e$, $\ell_A(M_5, t_2) = \ell(M_7, t_2) = f$ and $\ell_A(M_4, t_4) = \ell_A(M_5, t_4) = \varepsilon$; the alphabet is $\Sigma_A = \{a, b, c, d, e, f\}$.

Nevertheless, the coloring problem of graph $\hat{\mathcal{G}}_A$ can be solved by using three colors. Thus, the equivalent ALPN with a minimal alphabet is $\forall M \in R(N, M_0)$, $\ell_A(M, t_1) = \ell_A(M, t_2) = a$, $\ell_A(M, t_3) = b$ and $\ell_A(M, t_4) = \varepsilon$. This ALPN is also an LPN.

If we apply the brute-force approach, according to Table 3.3, the equivalent ALPN is $\ell_A(M_0, t_1) = \ell_A(M_2, t_1) = \ell_A(M_2, t_2) = \ell_A(M_4, t_1) = \ell_A(M_6, t_1) = [a, -2]$, $\ell_A(M_1, t_1) = \ell_A(M_1, t_2) = \ell_A(M_3, t_2) = \ell_A(M_5, t_2) = \ell_A(M_7, t_2) = [a, 2]$, $\ell_A(M_2, t_3) = \ell_A(M_3, t_3) = [a, 0]$, $\ell_A(M_4, t_4) = \ell_A(M_5, t_4) = \varepsilon$ and the alphabet is $\Sigma_A = \{[a, -2], [a, 2], [a, 0]\}$. $\diamond$

Note that Algorithm 2 is a general procedure that can be applied to any arbitrary bounded LPNO. For some special subclasses, e.g., LPNAFs, the conversion from LPNOs to ALPNs has polynomial complexity (trivially follows from the proof of Proposition 3.1). However, this method cannot assure a minimal alphabet for the LPN. In some cases, even the brute-force approach may provide a fast way to compute the equivalent ALPN, especially for LPNOs with very simple output functions. However, the alphabet of the obtained ALPN is not necessarily minimal and many redundant labels may be introduced. To eliminate redundant labels, further analysis on the confusion relation is required, i.e., the vertex-coloring-based approach is needed.

3.5 Conversion of Bounded LPNOs into LPNs

The results in the previous section show that how any bounded LPNO can be converted into an equivalent ALPN not only with a finite alphabet, but with a minimal alphabet. This however does not ensure the existence of an equivalent LPN.

In this section, for bounded LPNOs, a sufficient and necessary condition for the existence of an equivalent LPN is proposed. If the condition is satisfied, the LPNO can be converted into an equivalent LPN by applying the algorithm presented in this section.

Considering that LPNs are a special case of ALPNs, a necessary condition for the existence of an equivalent LPN is obtained.

Proposition 3.7. Let $G_O = (N, M_0, \Sigma, \ell, f)$ be a bounded LPNO whose conflict graph is k-chromatic. If $|T| < k$, there is no LPN equivalent to G_O.

Proof: Assume that there is an LPN $G_L = (N, M_0, \Sigma, \ell)$ equivalent to G_O. Then the maximal number of labels of G_L is $|T|$, i.e., $|\Sigma| \leq |T|$. Since the conflict graph of G_L is k-chromatic, there is an equivalent ALPN $G_A = (N, M_0, \Sigma_A, \ell_A)$ with $|\Sigma_A| = k$. Considering LPNs are a special class of ALPNs, we have $|\Sigma_A| \leq |\Sigma| \leq |T|$. Then the proposition holds. $\square$

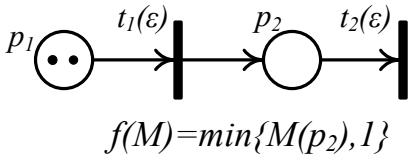

Fig. 3.17 LPNO without equivalent LPN.

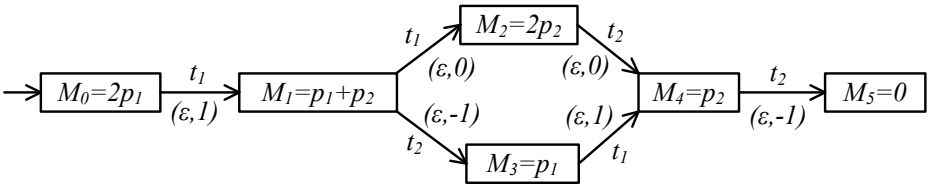

Fig. 3.18 RG of the net in Fig. 3.17.

The following counter example shows that the condition is not sufficient.

Example 3.14. Consider the LPNO in Fig. 3.17 and its corresponding RG in Fig. 3.18. By applying Algorithm 2 and solving the vertex coloring problem, the colored conflict graph is shown in Fig. 3.19. The equivalent ALPN with a minimal alphabet is $\ell_A(M_1, t_1) = \ell_A(M_2, t_2) = \varepsilon$, $\ell_A(M_0, t_1) = \ell_A(M_1, t_2) = \ell_A(M_3, t_1) = \ell_A(M_4, t_2) = \alpha$ and $\Sigma_A = \{\alpha\}$. It satisfies $|\Sigma_A| < |T|$ but there is no LPN equivalent to the LPNO, since all vertex colorings of $\hat{\mathcal{G}}_A$ correspond to ALPNs. ◇

By characterizing the conflict graph, a sufficient and necessary condition that verifies the existence of an equivalent LPN is proposed. First, we introduce some new notations for the conflict graph $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$ of a given LPNO G_O. For a transition $t \in T$, the notation $[\cdot, t]$ denotes a marking-transition pair $[M, t]$ without specifying marking M. The set $T_c(t)$ of a given transition t is defined as

$$T_c(t) = \{t' \in T | \exists \hat{v} \in \hat{V} : [\cdot, t], [\cdot, t'] \in \hat{v})\};$$

If a transition $t' \in T_c(t)$, there exists a node $\hat{v} \in \hat{V}$ to which both $[\cdot, t]$ and $[\cdot, t']$ belong. The set $T_c(t)$ of t is a nonempty set as $t \in T_c(t)$. According to the analysis in the previous

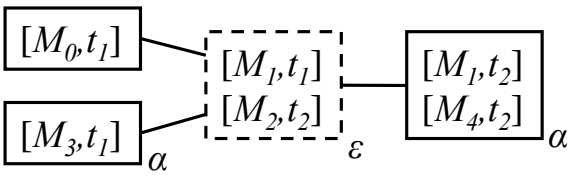

Fig. 3.19 Colored conflict graph $\hat{\mathcal{G}}_A$.

section, in the equivalent ALPN, transitions $t' \in T_c(t)$ will be assigned the same label of transition t at some markings. The set $T_l(t)$ of a given transition t is defined as

$$T_l(t) = \{ \quad t' \in T | \exists \hat{v}_i, \hat{v}_j \in \hat{V} : \\ [\cdot, t] \in \hat{v}_i, [\cdot, t'] \in \hat{v}_j, (\hat{v}_i, \hat{v}_j) \in \hat{E}\}.$$

If $t' \in T_l(t)$, in $\hat{\mathcal{G}}_A$ there are two adjacent nodes $\hat{v}_i$ and $\hat{v}_j$ that contain $[\cdot, t]$ and $[\cdot, t']$, respectively. Therefore, there are markings at which transitions t' and t are assigned different labels in the equivalent ALPN.

Now we discuss the complexity of computing sets $T_c(t)$ and $T_l(t)$ of a given transition t. To compute $T_c(t)$, we first compute the set of nodes $\hat{v}_i \in \hat{V}$ such that $[\cdot, t] \in \hat{v}_i$. The transitions t' of which $[\cdot, t'] \in \hat{v}_i$, belong to $T_c(t)$. Therefore, the complexity of computing $T_c(t)$ is $\mathcal{O}(|\hat{V}|)$. On the other hand, to compute $T_l(t)$, first we select a node $\hat{v}_i \in \hat{V} : \exists [\cdot, t] \in \hat{v}_i$ and then compute a set of nodes $\hat{v}_j \in \hat{V}$ such that there is a edge between $\hat{v}_i$ and $\hat{v}_j$. Finally, the transitions t' of which $[\cdot, t'] \in \hat{v}_j$, belong to $T_l(t)$. Therefore, the complexity of computing $T_l(t)$ is $\mathcal{O}(|\hat{V}|^2)$.

Example 3.15. Consider the conflict graph in Fig. 3.16. We have $T_c(t_1) = T_c(t_2) = \{t_1, t_2\}$, $T_c(t_3) = \{t_3\}$, $T_c(t_4) = \{t_4\}$, $T_l(t_1) = T_l(t_2) = \{t_3, t_4\}$, $T_l(t_3) = \{t_1, t_2, t_4\}$ and $T_l(t_4) = \{t_1, t_2, t_3\}$. ◇

Proposition 3.8. Given an LPNO G_O and its conflict graph $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$, there exists an LPN equivalent to G_O if and only if $T_c(t) \cap T_l(t) = \emptyset$ holds, $\forall t \in T$.

Proof: If the LPNO G_O satisfies $T_c(t) \cap T_l(t) = \emptyset$, then no transition has to be assigned to different labels at different markings. Thus there is a vertex coloring that corresponds to an equivalent LPN. Suppose $T_c(t) \cap T_l(t) \neq \emptyset$. Let $t' \in T_c(t) \cap T_l(t)$. There is a node $\hat{v}_i \in \hat{V}$ that includes $[M_a, t]$ and $[M_b, t']$. Therefore $\ell_A(M_a, t) = \ell_A(M_b, t')$. Since $t' \in T_l(t)$, there are adjacent nodes $\hat{v}_j$ and $\hat{v}_k$ in $\hat{\mathcal{G}}_A$ where $[M_c, t] \in \hat{v}_j$ and $[M_d, t'] \in \hat{v}_k$. We have $\ell_A(M_c, t) \neq \ell_A(M_d, t')$. Hence there exists no vertex coloring corresponding to an LPN and based on Proposition 3.6, there is no equivalent LPN. □

Note that $\forall t' \in T_c(t) \cap T_l(t)$, t' is adaptively labeled in the equivalent ALPN. If an LPNO satisfies Proposition 3.8, there exists a vertex coloring by which the equivalent LPN can be computed. Given a transition $t \in T$, the nodes in $\hat{\mathcal{G}}_A$ containing $[\cdot, t]$ can be merged as one, since $[\cdot, t]$ can be in the same label. To obtain a vertex coloring that corresponds to an LPN, the set of vertexes $\hat{V}$ needs to be reconstructed and Algorithm 3 realizes such a reconstruction.

Algorithm 3 Reconstruction of $\hat{V}$

Input: the set $\hat{V}$ of $\hat{\mathcal{G}}_A$
Output: a new set $\hat{V}_{new}$

```
1: V̂_new := V̂
2: for all v̂_i ∈ V̂_new, do
3:     for all v̂_j ∈ V̂_new \ {v̂_i}, do
4:         if ∃[·, t] ∈ v̂_i : [·, t] ∈ v̂_j, then
5:             v̂_i = v̂_i ∪ v̂_j;
6:             V̂_new = V̂_new \ {v̂_j};
7:         end if
8:     end for
9: end for
10: Output V̂_new.
```

To obtain the final set $\hat{V}_{new}$, first we select a node $\hat{v}_i$ in $\hat{V}$ and find another node $\hat{v}_j \in \hat{V}$ such that $\hat{v}_i$ and $\hat{v}_j$ contain the same transition t. Then we merge $\hat{v}_i$ and $\hat{v}_j$ and remove $\hat{v}_j$ from $\hat{V}$. Note that the obtained node $\hat{v}_i$ will not be treated as a new node. Therefore, the complexity of Algorithm 3 is $\mathcal{O}(|\hat{V}|^2)$. As soon as the set $\hat{V}$ is rebuilt as $\hat{V}_{new}$, the conflict graph $\hat{\mathcal{G}}_A$ should also be reconstructed by Definition 3.15 (in order to avoid confusion, the reconstructed conflict graph is denoted as $\hat{\mathcal{G}}_{Anew}$). Then, by computing a vertex coloring of $\hat{\mathcal{G}}_{Anew}$ the equivalent LPN is obtained. In conclusion, the procedure of finding an equivalent LPN of a bounded LPNO is stated as follows:

Step 1 Construct the conflict graph $\hat{\mathcal{G}}_A$.

Step 2 Check if Proposition 3.8 is verified:

"Yes" — go to Step 3;

"No" — stop, as there is no equivalent LPN.

Step 3 Apply Algorithm 3.

Step 4 Construct the new conflict graph $\hat{\mathcal{G}}_{Anew}$ by Definition 3.15.

Step 5 Compute a vertex coloring of $\hat{\mathcal{G}}_{Anew}$.

Example 3.16. Example 3.15 shows that $\forall t \in T, T_c(t) \cap T_l(t) = \emptyset$, i.e., the LPNO in Fig. 3.12 satisfies Proposition 3.8 and thus there is an LPN equivalent to the LPNO. The conflict graph is reconstructed by applying Algorithm 3 and the colored one is shown in Fig. 3.20. Therefore, the equivalent LPN is $\ell(t_1) = \ell(t_2) = a$, $\ell(t_3) = b$, $\ell(t_4) = \varepsilon$ and $\Sigma = \{a, b\}$.

Consider the LPNO in Example 3.14. According to the conflict graph in Fig. 3.19, there is no LPN equivalent to it since $\forall t \in T, T_c(t) \cap T_l(t) = T$. Results in Example 3.14 also verify this. ◇

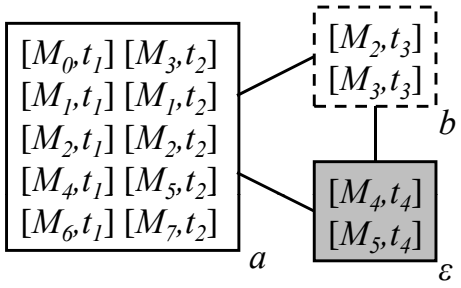

Fig. 3.20 Colored conflict graph $\hat{\mathcal{G}}_{Anew}$.

3.5.1 Further Discussion on the Number of Labels

It is known that the number of colors that can be used to color a graph is not unique, as well as the way of coloring it. If the conflict graph $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$ of an LPNO G_O is k-chromatic, and $|\hat{V}| = \lambda$, the bound of labels of the equivalent ALPN is $k \leq |\Sigma_A| \leq \lambda$. Then, it is important to answer the question whether the lower bound of labels necessarily increases/decreases when an equivalent LPN is required.

Proposition 3.9. Given an LPNO satisfying Proposition 3.8, the minimal number of labels in equivalent LPNs is k if and only if the conflict graph $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$ of G_O is k-chromatic.

Proof: Since the LPNO satisfies Proposition 3.8, there is an equivalent LPN and its conflict graph $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$ can be reconstructed into $\hat{\mathcal{G}}_{Anew}$ by applying Algorithm 3 and Definition 3.15. The reconstruction of $\hat{V}$ does not change the coloring relation between $[M, t]$ pairs. That is to say, even though some $[M, t]$ pairs that are not necessarily in the same node in $\hat{\mathcal{G}}_A$ are absorbed into the same node of $\hat{\mathcal{G}}_{Anew}$, this does not violate the coloring rule since the nodes that belong to $\hat{\mathcal{G}}_A$ are not connected. Therefore, if $\hat{\mathcal{G}}_A$ is k-chromatic, so is $\hat{\mathcal{G}}_{Anew}$, i.e., the minimal number of labels of equivalent LPNs is k.

Proposition 3.6 shows that the vertex colorings of the conflict graph characterize all equivalent ALPNs. Since LPNs are a special class of ALPNs, if the minimal number of labels of equivalent LPNs is k, then $\hat{\mathcal{G}}_A = (\hat{V}, \hat{E})$ of G_O is k-chromatic. □

Therefore, the bound of labels of the equivalent LPN is $k \leq |\Sigma| \leq |T|$. The requirement of equivalent LPNs does not change the minimal number of labels. Proposition 3.9 also implies that if there is no vertex coloring with the chromatic number of labels corresponding to an LPN, then there is no LPN equivalent to the LPNO.

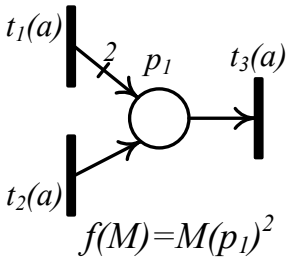

Fig. 3.21 Unbounded LPNO.

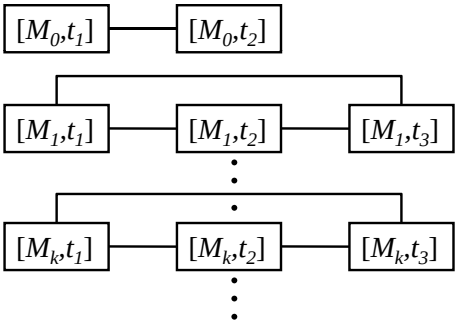

Fig. 3.22 Conflict Graph $\hat{\mathcal{G}}_A$.

3.6 Conversion of Unbounded LPNOs

The conversion algorithms and propositions proposed in the previous sections are applicable to bounded LPNOs. For unbounded LPNOs, the conflict graph may not be feasible to be constructed and analyzed because of the infinite number of markings. Although we lack general results, we give an example to show that in some cases it is possible to convert an unbounded LPNO into an equivalent ALPN by using the same technique.

Example 3.17. Consider the LPNO in Fig. 3.6. Since at each marking, the firings of t_1, t_2 and t_3 produce different observations, no marking is confusable with others. The conflict graph $\hat{\mathcal{G}}_A$ is shown in Fig. 3.6, where $M_i = [i], i = 0, 1, 2, \cdots$. For any $t \in T$, $T_c(t) = \{t\}$ and $T_l(t) = T \setminus \{t\}$. Therefore, the LPNO satisfies Proposition 3.8. By applying Algorithms 2 and 3, the colored conflict graph $\hat{\mathcal{G}}_{Anew}$ is shown in Fig. 3.23 and the equivalent LPN is $\ell(t_1) = a_1, \ell(t_2) = a_2, \ell(t_3) = a_3$ and $\Sigma = \{a_1, a_2, a_3\}$. If we apply the brute force approach to obtain the equivalent ALPN, we need an infinite number of labels. ◇

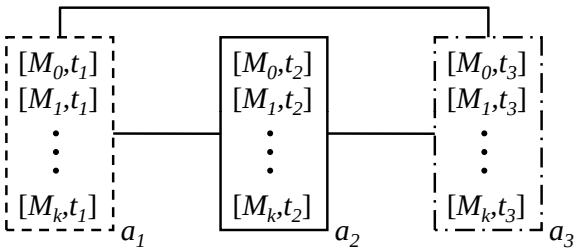

Fig. 3.23 Colored conflict graph $\hat{\mathcal{G}}_{Anew}$.

Example 3.17 shows that even though the conflict graph is infinite, the alphabet of the equivalent ALPN could be finite. This result can be explained by the following theorem concerning the coloring problem in infinite graphs.

Theorem 3.1. [De Bruijin-Erdős theorem (1951)] If and only if all finite subgraphs of an infinite graph $\hat{\mathcal{G}}_A$ can be colored by ρ colors, then $\chi(\hat{\mathcal{G}}_A) \leq \rho$.

3.7 Conclusions

In this chapter different observation structures for Petri net generators are developed. In particular two classes of Petri net generators are defined: labeled Petri nets with outputs (LPNOs) and adaptive labeled Petri nets (ALPNs). The two classes are proper generalizations of labeled Petri nets (LPNs) usually considered in the literature. The notion of observation equivalence is formulated and used to compare the modeling power of different classes of Petri net generators. It is shown that LPNOs and ALPNs have the highest modeling power. Algorithms converting bounded LPNOs to equivalent ALPNs and LPNs with a minimal alphabet are proposed, whose complexity mainly depends on the computation of confusion relations and solving the vertex coloring problem of a particular graph that is called a conflict graph. In the case of unbounded LPNOs, the algorithms may also be applicable.

We believe that LPNOs provide an intuitive way to model systems with various kinds of sensors. However, it may be difficult to analyze the system behavior according to the information provided by the labeling function and output functions in a systematic way. This chapter addressing the conversion from LPNOs to equivalent LPNs provides some useful tools to analyze LPNOs.

The work of this chapter has been published as:

Y. Tong, Z. W. Li, A. Giua. "General Observation Structures for Petri Nets", In *Proceedings of the 18th IEEE International Conference on Emerging Technologies and Factory Automation* (ETFA'13), 2013.

Y. Tong, Z. W. Li, A. Giua. "Observation Equivalence of Petri Net Generators", In *Proceedings of the 12th International Workshop on Discrete Event Systems* (WODES'14), 2014: 338 - 343.

Y. Tong, Z. W. Li, and A. Giua, "On the Equivalence of Observation Structures for Petri Net Generators", *IEEE Transactions on Automatic Control*, Vol. 61, No. 9, pp. 2448 - 2462, **2016**.

Chapter 4 Notions of Opacity in Discrete Event Systems

4.1 Introduction

In this chapter we first recall the formal definitions of current-state opacity, initial-state opacity, and language opacity in the framework of automata [19, 20, 22]. Then we redefine and extend them to the Petri net model: labeled Petri nets. Note that LPNOs and ALPNs are not considered hereafter since the novelty of the proposed approaches would be more easily recognized in the standard Petri net model.

We show that initial-state opacity in Petri nets defined by Bryans *et al.* [5] is not a counterpart of initial-state opacity in automata, and thus we call the former one B-initial-state opacity and redefine initial-state opacity in Petri nets. Furthermore, in Petri nets we define a new notion of language-based opacity: *strict language opacity*, to model the case where the intruder only cares the order of some transitions in the secret words. Finally, we investigate the relation between language opacity and strict language opacity.

4.2 Opacity Properties in Automaton Formulation

In the automaton framework, it is assumed that the intruder knows the structure of the automaton $\mathcal{A} = (X, E, \delta, x_0)$ but partially observes the system. we use $E_I \subseteq E$ to denote the set of events observable by the intruder and $P_{E_I} : E^* \to E_I^*$ to denote the natural projection from E to E_I. Note that for initial-state opacity, the initial state x_0 is not specified.

4.2.1 Current-State Opacity

Definition 4.1. Given a DFA $\mathcal{A} = (X, E, \delta, x_0)$, a secret $S \subseteq X$, and a set E_I of events observable by the intruder, the system is said to be *current-state opaque* (CSO) wrt S and E_I if $\forall \sigma \in L(\mathcal{A})$ such that $\delta(x_0, \sigma) \in S$,

$$\exists \sigma' \in L(\mathcal{A}) : P_{E_I}(\sigma') = P_{E_I}(\sigma) \text{ and } \delta(x_0, \sigma') \notin S. \qquad \diamond$$

A system is said to be CSO if for all possible observations, there exists a sequence of events that produces the observation but reaches a state that is not in the secret. Therefore, the intruder cannot establish if the current state belongs to the secret.

Example 4.1. Consider the DFA in Fig. 4.1 and let $S = \{3, 5\}$, $E_I = \{b, c\}$. The system is CSO wrt S and E_I because when nothing is observed by the intruder, the current state could

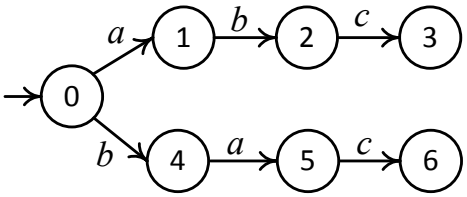

Fig. 4.1 DFA that is CSO wrt $\{3, 5\}$ and $\{b, c\}$.

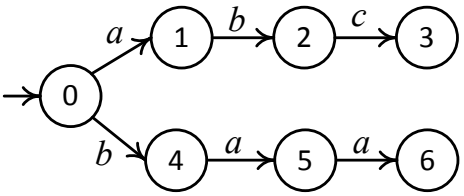

Fig. 4.2 DFA that is not CSO wrt $\{3, 5\}$ and $\{b, c\}$.

be 0 or 1; with observing b, the intruder would conclude that the current state could be 2, or 4 or 5; finally with observing bc, the current state could be 3 or 6. Therefore, no matter what the intruder observes it is not able to conclude that the current state of the system is in S, i.e., the system is not CSO wrt S and E_I.

Now consider the DFA in Fig. 4.2. The system is not CSO wrt S and E_I because after the intruder observes bc, it knows the current state of the system is 3 and it is in the secret. $\diamond$

4.2.2 Initial-State Opacity

Definition 4.2. Given a DFA $\mathcal{A} = (X, E, \delta)$, a secret $S \subseteq X$, and a set E_I of events observable by the intruder, the system is said to be *initial-state opaque* (ISO) wrt S and E_I if $\forall \sigma \in L(\mathcal{A}, S)$,

$$\exists \sigma' \in L(\mathcal{A}, X \setminus S) : P_{E_I}(\sigma') = P_{E_I}(\sigma). \qquad \diamond$$

In words, a system is said to be ISO if for all observations generated from a secret state, there exists at least a nonsecret state from which the same observation can be generated. Therefore, given an observation the intruder cannot establish if the observation is generated from a secret state.

Example 4.2. Consider the DFA in Fig. 4.1 and $E_I = \{b, c\}$ and $S = \{0, 2\}$. Clearly, the system is ISO wrt S and E_I since whatever the intruder observes, it does not know the observation is generated from a secret state.

Let us consider the DFA in Fig. 4.2. However, the system is not ISO wrt S and E_I, since when the intruder observes c, it knows that c is generated from 2. $\diamond$

4.2.3 Language Opacity

Current-state opacity and initial-state opacity belong to state-based opacity properties as the secret is defined as a set of states. On the other hand, if the secret is defined as a language then such an opacity property is called *language-based opacity*. There are many language-based opacity properties have been defined in literature such as strong language

opacity and weak language opacity [6]. In the thesis, we study the very basic one: language opacity[1].

Definition 4.3. Given a DFA $\mathcal{A} = (X, E, \delta, x_0)$, a secret $S \subseteq L(\mathcal{A})$, and a set E_I of events observable by the intruder, the system is said to be *language opaque* (LO) wrt S and E_I if $\forall \sigma \in S$,

$$\exists \sigma' \in L(\mathcal{A}) \setminus S : P_{E_I}(\sigma') = P_{E_I}(\sigma). \qquad \diamond$$

A system is language opaque wrt the secret if for all secret strings there exists a non-secret sting that produces the same observation. Therefore, the intruder cannot establish if some secret sequences have occurred based on its observation.

Example 4.3. Consider the DFA in Fig. 4.1 and let $S = \{abc\}$, $E_I = \{b, c\}$. Clearly, the system is LO wrt S and E_I since for $\sigma = abc$, there exists $\sigma' = bac$ such that $P_{E_I}(\sigma) = P_{E_I}(\sigma') = bc$.

Now consider the DFA in Fig. 4.2. The system is not LO wrt S and E_I because when the intruder observes bc, it knows abc has occurred. $\diamond$

4.3 Opacity Properties in Petri Net Formulation

In this section we define opacity properties in Petri nets and study their relations. It is assumed that an intruder knows the structure and the initial marking of the system, however, it can only partially observe the occurrence of events of the system. To present such an observation structure, the system under consideration is modeled by a labeled Petri net system, where the observation function is static and the states are not observable.

4.3.1 Current-State Opacity

Definition 4.4. An LPN system $G = (N, M_0, \Sigma, \ell)$ is said to be *current-state opaque* (CSO) wrt a secret $S \subseteq R(N, M_0)$ if $\forall \sigma \in L(N, M_0)$ such that $M_0[\sigma\rangle M \in S$,

$$\exists \sigma' \in L(N, M_0) : M_0[\sigma'\rangle M' \notin S \text{ and } \ell(\sigma) = \ell(\sigma') \qquad \diamond$$

We can see Definition 4.4 and Definition 4.1 have the same meaning but in different form. An LPN system is CSO means for all transition sequences leading to a secret marking there exists a transition sequence producing the same observation but leading to a nonsecret marking.

[1]This notion of opacity was defined in [22] but was called language-based opacity. To avoid confusing, we call it language opacity in the thesis.

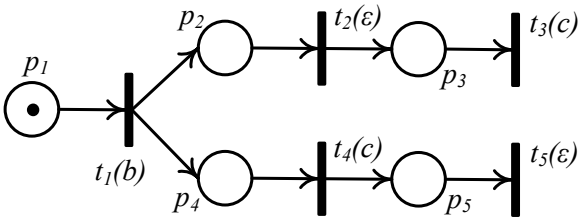

Fig. 4.3 LPN system that is CSO wrt $\{p_2 + p_4, p_2 + p_5\}$.

Example 4.4. Consider the LPN system in Fig. 4.3 and let $S = \{p_2+p_4, p_2+p_5\}$. For $\sigma = t_1$ with $M_0[\sigma\rangle p_2+p_4$, there exists $\sigma' = t_1t_2$ such that $M_0[\sigma'\rangle p_3+p_4 \notin S$ and $\ell(\sigma) = \ell(\sigma') = b$; for $\sigma = t_1t_4$ with $M_0[\sigma\rangle p_2 + p_5$, there exists $\sigma' = t_1t_4t_5$ such that $M_0[\sigma'\rangle p_2 \notin S$ and $\ell(\sigma) = \ell(\sigma') = bc$. Therefore, the system is CSO wrt S.

However, the system is not CSO wrt $S = \{p_2+p_4, p_3+p_4\}$ since if the intruder observes b, it knows that the current-state of the system could only be $p_2 + p_4$ or $p_3 + p_4$. ◇

4.3.2 Initial-State Opacity

The notion of initial-state opacity was first defined for Petri nets by Bryans *et al.* in [5]. According to the definition given by Bryans *et al.*, when the intruder starts its observation it does not know in which marking the system is, but simply knows that it belongs to a given set $\mathcal{M}_0 \in \mathbb{N}^m$. The secret set S is a subset of $\mathcal{M}_0$. If the system is initial-state opaque, then the intruder cannot infer, based on its observation, whether the evolution has started from a secret marking or a nonsecret one. In the thesis, we called it *B-initial-state opacity* to distinguish it from the new notion of initial-state opacity proposed in the thesis.

Definition 4.5. Let $G = (N, \mathcal{M}_0, \Sigma, \ell)$ be an LPN system and $S \subseteq \mathcal{M}_0$ be a secret set. G is said to be *B-initial-state opaque* (B-ISO) wrt S if for all $w \in \mathcal{L}(G, S)$,

$$\exists M' \in \mathcal{M}_0 \setminus S : w \in \mathcal{L}(G, M'). \qquad \diamond$$

A system is said to be B-ISO if for any observation generated from a secret marking, there always exists a nonsecret marking in $\mathcal{M}_0$ from which the same observation can be generated.

Example 4.5. Consider the LPN in Fig. 4.3 and $\mathcal{M}_0 = \{2p_4, p_2 + p_3\}$. Let $S = \{p_2 + p_3\}$. Thus $\mathcal{L}(G, S) = \{c, cc\}$. The LPN system is B-ISO wrt S since for any $w \in \mathcal{L}(G, S)$, there exists $2p_4 \notin S$ such that $w \in \mathcal{L}(G, 2p_4)$. ◇

Initial-state opacity studied in the thesis is defined as follows.

Definition 4.6. Let $G = (N, M_{st}, \Sigma, \ell)$ be an LPN system and $S \subseteq R(N, M_{st})$ be a secret set. G is said to be *initial-state opaque* (ISO) wrt S if for all $w \in \mathcal{L}(G, S)$,

$$\exists M' \in R(N, M_{st}) \setminus S : w \in \mathcal{L}(G, M'). \qquad \diamond$$

According to Definition 4.6, a system $G = (N, M_{st}, \Sigma, \ell)$ is said to be ISO if for any secret marking and any observation generated from such a secret marking, there exists a nonsecret marking in $R(N, M_{st})$ from which the same observation can be generated. Comparing Definition 4.6 with Definition 4.5, it is evident that ISO is a special case of B-ISO where $\mathcal{M}_0$ is a subset of $\mathbb{N}^m$ such that $\mathcal{M}_0 = R(N, M_{st})$. Namely, ISO only considers a special class of $\mathcal{M}_0$ instead of an arbitrary subset of $\mathbb{N}^m$. However, as discussed in the following, it is still worth studying them separately.

The ISO problem considered in the automaton setting can be summarized as follows: given the structure of an automaton whose initial state is unknown, determine if the intruder can infer the state belongs to the secret by observing the system's evolution. Note that the structure of the automaton explicitly contains the full knowledge of the system's state space and thus it is implicitly assumed that the initial state must belong to such a space.

Consider on the contrary a labeled Petri net. The structure of the net (N, Σ, ℓ) is not a dynamical system and contains no information on the state space. We need to associate to the net an initial marking so that the state space can be determined by computing its reachability set. Therefore, the counterpart for Petri nets of the ISO problem defined above can be stated as follows: given a Petri net *system* and its reachability set, assuming its initial marking is unknown, determine if the intruder can infer whether such a marking belongs to the secret by observing the system's evolution.

Therefore, the new definition we propose of ISO not only formalizes an important property that so far has not been discussed, but also clarifies the difference between B-ISO for Petri nets and ISO for automata.

Example 4.6. Consider the LPN system in Fig. 4.3 and let $S = \{p_2 + p_4, p_3 + p_5\}$. We have $\mathcal{L}(G, S) = \{c, cc\}$. Observation $w = c$ may also be generated from $p_2 + p_5$ or $p_3 + p_4$, and observation cc may also be generated from $p_3 + p_4$. Thus the system is ISO wrt S. Let $\mathcal{M}_0 = R(N, M_0)$. Clearly, $G = (N, \mathcal{M}_0, \Sigma, \ell)$ is B-ISO wrt S.

However, the system is not ISO wrt $S = \{M_0\}$ since when the intruder observes b, or bc, or bcc, it knows the observation is generated from M_0. $\diamond$

4.3.3 Language-Based Opacity

When the secret is defined as a language, an opacity property is categorized as language-based opacity. In Petri nets, to be more specific, we define two language-based opacity properties: *language opacity* and *strict language opacity*.

Definition 4.7. An LPN system $G = (N, M_0, \Sigma, \ell)$ is said to be *language opaque* (LO) wrt $S \subseteq L(N, M_0)$ if $\forall \sigma \in S$,

$$\exists \sigma' \in L(N, M_0) \setminus S : \ell(\sigma) = \ell(\sigma'). \qquad \diamond$$

In words, a system is language opaque if for all transition sequence in the secret there is another transition sequence that is not in the secret and produces the same observation. Therefore, no matter what the intruder observes, it does not know whether such an observation is produced by a secret sequence or not.

Example 4.7. Consider the LPN system in Fig. 4.3 and let $S = \{t_1t_2t_3\}$. Therefore, the observation produced by the secret is $\ell(t_1t_2t_3) = bc$. Meanwhile, there is $t_1t_4t_5 \notin S$ and $\ell(t_1t_4t_5) = bc$. Thus, the system is language opaque wrt S. However, the system is not language opaque wrt $S = \{t_1, t_1t_2\}$ since when the intruder observes b it knows that sequence in S has occurred. $\diamond$

Language opacity defined above is a counterpart of language opacity in automata. From its definition, we can see in language opacity a transition sequence is secret if and only if it belongs to S, i.e., the order of transitions is same as one element in S. However, there exist many practical cases where the intruder only cares about a subset of transitions. As an example, in a banking environment, the intruder may not care if a customer checks the account but may only be interested in the withdrawal of money. Motivated by this, we propose a generalization of the language opacity property and introduce the notion of *strict language opacity*. In particular, a system is strictly language opaque if the intruder can never establish if the transitions in which it is interested have fired in some given order (as described by the secret).

Definition 4.8. Given an LPN system $G = (N, M_0, \Sigma, \ell)$, a set of transitions $\hat{T} \subseteq T$, and a secret $\hat{S} \subseteq \hat{T}^*$. G is said to be *strictly language opaque* wrt to $\hat{T}$ and $\hat{S}$ if $\forall \sigma \in L(N, M_0)$ such that $P_{\hat{T}}(\sigma) \in \hat{S}$, there exists $\sigma' \in L(N, M_0)$ such that

$$\ell(\sigma') = \ell(\sigma) \text{ and } P_{\hat{T}}(\sigma') \notin \hat{S},$$

where $P_{\hat{T}} : T^* \to \hat{T}^*$ is the natural projection from T to $\hat{T}$. $\diamond$

In words, a system is strictly language opaque wrt $\hat{T}$ and $\hat{S}$ if for any observation that can be explained with a sequence whose projection on $\hat{T}$ belongs to the secret, there exists another explanation whose projection on $\hat{T}$ does not belong to the secret. Note that for strict language opacity, the secret $\hat{S}$ may not be a subset of $L(N, M_0)$. Obviously, if $\hat{T} = T$, strict language opacity is identical to language opacity. Next we further discuss some properties of strict language opacity.

Proposition 4.1. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system, $\hat{T}' \subseteq \hat{T} \subseteq T$, and $\hat{S} \subseteq \hat{T}^*$ a secret. If G is strictly language opaque wrt $\hat{T}'$ and $\hat{S}'$, where $\hat{S}' = P_{\hat{T}'}(\hat{S})$, then G is strictly language opaque wrt $\hat{T}$ and $\hat{S}$.

Proof: This is proved by showing that if G is not strictly language opaque wrt $\hat{T}$ and $\hat{S}$ then G is not strictly language opaque wrt $\hat{T}'$ and $\hat{S}'$. Given a transition sequence $\sigma \in L(N, M_0)$, the set of transition sequences having the same observation with σ is denoted as $\Sigma = \{\sigma' \in L(N, M_0) | \ell(\sigma') = \ell(\sigma)\}$. Since G is not strictly language opaque wrt $\hat{T}$ and $\hat{S}$, there exists $\sigma \in L(N, M_0)$ such that $P_{\hat{T}}(\sigma) \in \hat{S}$ and $\forall \sigma' \in \Sigma$, $P_{\hat{T}}(\sigma') \in \hat{S}$ holds. While projecting σ and σ' on $\hat{T}'$, it holds $P_{\hat{T}'}(P_{\hat{T}}(\sigma)) \in \hat{S}'$ and $P_{\hat{T}'}(P_{\hat{T}}(\sigma')) \in \hat{S}'$. Since $\hat{T}' \subseteq \hat{T}$, $P_{\hat{T}'}(P_{\hat{T}}(\sigma)) = P_{\hat{T}'}(\sigma)$ and $P_{\hat{T}'}(P_{\hat{T}}(\sigma')) = P_{\hat{T}'}(\sigma')$ hold. Thus, for $\sigma \in L(N, M_0)$ such that $P_{\hat{T}'}(\sigma) \in \hat{S}'$, $P_{\hat{T}'}(\sigma') \in \hat{S}'$ holds for all $\sigma' \in \Sigma$, i.e., G is not strictly language opaque wrt $\hat{T}'$ and $\hat{S}'$. □

Proposition 4.1 provides a semi-decision procedure (only sufficient) to verify strict language opacity wrt $\hat{T}$ and $\hat{S}$. More precisely, one can choose a subset $\hat{T}'$ of $\hat{T}$ and verify strict language opacity wrt $\hat{T}'$ and $\hat{S}'$. Note that as aforementioned, if $\hat{T} = T$, strict language opacity is identical to language opacity. Therefore, language opacity wrt a given secret $S \subseteq T^*$ can be sufficiently decided by verifying the strict language opacity property wrt a set $\hat{T} \subseteq T$ and $P_{\hat{T}}(S)$. In the following example, it is shown that the converse of Proposition 4.1 may not hold. Namely, it may occur that even if a system is strictly language opaque wrt $\hat{T}$ and $\hat{S}$, it may not be strictly language opaque wrt $\hat{T}'$ and $\hat{S}'$.

Example 4.8. Consider the LPN system in Fig. 4.3. Let $\hat{T} = T$ and $\hat{S} = \{t_1 t_2\}$. The system is strictly language opaque wrt $\hat{T}$ and $\hat{S}$ (equivalently, G is language opaque wrt $\hat{S}$), since there exists $\sigma = t_1$ such that $P_{\hat{T}}(\sigma) \notin \hat{S}$ and $\ell(t_1) = \ell(t_1 t_2) = b$. Let $\hat{T}' = \{t_1\} \subset \hat{T}$. The system is not strictly language opaque wrt $\hat{T}'$ and $\hat{S}'$, where $\hat{S}' = P_{\hat{T}'}(\hat{S}) = \{t_1\}$, since there does not exist a sequence generating b but whose projection on $\hat{T}'$ is not in $\hat{S}'$. ◇

In the supervisory control, the *normality* [57] property of a language is introduced. Herein this notion is slightly extended as follows.

Definition 4.9. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system and $\hat{T} \subseteq T$ a subset of transitions. A language $S \subseteq T^*$ is said to be *normal* wrt $L(N, M_0)$ and $P_{\hat{T}}$ if

$$L(N, M_0) \cap S = L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S)). \qquad \diamond$$

A language S is normal wrt a given language $L(N, M_0)$ and the natural projection $P_{\hat{T}}$ if its intersection with $L(N, M_0)$ is the largest sublanguage of $L(N, M_0)$ whose projection is $P_{\hat{T}}(S)$.

Proposition 4.2. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system, $\hat{T} \subseteq T$ a subset of T, $S \subseteq T^*$ a secret, and $\hat{S} = P_{\hat{T}}(S)$. G is strictly language opaque wrt $\hat{T}$ and $\hat{S}$, if and only if G is language opaque wrt $P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$.

Proof: ($\Rightarrow$) Follows from Proposition 4.1.

($\Leftarrow$) Given a transition sequence $\sigma \in L(N, M_0)$, the set of transition sequences having the same observation with σ is denoted as $\Sigma = \{\sigma' \in L(N, M_0) | \ell(\sigma') = \ell(\sigma)\}$. Assume that G is not strictly language opaque wrt $\hat{T}$ and $\hat{S}$, therefore there exists a sequence $\sigma \in L(N, M_0)$ such that $P_{\hat{T}}(\sigma) \in \hat{S}$ and $\forall \sigma' \in \Sigma$, it holds $P_{\hat{T}}(\sigma') \in \hat{S}$. Since $\hat{S} = P_{\hat{T}}(S)$, $\sigma \in L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$ and $\sigma' \in L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$ hold. Thus, G is not language opaque wrt $P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$. $\square$

Proposition 4.2 shows that strict language opacity wrt $\hat{T}$ and $\hat{S}$ is identical to language opacity wrt $P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$.

Corollary 4.1. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system and $\hat{T} \subseteq T$. Let $S \subseteq T^*$ be a secret that is normal wrt $L(N, M_0)$ and $P_{\hat{T}}$ and $\hat{S} = P_{\hat{T}}(S)$. G is strict language opaque wrt $\hat{T}$ and $\hat{S}$ if and only if G is language opaque wrt S.

Proof: According to Definition 4.7, given a secret $S \subseteq T^*$, G is language opaque wrt S is identical to G is language opaque wrt $S \cap L(N, M_0)$. By Proposition 4.2, G is strictly language opaque wrt $\hat{T}$ and $\hat{S}$ if and only if G is language opaque wrt $L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$. Since S is normal, i.e., $L(N, M_0) \cap S = L(N, M_0) \cap P_{\hat{T}}^{-1}(P_{\hat{T}}(S))$, G is strictly language opaque wrt $\hat{T}$ and $\hat{S}$ if and only if G is language opaque wrt S. $\square$

In general, language opacity wrt $S \subseteq T^*$ (that is identical to strict language opacity wrt T and S) does not imply strict language opacity wrt $\hat{T}$ and $P_{\hat{T}}(S)$ if $\hat{T} \subseteq T$ (see Proposition 4.1). However, by Corollary 4.1, such an implication holds if S is normal wrt $L(N, M_0)$ and $P_{\hat{T}}$. In other words, given an arbitrary set $\hat{S} \subseteq \hat{T}^*$, verifying strict language opacity wrt $\hat{S}$ and $\hat{T}$ can be reduced to verifying language opacity wrt $P_{\hat{T}}^{-1}(\hat{S})$. Thus, the complexity of verifying strict language opacity would not be higher than that of verifying language

opacity. Furthermore, in Chapter 6 we show that under proper assumptions, verification of strict language opacity is of lower complexity.

4.4 Conclusion

Definitions of current-state opacity, initial-state opacity, and language opacity were recalled in this chapter. Furthermore, we formalized the aforementioned opacity properties in Petri nets and compared their differences with ones in [5]. Finally, we defined a new notion of language-based opacity in Petri nets, called *strict language opacity* and studied its relation with language opacity.

Partial work of this chapter has been published as:

Y. Tong, Z. Y. Ma, Z. W. Li, C. Seatzu, A. Giua, "Verification of Language-Based Opacity in Petri Nets Using Verifier", the 35th American Control Conference (ACC'16), 2016: 757-763.

Chapter 5 Decidability of Opacity Verification Problems in Petri Nets

5.1 Introduction

In this chapter, we study decidability of opacity verification problems in Petri nets. Opacity verification [6, 21, 22, 28] consists in determining whether a system is opaque with respect to a given secret. In the sequel of this chapter we use "opacity problem" to replace "opacity verification problem" for simplicity.

Note that in this chapter, we consider a more general LPN system that has a set $\mathcal{M}_0 \subseteq \mathbb{N}^m$ (may be infinite) of initial markings. In such a case, the RG of the LPN system $G = (N, \mathcal{M}_0, \Sigma, \ell)$ is $R(N, \mathcal{M}_0) = \bigcup_{M_0 \in \mathcal{M}_0} R(N, M_0)$, and the set of transition sequences enabled from the initial marking is $L(N, \mathcal{M}_0)$. Thus, to be more clear, opacity properties are redefined in the new LPN framework. We also point out that an LPN system with a finite set of initial markings can always be converted into an equivalent LPN system[1] with one initial marking. The procedure requires adding two new places, called p_0 and p_0', and $r = |\mathcal{M}_0|$ new unobservable transitions, called $t_{u1}, \ldots, t_{ur}$. The initial marking of the new net assigns a single token to place p_0. The firing of a transition t_{ui} (with $i = 1, \ldots, r$) moves the token from p_0 to p_0' and produces in the other places a token configuration that coincides with the i-th marking in $\mathcal{M}_0$. To prevent transitions (in particular source transitions) from firing before one of the transitions t_{ui} does, self-loops are added between p_0' and all other transitions except t_{ui} for $i = 1, \ldots, r$ (cf. the proof of Theorem 5.2). Therefore, opacity properties defined in this chapter are equivalent to those defined in Section 4.3.

The rest of the chapter is organized as follows. In Sections 5.2, 5.3, and 5.4, the decidability of the current-state, initial-state and language opacity problems is discussed, respectively. Finally, conclusions are drawn in Section 5.5.

5.2 Decidability of the Current-State Opacity Problem

In this section we discuss the decidability of the current-state opacity problem in LPN systems.

Definition 5.1. Let $G = (N, \mathcal{M}_0, \Sigma, \ell)$ be an LPN system and $S \subseteq R(N, \mathcal{M}_0)$ be a secret set. G is said to be *current-state opaque* (CSO) wrt S if for all $M_0 \in \mathcal{M}_0$, $M \in S$ and

[1] "Equivalent" refers to the fact that two nets have the same opacity property.

$\sigma \in L(N, M_0)$ such that $M_0[\sigma\rangle M$, there exists $M'_0 \in \mathcal{M}_0, \sigma' \in L(N, M'_0)$ such that $\ell(\sigma') = \ell(\sigma)$ and $M'_0[\sigma'\rangle M' \notin S$. $\diamond$

Definition 5.2. Consider an LPN system $G = (N, \mathcal{M}_0, \Sigma, \ell)$ and a secret set $S \subseteq R(N, \mathcal{M}_0)$. The *Petri net current-state opacity problem* consists in determining whether G is current-state opaque wrt S or not. $\diamond$

In [5] it has been proven that if G is bounded, which also implies that $\mathcal{M}_0$ is finite, the Petri net current-state opacity problem is decidable. In the following, we show that in general such a problem is undecidable.

Theorem 5.1. The Petri net current-state opacity problem is undecidable.

Proof: We preliminarily recall that the Petri net language containment problem, i.e., the problem of determining whether the language generated by an LPN system is contained in the language generated by another LPN system, is not decidable [58]. We now prove the theorem by showing that the Petri net language containment problem can be reduced (in polynomial time) to the Petri net current-state opacity problem for a singleton secret set and a single initial marking.

Let $\mathcal{L}(G_1, M_{01})$ and $\mathcal{L}(G_2, M_{02})$ be the languages generated by two arbitrary LPN systems $G_1 = (N_1, M_{01}, \Sigma_1, \ell_1)$ and $G_2 = (N_2, M_{02}, \Sigma_2, \ell_2)$, respectively. Let P_i ($|P_i| = m_i$) and T_i ($|T_i| = n_i$), respectively, be the set of places and transitions of G_i, for $i = 1, 2$. We construct a new LPN system $G = (N, M_0, \Sigma, \ell)$ based on G_1 and G_2 by the following steps:

i) Duplicate the structures of G_1 and G_2 in G.

ii) Add to G places: $\overline{p}_0$, $\overline{p}_1$, $\overline{p}_2$, and $\overline{p}_3$, unobservable transitions: $\bar{t}_1$ and $\bar{t}_2$, and observable transitions: $\bar{t}_3$ and $\bar{t}_4$ such that $\ell(\bar{t}_3) = \ell(\bar{t}_4) = z \notin (\Sigma_1 \cup \Sigma_2)$.

iii) Add new arcs: $Pre(\overline{p}_0, \bar{t}_i) = 1$ for $i = 1, 2$; $Pre(\overline{p}_1, \bar{t}_3) = 1$; $Pre(\overline{p}_2, \bar{t}_4) = 1$; $\forall t \in T_1$, $Pre(\overline{p}_1, t) = 1$, $Post(\overline{p}_1, t) = 1$; $\forall t \in T_2$, $Pre(\overline{p}_2, t) = 1$, $Post(\overline{p}_2, t) = 1$; $Post(\overline{p}_1, \bar{t}_1) = 1$; $Post(\overline{p}_2, \bar{t}_2) = 1$; $Post(\overline{p}_3, \bar{t}_3) = 1$; $\forall p \in P$ such that $M_{01}(p) \neq 0$, $Post(p, \bar{t}_1) = M_{01}(p)$; $\forall p \in P$ such that $M_{02}(p) \neq 0$, $Post(p, \bar{t}_2) = M_{02}(p)$.

iv) $M_0 = \overline{p}_0$.

As a result, the number of places and transitions in G are $|P| = m_1 + m_2 + 4$ and $|T| = n_1 + n_2 + 4$, respectively, and $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \{z\}$. The LPN system G is depicted in Fig. 5.1. For $i = 1, 2$, the firing of $\bar{t}_i$ initializes G_i. Namely, the markings reached after

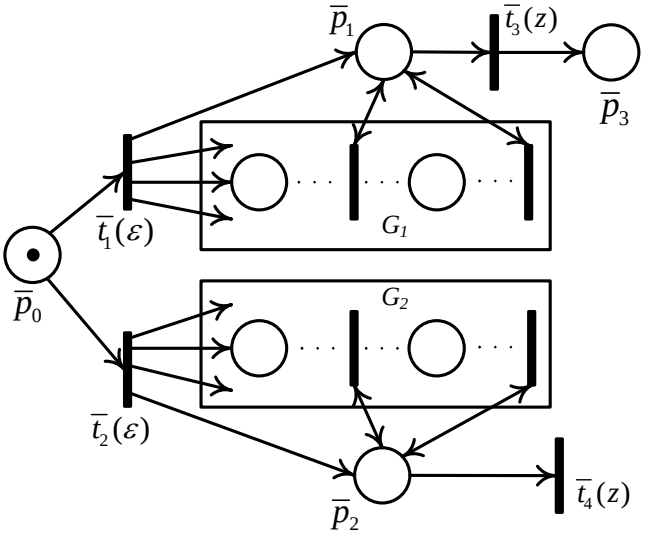

Fig. 5.1 LPN system G constructed in the proof of Theorem 5.1.

firing $\bar{t}_i$ are $M = \overline{p}_i + \Sigma_{p \in P_i} M_{0i}(p) \cdot p$. Self-loops between $\overline{p}_i$ and transitions in G_i prevent source transitions from firing before $\bar{t}_i$ fires.

Let us consider the secret set $S = \{\overline{p}_3\}$. In the following we prove that

$$\mathcal{L}(G_1, M_{01}) \subseteq \mathcal{L}(G_2, M_{02}) \Leftrightarrow G \text{ is current-state opaque wrt } S.$$

We first prove that if G is current-state opaque wrt S, then $\mathcal{L}(G_1, M_{01}) \subseteq \mathcal{L}(G_2, M_{02})$ holds. Assume that G is current-state opaque wrt S. Then for every σ leading to the secret marking, there exists $\sigma' \in L(N, M_0)$ that does not lead to the secret but produces the same observation, i.e., $\ell(\sigma) = \ell(\sigma')$. Based on the structure of G, the transition sequences that lead to the secret marking take the form $\sigma = \bar{t}_1 \sigma_1 \bar{t}_3$, where $\sigma_1 \in L(N_1, M_{01})$ and produce observation $\ell(\sigma) = \ell(\sigma_1)z$, where $\ell(\sigma_1) \in \mathcal{L}(G_1, M_{01})$. Moreover, it appears evident that σ' should take the form $\sigma' = \bar{t}_2 \sigma_2 \bar{t}_4$, where $\sigma_2 \in L(N_2, M_{02})$. Indeed, these are the only sequences that produce an observation ending with z and not leading to the secret marking. This implies that for any $\sigma_1 \in L(N_1, M_{01})$, there exists $\sigma_2 \in L(N_2, M_{02})$ such that $\ell(\sigma_1) = \ell(\sigma_2)$, i.e., $\mathcal{L}(G_1, M_{01}) \subseteq \mathcal{L}(G_2, M_{02})$.

Analogously, we can prove that if $\mathcal{L}(G_1, M_{01}) \subseteq \mathcal{L}(G_2, M_{02})$ then G is current-state opaque wrt S. Indeed, if $\mathcal{L}(G_1, M_{01}) \subseteq \mathcal{L}(G_2, M_{02})$, then $L(N_1, M_{01}) \subseteq L(N_2, M_{02})$ and for any sequence $\sigma = \bar{t}_1 \sigma_1 \bar{t}_3$ that leads to the secret marking, it corresponds a sequence $\sigma' = \bar{t}_2 \sigma_2 \bar{t}_4$, where $\sigma_1 \in L(N_1, M_{01})$ and $\sigma_2 \in L(N_2, M_{02})$, that produces the same observation but leads to a nonsecret marking, i.e., G is current-state opaque wrt S.

Therefore, for the general case where the secret is an arbitrary subset of $R(N, M_0)$ and the initial marking set may not be a singleton, the Petri net current-state opacity problem is

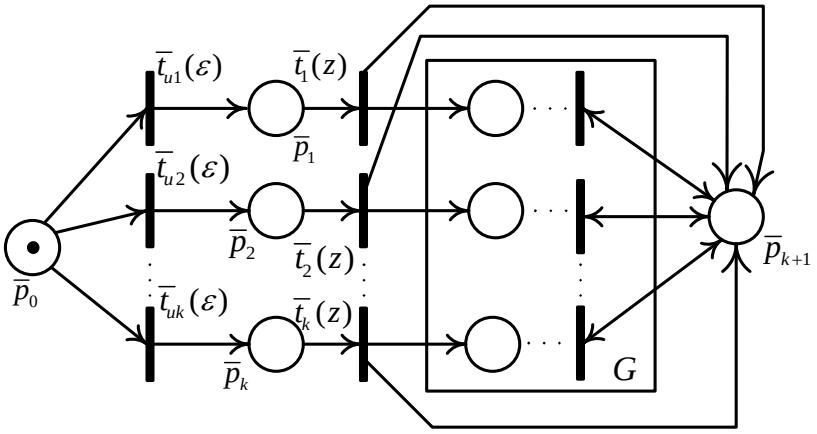

Fig. 5.2 The LPN system G' constructed in the proof of Theorem 5.2.

undecidable. □

5.3 Decidability of the Initial-State Opacity Problem

Definition 5.3. Consider an LPN system G and a secret set S. The *Petri net initial-state opacity problem* consists in determining whether G is initial-state opaque wrt S or not. ◇

Based on the results in [5], if the net is bounded, i.e., $R(N, M_{st})$ is finite, the Petri net ISO problem is decidable. On the other hand, the undecidability of the Petri net B-ISO problem does not imply its undecidability for a special class of $\mathcal{M}_0$. Therefore, it is necessary to investigate the decidability of the Petri net ISO problem. In the following, we prove its undecidability.

Theorem 5.2. The Petri net initial-state opacity problem is undecidable.

Proof: It has been proven that the B-ISO problem in Petri nets is undecidable [24], where $\mathcal{M}_0$ is finite. We prove this theorem by showing that the B-ISO problem for finite secret sets, which is undecidable, can be reduced into the Petri net ISO problem in polynomial time.

Consider an LPN system $G = (N, \mathcal{M}_0, \Sigma, \ell)$ with $|P| = m$ and $|T| = n$, where $\mathcal{M}_0 = \{M_0^1, M_0^2, \cdots, M_0^k\} \subseteq \mathbb{N}^m$ is a finite set of initial markings, and a secret set $S = \{M_0^1, M_0^2, \cdots M_0^r\} \subseteq \mathcal{M}_0$ with $r \leq k$. Starting from G, let us construct a new LPN system $G' = (N', M'_{st}, \Sigma', \ell')$, where $N' = (P', T', Pre', Post')$, by the following steps:

i) Add to G places: $\overline{p}_0, \overline{p}_1, \ldots, \overline{p}_{k+1}$, unobservable transitions $\overline{t}_{u1}, \overline{t}_{u2}, \ldots, \overline{t}_{uk}$, and observable transitions: $\overline{t}_1, \overline{t}_2, \ldots, \overline{t}_k$, such that $\ell(\overline{t}_1) = \cdots = \ell(\overline{t}_k) = z \notin \Sigma$.

ii) Add arcs: for $i = 1, 2, \cdots, k$, $Pre(\overline{p}_0, \overline{t}_{ui}) = 1$, $Pre(\overline{p}_i, \overline{t}_i) = 1$, $Post(\overline{p}_i, \overline{t}_{ui}) = 1$, $Post(\overline{p}_{k+1}, \overline{t}_i) = 1$; $\forall p \in P$ such that $M_0^i(p) \neq 0$, $Post(p, \overline{t}_i) = M_0^i(p)$; $\forall t \in T$, $Pre(\overline{p}_{k+1}, t) = 1$, $Post(\overline{p}_{k+1}, t) = 1$.

iii) $M'_{st} = \overline{p}_0$.

The resulting G' is depicted in Fig. 5.2. Obviously $\Sigma' = \Sigma \cup \{z\}$. Moreover, the number of places and transitions in G' are $|P'| = m + k + 2$ and $|T'| = n + 2k$, respectively. The firing of $\overline{t}_{ui}\overline{t}_i$ initializes G at M_0^i (for $i = 1, 2, \ldots, k$). Place $\overline{p}_{k+1}$ is added to prevent source transitions in G from firing before the firing of $\overline{t}_{ui}\overline{t}_i$.

Let us consider the secret set $S' = \{\overline{p}_0, \overline{p}_1, \ldots, \overline{p}_r\}$. In the following we prove that

$$G \text{ is B-ISO wrt S} \Leftrightarrow G' \text{ is ISO wrt } S'.$$

First we prove that if G' is ISO wrt S', then G is B-ISO wrt S. Assume that G' is ISO wrt S'. Then for any observation w generated from markings in S', there exists a marking $M' \in R(N', M'_{st}) \setminus S'$ from which the same observation w can be generated, i.e.,

$$\mathcal{L}(G', S') \subseteq \mathcal{L}(G', R(N', M'_{st}) \setminus S'). \tag{5-1}$$

By the structure of G',

$$\mathcal{L}(G', S') = \{w' \in E'^* | w' = zw, w \in \mathcal{L}(G, S)\} \tag{5-2}$$

holds. Moreover, the set of words in $\mathcal{L}(G', R(N', M'_{st}) \setminus S')$ having z as the prefix is equal to $\mathcal{L}(G', \{\overline{p}_{r+1}, \ldots, \overline{p}_k\})$. Therefore, by Eq. (5-1), we have

$$\mathcal{L}(G', S') \subseteq \mathcal{L}(G', \{\overline{p}_{r+1}, \ldots, \overline{p}_k\}). \tag{5-3}$$

Again by the structure of G', we have

$$\mathcal{L}(G', \{\overline{p}_{r+1}, \ldots, \overline{p}_k\}) = \{w' \in E'^* | w' = zw, w \in \mathcal{L}(G, \mathcal{M}_0 \setminus S)\} \tag{5-4}$$

By Eqs. (5-2), (5-3) and (5-4), it follows that $\mathcal{L}(G, S) \subseteq \mathcal{L}(G, \mathcal{M}_0 \setminus S)$. Namely, for all $M \in S$, $w \in \mathcal{L}(G, M)$, there exists $M' \in \mathcal{M}_0 \setminus S$ such that $w \in \mathcal{L}(G, M')$, i.e., G is B-ISO wrt S.

Following the same reasoning, we can prove that if G is B-ISO wrt S, then G' is ISO wrt S'. In more detail, if G is B-ISO wrt S, then $\mathcal{L}(G, S) \subseteq \mathcal{L}(G, \mathcal{M}_0 \setminus S)$ holds. This implies the inclusion relationship in Eq. (5-3) and, taking into account the structure of G', the inclusion relationship in Eq. (5-1). Therefore, we conclude that G' is ISO wrt S'. □

5.4 Decidability of the Language Opacity Verification Problem

Language opacity was first introduced in [29] in the framework of finite automata and then extended to Petri nets in Chapter 4. In the case of language opacity the secret is defined as a language. In this section we first recall the notion of language opacity in LPN systems, then we formalize the language opacity problem, and finally, we prove that such a problem is undecidable.

Definition 5.4. Let $G = (N, \mathcal{M}_0, \Sigma, \ell)$ be an LPN system and $S \subseteq L(N, \mathcal{M}_0)$ be a secret language. G is said to be *language opaque* (LO) wrt S if for all $\sigma \in S$, there exists $\sigma' \in L(N, \mathcal{M}_0) \setminus S$ such that $\ell(\sigma) = \ell(\sigma')$. $\diamond$

In other words, a system is language opaque wrt a given secret if for any observation that can be generated by a sequence in the secret, there exists another nonsecret sequence generating the same observation.

Definition 5.5. Consider an LPN system $G = (N, \mathcal{M}_0, \Sigma, \ell)$ and a secret language $S \subseteq L(N, \mathcal{M}_0)$. The *language opacity problem* consists in determining whether G is language opaque wrt S or not. $\diamond$

Theorem 5.3. The Petri net language opacity problem is undecidable.

Proof: The proof is carried out by showing that the Petri net current-state opacity problem for finite secret sets, which is proven undecidable by Theorem 5.1, can be reduced into the Petri net language opacity problem in polynomial time.

Consider an LPN system $G = (N, \mathcal{M}_0, \Sigma, \ell)$ and a secret set $S \subseteq R(N, \mathcal{M}_0)$. Let us prove that

$$G \text{ is CSO wrt } S \Leftrightarrow G \text{ is LO wrt } S',$$

where $S' = \{\sigma \in T^* | \exists M_0 \in \mathcal{M}_0, M \in S : M_0[\sigma\rangle M\}$.

First we prove that if G is current-state opaque wrt S, then G is language opaque wrt S'. Assume that G is current-state opaque wrt S. Then for any $M_0 \in \mathcal{M}_0$, $M \in S$ and $\sigma \in T^*$ such that $M_0[\sigma\rangle M$, there exist $M_0' \in \mathcal{M}_0$, $M' \in R(N, \mathcal{M}_0) \setminus S$ and $\sigma' \in T^*$ such that $M_0'[\sigma'\rangle M'$ and $\ell(\sigma) = \ell(\sigma')$. This implies that for all $\sigma \in S'$, there exists $\sigma' \in L(N, \mathcal{M}_0) \setminus S'$ with $\ell(\sigma) = \ell(\sigma')$. Therefore, G is language opaque wrt S'.

Now we prove that if G is language opaque wrt S', then G is current-state opaque wrt S. Assume that G is language opaque wrt S'. Then for any $\sigma \in L(N, \mathcal{M}_0) \cap S'$, there exists at least a firing sequence $\sigma' \in L(N, \mathcal{M}_0) \setminus S'$ such that $\ell(\sigma') = \ell(\sigma)$. Since $\sigma' \notin S'$,

$M' \notin S$, where $M_0[\sigma'\rangle M'$ and $M_0 \in \mathcal{M}_0$. Namely, for any transition sequence leading to a marking in S, there exists a transition sequence producing the same observation but leading to a marking not in S. Therefore, G is current-state opaque wrt S. □

Corollary 5.1. The Petri net language opacity problem in bounded Petri net systems is decidable.

Proof: Follows from the proof of Theorem 5.3. □

Furthermore, verification of strict language opacity in bounded Petri nets is decidable, since by Corollary 4.1 verification of strict language opacity can be reduced to verification of language opacity.

5.5 Conclusion

In this chapter, the decidability of current-state, initial-state and language opacity problems in Petri nets is addressed, where initial-state opacity is a special case of B-initial-state opacity defined in [5]. In particular, showing that all such problems are undecidable for special classes of secrets, we conclude that, in general, Petri net current-state, initial-state, and language opacity problems are undecidable since if a problem is undecidable under special assumptions (e.g., the secret set is finite), the same problem under less restrictive assumptions is obviously undecidable as well.

The work of this chapter has been accepted as:

Y. Tong, Z. W. Li, C. Seatzu, and A. Giua, "Decidability of Opacity Problems in Labeled Petri Nets", *Automatica*, **2017**.

Chapter 6 Opacity Verification Using Petri Nets

6.1 Introduction

In this chapter, we focus on the verification of two important state-based opacity properties: current-state opacity (CSO) and initial-state opacity (ISO), and strict language opacity (SLO) in DESs modeled by bounded labeled Petri nets, which have been proven decidable in Chapter 5. The system under consideration is modeled by a bounded *labeled Petri net* (LPN), where the observation function is static and the states are not observable [23, 24].

To overcome the state explosion, in this chapter we use the notions of *basis markings* and *minimal explanations*. Such notions have been first introduced in [59–63] to solve the problems of state estimation, fault diagnosis, diagnosability analysis and reachability analysis in LPNs. They allow one to avoid an exhaustive enumeration of the reachability space. Only a subset of reachable markings, i.e., the basis markings, should be enumerated, while other reachable markings are characterized by linear systems, one for each basis marking. Therefore, the RG can be compactly represented by the *basis reachability graph* (BRG), a graph describing the transition relation between basis markings.

The main contributions of this chapter can be summarized as follows:

1. Necessary and sufficient conditions for current-state opacity with respect to an arbitrary secret are provided. A novel approach based on the BRG (with appropriate changes) is proposed that enables one to avoid RG analysis. Moreover, if the secret is defined as the intersection of a series of generalized mutual exclusion constraints (GMECs), then current-state opacity can be verified by solving a set of integer linear programming problems (ILPPs) instead of exhaustively enumerating the unobservable reach of basis markings. Finally, if the incidence matrix is totally unimodular, then these ILPPs can be relaxed to linear programming problems (LPPs).

2. We define *exposable* and *weakly exposable* markings. In particular, we prove that if no weakly exposable marking is contained in the secret, then current-state opacity can be efficiently verified without solving ILPPs. Moreover, the proposed approach is extended to the case where the intruder has uncertainties about the initial marking.

3. We provide necessary and sufficient conditions for initial-state opacity with respect to an arbitrary secret. We show that if no weakly exposable marking belongs to the

secret, initial-state opacity can be efficiently verified using the BRG. Otherwise, we propose a modified BRG (MBRG) to verify initial-state opacity.

4. A MATLAB tool is developed to implement most of the proposed approaches. Numerical results are illustrated to corroborate their effectiveness.

5. A finite structure called *verifier* is proposed to verify strict language opacity. Such an approach works under the assumption that the intruder is interested in the set of observable transitions and the secret is the set of all firable transition sequences in a bounded LPN (excluding the empty string). The proposed approach is proven of lower complexity than other methods in literature.

The rest of this chapter is organized as follows. Firstly, in Section 6.2 we briefly recall the notion of minimal explanations, basis markings and basis reachability graphs. In Section 6.3, approaches to verifying CSO are proposed. Verification of ISO is addressed in Section 6.4. Numerical examples for verifying CSO and ISO are presented in Section 6.5. The method of verifying SLO is developed in Section 6.6. Finally, conclusions are drawn.

6.2 Basis Reachability Graph

In [59, 60], a compact way to represent the reachability set of a Petri net was proposed to solve the fault diagnosis problem. Under the assumption that the T_u-induced (unobservable) subnet is acyclic, only a subset of the reachable markings, called *basis markings*, are computed, while, all non-basis markings are characterized by a set of linear equations associated with each basis marking. Using the notion of basis markings, the *basis reachability graph* (BRG) is defined. The BRG as proposed in [59, 60] also includes some diagnosis information, which are redundant for its application in the thesis. Herein we redefine it neglecting such information. Now we recall some key definitions from [60].

Definition 6.1. Given a marking M and an observable transition $t \in T_o$, we define

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

as the set of *explanations* of t at M and $Y(M,t) = \{y_u \in \mathbb{N}^{n_u} | \exists \sigma \in \Sigma(M,t) : y_u = \pi(\sigma)\}$ as the set of e*-vectors*. $\diamond$

Thus $\Sigma(M,t)$ is the set of unobservable sequences whose firing at M enables t and $Y(M,t)$ is the set of firing vectors of the explanations. Among all the explanations, we are interested in finding the minimal ones, i.e., those whose firing vector is minimal.

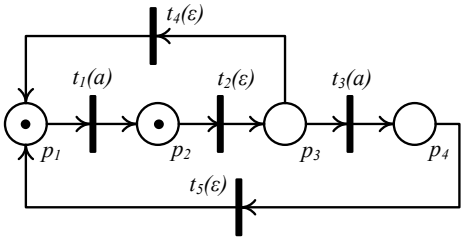

Fig. 6.1 LPN where $\ell(t_1) = \ell(t_3) = a$.

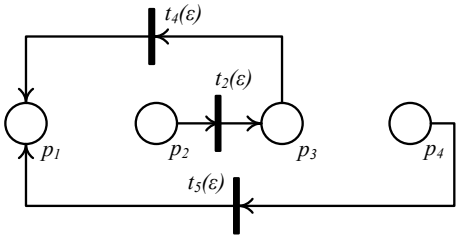

Fig. 6.2 T_u-induced subnet of the LPN in Fig. 6.1.

Definition 6.2. Given a marking M and an observable transition $t \in T_o$, we define

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

as the set of *minimal explanations* of t at M and $Y_{min}(M,t) = \{y_u \in \mathbb{N}^{n_u} | \exists \sigma \in \Sigma_{min}(M,t) : y_u = \pi(\sigma)\}$ as the corresponding set of *minimal* e*-vectors*. $\diamond$

Generally, given a marking M and a transition t, its minimal explanation $\Sigma_{min}(M,t)$ is not a singleton because an unobservable transition may have more than one input places whose input transitions are also unobservable. Many approaches can be applied to computing $Y_{min}(M,t)$. In particular, when the T_u-induced subnet is acyclic the approach proposed by Cabasino *et al*. [60] only requires algebraic manipulations, which is presented in Algorithm 4.

Algorithm 4 Computation of $Y_{min}(M,t)$

Input: A Petri net N, a marking M, and a transition $t \in T_o$
Output: $Y_{min}(M,t)$

1: Let $\Gamma = \left[\begin{array}{c|c} C_u^T & I_{n_u \times n_u} \\ \hline A & B \end{array}\right]$ where $A := (M - Pre(\cdot,t))^T$, $B := \vec{0}_{n_u}^T$;
2: **while** A has negative entries, **do**
3: choose an element $A(i^*, j^*) < 0$;
4: let $\mathcal{I}^+ = \{i | C_u^T(i, j^*) > 0\}$;
5: **for all** $i \in \mathcal{I}^+$, **do**
6: add to $[A|B]$ a new row

$$[A(i^*,\cdot) + C_u^T(i,\cdot) | B(i^*,\cdot) + \vec{e_i}^T],$$

where $\vec{e_i}$ is the i-th canonical basis vector;
7: **end for**
8: remove the row $[A(i^*,\cdot)|B(i^*,\cdot)]$ from the table;
9: **end while**
10: Remove from B any row that covers other rows;
11: Each row of B is a vector in $Y_{min}(M,t)$.

Example 6.1. Consider the LPN system in Fig. 6.1. Its T_u-induced subnet (see Fig. 6.2) is acyclic. Let $M = [0\ 1\ 0\ 1]^T$ and $t = t_1$. Then $(M - Pre(\cdot, t))^T = [-1\ 1\ 0\ 1]^T$ and

$$\Gamma = \left[\begin{array}{cccc|ccc} 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ \hline \underline{-1} & 1 & 0 & 1 & 0 & 0 & 0 \end{array}\right].$$

There is only one element of A, i.e., $A(1,1) = -1$, is negative. Moreover, $\mathcal{I}^+ = \{2, 3\}$. Following Steps 5 to 8 in Algorithm 4, first choosing $i = 2$, we add Γ a new row

$$|0 \quad 1 \quad -1 \quad 1 \,|\, 0 \quad 1 \quad 0|$$

obtained from the first row of A adding $\Gamma(2, \cdot)$. Then choosing $i = 3$, we add Γ a new row

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

obtained from the first row of A adding $\Gamma(3, \cdot)$. Finally, remove $A(1, \cdot)$ obtaining

$$\Gamma = \left[\begin{array}{cccc|ccc} 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ \hline 0 & 1 & \underline{-1} & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right].$$

Now $A(1, 3)$ is negative and we continue computing $\mathcal{I}^+ = \{1\}$. We add the following new row to Γ:

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

obtained from the first row of A adding $\Gamma(1, \cdot)$. Finally, we remove $A(1, \cdot)$ from Γ obtaining

$$\Gamma = \left[\begin{array}{cccc|ccc} 0 & -1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right].$$

Now we can stop because all entries of A are non negative. Since no rows of B cover the other, both rows of B, namely

$$|0\ 0\ 1|, \text{ and } |1\ 1\ 0|$$

are elements of $Y_{min}(M, t)$. ◇

Based on the notion of minimal explanations, the set of basis markings can be recursively defined.

Definition 6.3. Given an LPN system $G = (N, M_0, \Sigma, \ell)$, its set of basis markings $\mathcal{M}_B$ is a subset of $R(N, M_0)$ such that:

a) $M_0 \in \mathcal{M}_B$;

b) $\forall M \in \mathcal{M}_B, \forall t \in T_o, \forall y_u \in Y_{min}(M, t)$, it holds $M' \in \mathcal{M}_B$, where $M' = M + C(\cdot, t) + C_u \cdot y_u$. $\diamond$

In other words, the set of basis markings includes the initial marking and the set of all markings reachable by firing observable transitions together with their minimal explanations. All other intermediate markings reachable by the firing of unobservable transitions are disregarded.

Based on Definition 6.3, the following Algorithm 5 iteratively computes basis markings and constructs the BRG. We now briefly explain how Algorithm 5 works. The set $\mathcal{M}_B$ is initialized at $\mathcal{M}_B = \{M_0\}$. For all markings M in $\mathcal{M}_B$ that have not been studied yet, i.e., with no tag, and for all observable transitions t, we check whether the set of minimal e-vectors $Y_{min}(M, t)$ is not empty. If not, we compute the resulting basis markings. This procedure runs iteratively until there is no unchecked marking in $\mathcal{M}_B$.

We denote the BRG as an NFA $\mathcal{B} = (\mathcal{M}_B, \Sigma, \delta, M_0)$, where $\mathcal{M}_B$ is the state space, all events are observable, δ is the transition relation between basis markings, and M_0 is the initial state.

As Algorithm 5 shows, to construct the BRG one only needs to explore the minimal e-vectors for each basis marking and observable transition. This prevents us from exhaustively exploring the RG. Therefore, the complexity of constructing the BRG is lower than that of constructing the RG. It has been shown that in practical cases the size of the BRG can be order of magnitude smaller than that of the RG [63]. Given a word $w \in L(\mathcal{B})$, based on Algorithm 5, if $\delta(M_0, w) = M$ is defined in $\mathcal{B}$ then M is the basis marking reachable from M_0 by firing an observable sequence σ_o that produces w, eventually interleaved with some unobservable transitions whose firing is necessary to enable σ_o. We use

$$\mathcal{M}_b(w) = \mathcal{C}(w) \cap \mathcal{M}_B$$

to denote the set of basis markings consistent with w.

Example 6.2. Let us consider again the LPN system in Fig. 6.1. It has 10 reachable markings and its RG is shown in Fig. 6.3. However, there are only 5 basis markings $\mathcal{M}_B = \{M_0 - M_4\}$, and the corresponding BRG is shown in Fig. 6.4. For clarity of presentation, transitions are added in parenthesis on arcs even if they are not provided by Algorithm 5. Note that they

Algorithm 5 Construction of the BRG

Input: A bounded LPN system $G = (N, M_0, \Sigma, \ell)$ whose unobservable subnet is acyclic.
Output: The BRG $\mathcal{B} = (\mathcal{M}_B, \Sigma, \delta, M_0)$.

1: $\mathcal{M}_B := \{M_0\}$ and assign no tag to M_0;
2: **while** states with no tag exist, **do**
3: select a state $M \in \mathcal{M}_B$ with no tag;
4: **for all** $t \in T_o$ and $Y_{min}(M,t) \neq \emptyset$, **do**
5: **for all** $y_u \in Y_{min}(M,t)$, **do**
6: $M' := M + C_u \cdot y_u + C(\cdot, t)$;
7: $\delta(M, \ell(t)) := \emptyset$;
8: **if** $M' \notin \mathcal{M}_B$, **then**
9: $\mathcal{M}_B := \mathcal{M}_B \cup \{M'\}$;
10: assign no tag to M';
11: **end if**
12: $\delta(M, \ell(t)) := \delta(M, \ell(t)) \cup \{M'\}$;
13: **end for**
14: tag node M "old";
15: **end for**
16: **end while**
17: Remove all tags.

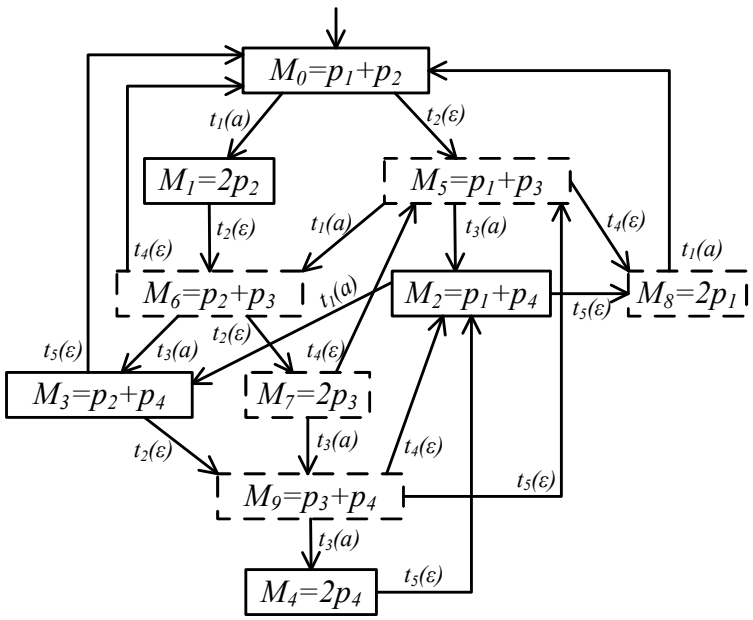

Fig. 6.3 RG of the LPN system in Fig. 6.1.

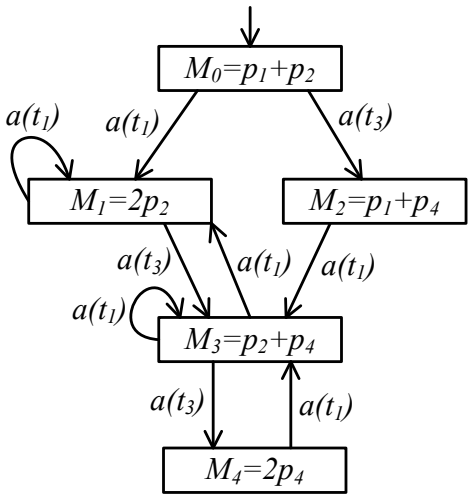

Fig. 6.4 BRG of the LPN system in Fig. 6.1.

should not be taken into account when establishing whether the BRG is either deterministic or not. ◇

Notice that to apply the BRG, two assumptions are made:

A1) the LPN system G is bounded, and

A2) the unobservable subnet of G is acyclic.

Assumption A1 guarantees that the number of basis markings is finite thus Algorithm 5 can halt and the BRG can be constructed. Assumption A2 allows us to iteratively compute the basis markings and to use the state equation to characterize the set of markings reachable from a basis marking by firing unobservable transitions (as shown in Theorem 6.1 after).

Let us now introduce the following definition that is useful to formalize the main result in this subsection.

Definition 6.4. Given an LPN system $G = (N, M_0, \Sigma, \ell)$ and a marking $M \in R(N, M_0)$, the *unobservable reach* of M is defined as $\mathcal{U}(M) = \{M' \in \mathbb{N}^m | \exists \sigma_u \in T_u^* : M[\sigma_u\rangle M'\}$. ◇

In simple words, the unobservable reach of a marking M is the set of markings reachable from M by firing only unobservable transitions.

Sets $\mathcal{M}_b(w)$ can be computed through constructing the observer of the BRG. In [59, 60], it has been proved that the set of markings consistent with an observation w can be characterized by the unobservable reaches of basis markings in $\mathcal{M}_b(w)$.

Table 6.1 Unobservable reaches of markings in Fig. 6.3

Marking M	$\mathcal{U}(M)$	Marking M	$\mathcal{U}(M)$
M_0	$\{M_0, M_5, M_8\}$	M_5	$\{M_5, M_8\}$
M_1	$\{M_0, M_1, M_5, M_6, M_7, M_8\}$	M_6	$\{M_0, M_5, M_6, M_7, M_8\}$
M_2	$\{M_2, M_8\}$	M_7	$\{M_5, M_7, M_8\}$
M_3	$\{M_0, M_2, M_3, M_5, M_8, M_9\}$	M_8	$\{M_8\}$
M_4	$\{M_2, M_4, M_8\}$	M_9	$\{M_2, M_5, M_8, M_9\}$

Theorem 6.1. [60] Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose T_u-induced subnet is acyclic. For all $w \in \mathcal{L}(G, M_0)$, it holds that

$$\begin{aligned}\mathcal{C}(w) &= \bigcup_{M_b \in \mathcal{M}_b(w)} \mathcal{U}(M_b) \\ &= \bigcup_{M_b \in \mathcal{M}_b(w)} \{M \in \mathbb{N}^m | \exists y_u \in \mathbb{N}^{n_u} : M = M_b + C_u \cdot y_u\}.\end{aligned}$$

In words, given an LPN system whose unobservable subnet is acyclic, and an observation w, a marking M is consistent with w if and only if it belongs to the unobservable reach of a basis marking M_b that is consistent with w. Since the unobservable subnet is acyclic, marking M belonging to $\mathcal{U}(M_b)$ means that $M = M_b + C_u \cdot y_u$ has a non-negative integer solution y_u.

Example 6.3. Consider the LPN system in Fig. 6.1. Unobservable reaches of all reachable markings are listed in Table 6.1. One can compute them by looking at the RG or by solving the equation in Theorem 6.1.

As discussed above, only markings M_0 to M_4 are basis markings. It can be easily observed that the union of the unobservable reaches of basis markings equals the set of reachable markings. ◇

Corollary 6.1 follows from Theorem 6.1.

Corollary 6.1. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose T_u-induced subnet is acyclic. There exists a firing sequence $\sigma = \sigma_{u1} t_{i1} \cdots \sigma_{uk} t_{ik}$ such that $M_0[\sigma_{u1}\rangle M_1[t_{i1}\rangle M_1' \cdots M_{k-1}[\sigma_{uk}\rangle M_k[t_{ik}\rangle$, where $\sigma_{uj} \in T_u^*$ and $t_{ij} \in T_o$, if and only if there exists $\sigma' = \sigma_{u1}' t_{i1} \cdots \sigma_{uk}' t_{ik}$ such that $M_0[\sigma_{u1}'\rangle \hat{M}_1[t_{i1}\rangle \hat{M}_1' \cdots \hat{M}_{k-1}[\sigma_{uk}'\rangle \hat{M}_k[t_{ik}\rangle$, where $\sigma_{uj}' \in \Sigma_{min}(\hat{M}_j, t_{ij})$.

In simple words, a sequence σ whose projection on T_o is $t_{i1} t_{i2} \cdots t_{ik}$ is firable if and only if there exists σ' such that the minimal explanation of each observable transition is not empty. Namely, to check if $\exists \sigma \in L(N, M_0) : P_{T_o}(\sigma) = t_{i1} t_{i2} \cdots t_{ik}$ there is no need to enumerate all $\sigma_u \in T_u^*$ that enable $t_{i1}, t_{i2}, \cdots, t_{ik}$ but only the minimal explanations. Markings $\hat{M}_j'$ are basis markings.

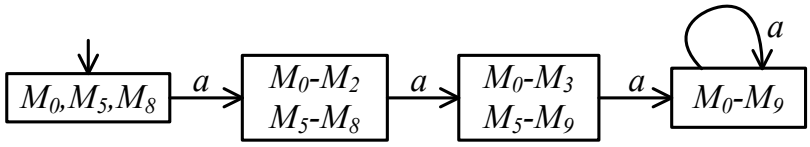

Fig. 6.5 Observer of the RG in Fig. 6.3.

6.3 Verification of Current-State Opacity

According to the definition of CSO (see Definition 4.4), we have the following fact:

Fact 1: An LPN system is CSO wrt a secret $S \subseteq R(N, M_0)$ if and only if

$$\forall w \in \mathcal{L}(G, M_0) : \mathcal{C}(w) \not\subseteq S.$$

In other words, to verify current-state opacity of an LPN system, we need to check if $\mathcal{C}(w) \not\subseteq S$ holds for all $w \in \mathcal{L}(G, M_0)$, which means that all sets $\mathcal{C}(w)$ need to be computed first. For a bounded LPN system, this can be done by constructing the *observer (automaton)* (see Section 2.1.2) of its RG.

Example 6.4. Let us consider the LPN system in Fig. 6.1. Let $S = \{M_2, M_3, M_6, M_7, M_8\}$. The observer of the RG (see Fig. 6.3) is shown in Fig. 6.5. Since none of the states of the observer is a subset of S, the LPN system is CSO wrt S. ◇

Clearly, computing all sets $\mathcal{C}(w)$, in general, requires to exhaustively enumerate all sequences of transitions that may fire, and the complexity of computing the observer of a given NFA with $|X|$ states is $\mathcal{O}(2^{|X|})$. Therefore, if the RG is too large, it may be impossible to construct the observer. In this section, based on the notion of basis markings and minimal explanations, an efficient approach to verifying current-state opacity is proposed. Let us first introduce the following definition.

Definition 6.5. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system and $S \subseteq R(N, M_0)$ be a secret. A reachable marking M is said to be *exposable* if it does not belong to the secret, i.e., $M \in R(N, M_0) \setminus S$. The set of exposable markings is $ex(S) = R(N, M_0) \setminus S$. A marking M is said to be *weakly exposable* if there exists a marking $M' \in R(N, M_0)$ such that $M' \in \mathcal{U}(M) \cap ex(S)$. The set of weakly exposable markings is denoted as $wex(S)$. ◇

In simple words, a marking M is weakly exposable if there exists an exposable marking M' that is reachable from it by firing unobservable transitions. Note that the firing sequence of unobservable transitions could be empty. Therefore, all exposable markings are also weakly exposable. Their relations are depicted by the Venn diagram in Fig. 6.6.

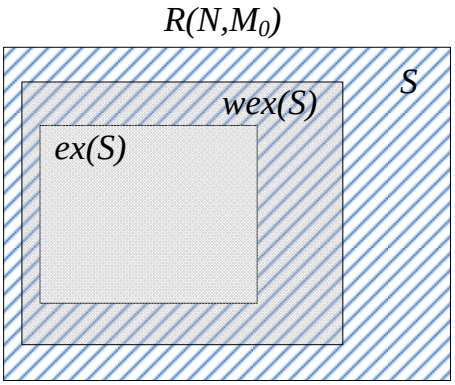

Fig. 6.6 Inclusion relationships among secret, exposable, weakly exposable, and reachable marking sets.

Example 6.5. Consider again the LPN system in Fig. 6.1. Given a secret $S = \{M_2, M_3, M_6, M_7, M_8\}$. The set of exposable markings is $ex(S) = \{M_0, M_1, M_4, M_5, M_9\}$. According to Table 6.1, $\mathcal{U}(M_2) \subseteq S, \mathcal{U}(M_3) \nsubseteq S, \mathcal{U}(M_6) \nsubseteq S, \mathcal{U}(M_7) \nsubseteq S$, and $\mathcal{U}(M_8) \subseteq S$. Therefore, the set of weakly exposable markings is $wex(S) = \{M_0, M_1, M_3 - M_7, M_9\}$. ◇

From Fact 1 and Definition 6.5, the following corollary follows.

Corollary 6.2. $G = (N, M_0, \Sigma, \ell)$ is current-state opaque wrt S iff $\forall w \in \mathcal{L}(G, M_0), \mathcal{C}(w) \cap ex(S) \neq \emptyset$ holds.

Example 6.6. Consider Examples 6.4 and 6.5. The LPN system is current-state opaque wrt S since $\forall w \in \mathcal{L}(G, M_0), \mathcal{C}(w) \cap ex(S) \neq \emptyset$. ◇

Based on Theorem 6.1, we derive the following necessary and sufficient condition for current-state opacity.

Theorem 6.2. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic and $S \subseteq R(N, M_0)$ be a secret. G is current-state opaque wrt S iff $\forall w \in \mathcal{L}(G, M_0)$, $\mathcal{M}_b(w) \cap wex(S) \neq \emptyset$ holds.

Proof: ($\Rightarrow$) Given an arbitrary observation $w \in \mathcal{L}(G, M_0)$, if there exists a basis marking $M_b \in \mathcal{M}_b(w)$ that is weakly exposable, then there is a marking $M \in \mathcal{U}(M_b)$ such that $M \in ex(S)$, and hence $M \in \mathcal{C}(w)$. This indicates that $\mathcal{C}(w) \cap ex(S) \neq \emptyset$. By Fact 1, the system is current-state opaque wrt S.

($\Leftarrow$) Assume that there is an observation $w \in \mathcal{L}(G, M_0)$ and none of the basis markings consistent with w are weakly exposable, i.e., $\forall M_b \in \mathcal{M}_b(w)$, $\mathcal{U}(M_b) \subseteq S$. Based on Theorem 6.1, all markings consistent with observation w belong to the secret, i.e., $\mathcal{C}(w) \cap ex(S) = \emptyset$. By Fact 1, the LPN system is not current-state opaque. □

As a result, instead of exhaustively computing the sets $\mathcal{C}(w)$ for all $w \in \mathcal{L}(G, M_0)$, according to Theorem 6.2, to determine if an LPN system is current-state opaque, we only need to compute the set of basis markings $\mathcal{M}_b(w)$ for all observations $w \in \mathcal{L}(G, M_0)$ and to check if it contains a weakly exposable basis marking.

6.3.1 BRG for Current-State Opacity

In this section, we propose a modified BRG that enables us to verify current-state opacity more efficiently.

Given a bounded LPN system G and a secret S, with each node $M_b \in \mathcal{M}_B$ of the BRG $\mathcal{B} = (\mathcal{M}_B, \Sigma, \delta, M_0)$ we associate a binary scalar $\alpha(M_b)$ defined as follows:

$$\alpha(M_b) = \begin{cases} 1, & \text{if } M_b \text{ is weakly exposable;} \\ 0, & \text{otherwise.} \end{cases} \tag{6-1}$$

The BRG for current-state opacity is denoted as $\mathcal{B}_c = (\tilde{\mathcal{M}}_B, \Sigma, \delta_c, (M_0, \alpha(M_0)))$, where $\tilde{\mathcal{M}}_B \subseteq \mathcal{M}_B \times \{0, 1\}$.

For all observations w, $\mathcal{M}_b(w)$ can be computed by converting the obtained BRG into its equivalent DFA by a standard determinization procedure. In the resulting DFA, called *current-state basis observer*, each state is a subset of $\tilde{\mathcal{M}}_B$ consistent with an observation. According to Theorem 6.2, if all states of the observer have at least a pair $(M, \alpha(M))$ with $\alpha(M) = 1$, the LPN is current-state opaque wrt S; otherwise, the LPN is not current-state opaque.

The number of states of the current-state basis observer in the worst case is $2^{|\mathcal{M}_B|} - 1$. Therefore, the space complexity of the proposed approach is $\mathcal{O}(2^{|\mathcal{M}_B|})$. However, since the RG-based approach has a space complexity of $\mathcal{O}(2^{|R(N,M_0)|})$, and $|\mathcal{M}_B|$ is typically greatly smaller than $|R(N, M_0)|$, we conclude that the BRG-based method is practically much more efficient. Some numerical results that validate this are given in Section 6.5. Moreover, once the current-state basis observer is constructed, there is no need to reconstruct it when the secret S changes. If S is changed to S', all we need is to update the value of $\alpha(M)$ in the current-basis observer for each basis marking M.

Example 6.7. Consider the LPN in Fig. 6.1 and the same secret $S = \{M_2, M_3, M_6, M_7, M_8\}$ in Example 6.5. By Eq. (6-1), the BRG for current-state opacity is illustrated in Fig. 6.7 and the corresponding observer is shown in Fig. 6.8. Since all nodes of the observer have at least a pair $(M, \alpha(M))$ with $\alpha(M) = 1$, then by Theorem 6.2, the LPN is current-state opaque wrt S. ◇

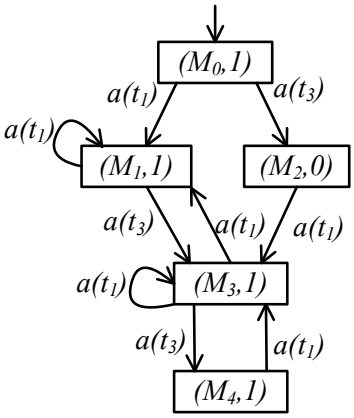

Fig. 6.7 BRG $\mathcal{B}_c$ for current-state opacity in Example 6.7.

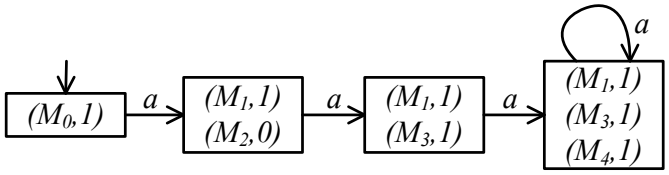

Fig. 6.8 Current-state basis observer of the BRG $\mathcal{B}_c$ in Fig. 6.7.

The following proposition provides a sufficient but not necessary condition for verifying current-state opacity without constructing the observer of the BRG.

Proposition 6.1. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic and $S \subseteq R(N, M_0)$ be a secret. If all basis markings $M_b \in \mathcal{M}_B$ of G are weakly exposable, i.e., $\mathcal{M}_B \subseteq wex(S)$, the system is current-state opaque wrt S.

Proof: Since all basis markings are weakly exposable, namely, for all observations $w \in \mathcal{L}(G, M_0)$, there is an exposable marking in $\mathcal{C}(w)$, according to Theorem 6.2, the LPN system is current-state opaque. □

If all states of the BRG have $\alpha(\cdot) = 1$, the LPN is current-state opaque; otherwise, current-state opacity requires further analysis. The result of Example 6.7 shows that the condition in Proposition 6.1 is not necessary for current-state opacity: even though in the BRG there is basis markings $M_2 \in \mathcal{M}_b(a)$ that is not weakly exposable, the LPN is current-state opaque wrt S. When $w = a$ is observed, consistent markings reached from basis marking M_2 belong to the secret. The intruder, however, still does not know for sure if the current state is in the secret, since the current state could be the one that is reachable from $M_1 \in \mathcal{M}_b(a)$ and that does not belong to the secret. For example, the current state could be M_5.

6.3.2 Secrets Described by GMECs

Let us now discuss some special cases for which the computation of the scalars $\alpha(M)$ could be simplified. To verify if a marking M is weakly exposable requires to test if there exists a nonsecret marking M' that belongs to its unobservable reach. This can be done exhaustively by solving the reachability problem in its unobservable subnet. If the unobservable subnet is acyclic, this can be done by checking if $M' = M + C_u \cdot y_u$ has a nonnegative integer solution. However, under special assumptions on the secret and/or the net structure, there may exist a more efficient way to do that. In this subsection we show that such is the case when the secret S is described by a set of *generalized exclusion mutual constraints* (GMECs) [64]. It is well-known that GMECs describe interesting subsets of the state space of a net and many interesting state-based specifications can be represented by GMECs. Furthermore, they allow one to solve analysis and control problems by means of simple linear algebraic tools [65–68]. We show that in such a case determining if a basis marking is weakly exposable does not require constructing the reachability set of the unobservable subnet, but only finding if a given set of linear integer constraints admits a feasible solution.

Definition 6.6. [64] Given a net N, a *single GMEC* is a pair $(\mathbf{w}, k)$, where $\mathbf{w} \in \mathbb{Z}^m, k \in \mathbb{Z}$, defining a set of legal markings $\mathcal{M}_{(\mathbf{w},k)} = \{M \in \mathbb{N}^m \mid \mathbf{w}^T \cdot M \leq k\}$. A *conjunctive GMEC* is a pair (W, K) where $W \in \mathbb{Z}^{r\times m}, K \in \mathbb{Z}^r$ defining a set of legal markings $\mathcal{M}_{(W,K)} = \{M \in \mathbb{N}^m \mid W^T \cdot M \leq K\}$. Given a conjunctive GMEC (W, K), we use $(\mathbf{w}_i, k_i)$ to denote the single GMEC $(W(i,\cdot), K(i))$. ◇

In this subsection we assume that the secret is described by a conjunctive GMEC, i.e.,

$$S = \{M \in \mathbb{N}^m \mid W^T \cdot M \leq K\}.$$

Definition 6.7. Let $M \in R(N, M_0)$ be a marking of an LPN system $G = (N, M_0, \Sigma, \ell)$, $S = \{M \in \mathbb{N}^m | W \cdot M \leq K\}$ be a secret and $(\mathbf{w}_i, k_i)$ be the i-th GMEC of the secret. The (i, M)-*constraint set* is defined as

$$\mathcal{Y}_i(M) = \begin{cases} M' = M + C_u \cdot y_u \\ \mathbf{w}_i^T \cdot M' > k_i \\ y_u \in \mathbb{N}^{n_u} \\ M' \in \mathbb{N}^m \end{cases}$$ ◇

Proposition 6.2. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic and $S = \{M \in \mathbb{N}^m | W \cdot M \leq K\}$ be a secret. A reachable marking $M \in R(N, M_0)$

is weakly exposable iff there exists a GMEC $(\mathbf{w}_i, k_i)$ of the secret such that the corresponding (i, M)-constraint set is feasible.

Proof: ($\Rightarrow$) Given a marking $M \in R(N, M_0)$, if there exists a GMEC whose (i, M)-constraint set is feasible, then there exists a marking M' that is reachable from M by firing unobservable transitions and that does not belong to the secret, i.e., M is weakly exposable.

($\Leftarrow$) If M is weakly exposable, then there exists a marking $M' \notin S$ with $M[\sigma\rangle M'$, $\sigma \in T_u^*$. Therefore, there exists a GMEC $(\mathbf{w}_j, k_j)$ such that M' and vector $y = \pi(\sigma)$ is a solution to the (j, M)-constraint set. □

In other words, when the secret is described by GMECs, verifying if a marking is weakly exposable can be done by solving integer linear programming problems (ILPPs). Therefore, the construction of BRG for current-state opacity requires solving $r \cdot z$ ILPPs, where r is the number of GMECs and z is the number of basis markings. Moreover, for some net structures the complexity of constructing the BRG can be further reduced by relaxing an ILPP into a linear programming problem (LPP).

Lemma 6.1. [69] If A is a totaly unimodular matrix[1] and b is a vector of integers, then a linear programming problem of the form $\min \{c \cdot x \mid A \cdot x \geq b, x \geq 0\}$ or $\max \{c \cdot x \mid A \cdot x \leq b\}$ has an integer optimal solution, for any c.

Proposition 6.3. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic, the corresponding incidence matrix C_u be a totally unimodular matrix, and $S = \{M \in \mathbb{N}^m | W \cdot M \leq K\}$ be a secret. A reachable marking $M \in R(N, M_0)$ is weakly exposable iff there exists a GMEC $(\mathbf{w}_i, k_i)$ whose (i, M)-constraint set $\mathcal{Y}_i(M)$ is feasible for $y \in \mathbb{R}^{n_u}_{\geq 0}$ and $M' \in \mathbb{R}^m_{\geq 0}$.

Proof: Trivially follows from Proposition 6.2 and Lemma 6.1. □

Note that there exist many interesting classes of nets whose incidence matrix is totally unimodular: examples are marked graphs and state machines [70].

Example 6.8. Consider again the LPN system in Fig. 6.1 whose unobservable subnet is a state machine. Let the secret be $S = \{M \in \mathbb{N}^4 | M(p_1) + M(p_4) \geq 2\}$, i.e., $W = [\ -1 \quad 0 \quad 0 \quad -1\ ]$ and $K = -2$. Since the observer of the BRG has been constructed in Example 6.7, only the value $\alpha(\cdot)$ of each basis marking needs to be updated. By solving the LPP, we obtain $\alpha(M_0) = 1$, $\alpha(M_1) = 1$, $\alpha(M_2) = 0$, $\alpha(M_3) = 1$, and $\alpha(M_4) = 0$. According to Theorem 6.2, the LPN system is current-state opaque wrt the secret S. ◇

[1] A matrix A is totally unimodular if each subdeterminant of A is 0, 1, or -1.

Notice that the observer of the BRG still needs to be constructed first. Providing a necessary but not sufficient condition for current-state opacity, Proposition 6.4 can be applied without constructing the observer and only requires solving LPPs.

Proposition 6.4. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic and $S = \{M \in \mathbb{N}^m | W \cdot M \leq K\}$ be a secret. The LPN is not current-state opaque if for all basis markings $M_b \in \mathcal{M}_B$ and all GMECs $(\mathbf{w}_i, k_i)$, the (i, M_b)-constraint sets $\mathcal{Y}_i(M_b)$ are not feasible for $y \in \mathbb{R}^{n_u}_{\geq 0}$ and $M \in \mathbb{R}^m_{\geq 0}$.

Proof: Given a basis marking M_b, if for all GMECs $(\mathbf{w}_i, k_i)$ the (i, M_b)-constraint sets $\mathcal{Y}_i(M_b)$ is not feasible for $y \in \mathbb{R}^{n_u}_{\geq 0}$ and $M \in \mathbb{R}^m_{\geq 0}$, they are not feasible for $y \in \mathbb{N}^{n_u}_{\geq 0}$ and $M \in \mathbb{N}^m_{\geq 0}$ either. According to Proposition 6.2, basis marking M_b is not weakly exposable. Since none of the basis markings is weakly exposable, by Theorem 6.2, the LPN system is not current-state opaque. □

6.3.3 Secrets with No Weakly Exposable Markings

In this subsection we focus on a special class of secrets. More precisely, given an LPN system, we assume that the secret satisfies the following additional assumption:

A3) none of the secret markings is weakly exposable, i.e., $M \in S \Rightarrow \forall M' \in \mathcal{U}(M) : M' \in S$ holds.

This means that if M is a secret marking, all markings in the unobservable reach of M are secret markings as well. This assumption allows to simplify the verification of current-state opacity (as shown by Theorem 6.3). Moreover, it is useful when studying the case of initial-state opacity in Section 6.4.

We denote $S_B = S \cap \mathcal{M}_B$ the set of basis markings that are in secret S, and $ex(S_B) = ex(S) \cap \mathcal{M}_B = \mathcal{M}_B \setminus S_B$ the set of exposable basis markings.

Theorem 6.3. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic and S be a secret satisfying Assumption A3. The LPN G is current-state opaque wrt S iff $\forall w \in \mathcal{L}(G, M_0)$, $\mathcal{M}_b(w) \cap ex(S_B) \neq \emptyset$ holds.

Proof: ($\Rightarrow$) Let $M_b \in \mathcal{M}_b(w) \cap ex(S_B)$. Therefore, $M_b \in \mathcal{C}(w)$ and $\mathcal{C}(w) \cap ex(S) \neq \emptyset$, i.e., G is current-state opaque wrt S.

($\Leftarrow$) Assume G is opaque. Then $\forall w \in \mathcal{L}(G, M_0), \mathcal{C}(w) \cap ex(S) \neq \emptyset$, i.e., $\exists M \in \mathcal{C}(w) : M \in ex(S)$. According to Theorem 6.1, $\exists M_b \in \mathcal{M}_B \cap ex(S) : M \in \mathcal{U}(M_b)$, otherwise, Assumption A3 would be violated. Therefore, $\mathcal{M}_b(w) \cap ex(S_B) \neq \emptyset$. □

Example 6.9. Consider again the LPN in Fig. 6.1. Consider a secret $S = \{M_0, M_2, M_5, M_8, M_9\}$ that satisfies Assumption A3. Then we have $S_B = \{M_0, M_2\}$ and $ex(S_B) = \{M_1, M_3, M_4\}$. Based on the observer of the BRG in Fig. 6.8, $\forall w \in \mathcal{L}(G, M_0), \mathcal{M}_b(w) \cap ex(S_B) \neq \emptyset$, and therefore, the LPN is current-state opaque wrt S. $\diamond$

In other words, if Assumption A3 is satisfied, then current-state opacity can be verified by simply checking if each state of the current-state basis observer contains at least one basis marking which is exposable (rather than *weakly* exposable). This can be easily done by checking if $\mathcal{M}_b(w) \cap (R(N, M_0) \setminus S) = \emptyset$.

We finally point out that Theorem 6.3 could also be useful when the secret does not satisfy Assumption A3. Indeed, given an arbitrary system G and a secret S, the following proposition shows that we can always find another secret S'' which satisfies Assumption A3 and G has the same current-state opacity property wrt both S and S'', and hence Theorem 6.3 can be applied.

Proposition 6.5. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic, and $S \subseteq R(N, M_0)$ be a secret. G is current-state opaque wrt S iff G is current-state opaque wrt S'', where $S'' = S \setminus S'$ and $S' = wex(S) \cap S$.

Proof: Assume that G is current-state opaque wrt S''. Therefore, $\forall w \in \mathcal{L}(G, M_0), \mathcal{C}(w) \cap ex(S'') \neq \emptyset$. Suppose that G is not current-state opaque wrt S, i.e., $\exists w \in \mathcal{L}(G, M_0) : \mathcal{C}(w) \cap ex(S) = \emptyset$. Since $ex(S'') = ex(S) \cup S'$, we have $\mathcal{C}(w) \cap ex(S'') = (\mathcal{C}(w) \cap ex(S)) \cup (\mathcal{C}(w) \cap S') = \mathcal{C}(w) \cap S' \neq \emptyset$. Let $M \in \mathcal{C}(w) \cap S'$. Therefore, there exists a marking $M' \in \mathcal{U}(M) : M' \in ex(S)$, and thus $M' \in \mathcal{C}(w) \cap ex(S)$, i.e., G is opaque wrt S.

It is clear that $ex(S) \subseteq ex(S')$ and $ex(S) \subseteq ex(S'')$, since $S' \subseteq S$ and $S'' \subseteq S$. Furthermore, since G is current-state opaque wrt S, i.e., $\mathcal{C}(w) \cap ex(S) \neq \emptyset$, it holds $\mathcal{C}(w) \cap ex(S') \neq \emptyset$ and $\mathcal{C}(w) \cap ex(S'') \neq \emptyset$. Thus G is current-state opaque wrt both S' and S'', respectively. $\square$

Proposition 6.5 indicates that given a system G and a secret S, to verify if G is current-state opaque wrt S we can pretreat the secret S by simply removing all weakly exposable markings in S to get S'' that satisfies Assumption A3, and then verify if G is current-state opaque wrt S'' using Theorem 6.3.

Example 6.10. Consider again the LPN system in Fig.6.1. Let $S = \{M_1, M_2, M_5, M_8\}$. The secret does not satisfy Assumption A3 since M_1 is weakly exposable. The secret can be partitioned into $S = S' \cup S''$, where $S' = \{M_1\}$ and $S'' = \{M_2, M_5, M_8\}$. Therefore, $S''_B =$

$\{M_2\}$ and $ex(S''_B) = \{M_0, M_1, M_3, M_4\}$. Since $\forall w \in \mathcal{L}(G, M_0)$, $\mathcal{M}_b(w) \cap ex(S''_B) \neq \emptyset$ holds, the LPN is current-state opaque wrt S'', or equivalently, by Proposition 6.5, the LPN is current-state opaque wrt S. $\diamond$

6.3.4 Uncertainty on the Initial Marking

In this section we focus on the problem of verifying current-state opacity under the more general assumption that the intruder has only partial knowledge of the initial marking of the net. In more detail, we assume that the intruder simply knows that the initial marking M_0 belongs to a set $\mathcal{M}_0 \subseteq \mathcal{M}_B$, i.e., $M_0 \in \mathcal{M}_0$. Clearly, this is equivalent to assume that the set of possible initial markings for the intruder is $\hat{\mathcal{M}}_0 = \bigcup_{M_b \in \mathcal{M}_0} \mathcal{U}(M_b)$. Obviously, if a Petri net system is current-state opaque wrt a secret when the intruder knows the initial marking M_0, a fortiori it is current-state opaque when the intruder simply knows that the initial marking belongs to set $\hat{\mathcal{M}}_0$. In this subsection, we show that current-state opacity with the above ambiguity on the initial marking can be verified by simply modifying the current-state basis observer.

Given an observation w, we have defined $\mathcal{C}(w)$ as the set of markings consistent with w, assuming that M_0 is known. Now, we generalize this notion to a given set of initial markings $\hat{\mathcal{M}}_0$, and define

$$\hat{\mathcal{C}}(w) = \{M \in \mathbb{N}^m | \exists M' \in \hat{\mathcal{M}}_0, \exists \sigma \in T^* : M'[\sigma\rangle M \text{ and } \ell(\sigma) = w\},$$

i.e., $\hat{\mathcal{C}}(w)$ is the set of possible current markings estimated by the intruder when observing w. Clearly, if $M_0 \in \mathcal{M}_0$ it holds $\hat{\mathcal{C}}(w) \supseteq \mathcal{C}(w)$, hence the condition in Theorem 6.2 is no longer necessary. Namely, the system could be current-state opaque even if there exists $w \in \mathcal{L}(G, M_0)$ such that $\mathcal{M}_b(w) \cap wex(S) = \emptyset$. If the initial state of the current-state basis observer is initialized at $\mathcal{M}_0$ directly, words that would never be generated by the LPN, i.e., words in $\mathcal{L}(G, \hat{\mathcal{M}}_0) \setminus \mathcal{L}(G, M_0)$, will be generated by the observer. Note that $\mathcal{L}(G, \hat{\mathcal{M}}_0) = \mathcal{L}(G, \mathcal{M}_0)$. As a result, current-state opacity cannot be verified looking at the current-state basis observer. To restrict the language of the current-basis observer to the language of the LPN and separately denote estimations made on the basis of false initial markings $\mathcal{M}_0 \setminus \{M_0\}$ and estimations made on the basis of the real initial marking M_0, as formalized in the following definition, we introduce an extended observer which is the synthesis of two BRG observers, initialized at $\mathcal{M}_0 \setminus M_0$ and M_0, respectively.

Definition 6.8. Let $G = (N, M_0, \Sigma, \ell)$ be a bounded LPN system whose unobservable subnet is acyclic, $S \subseteq R(N, M_0)$ be a secret, $\mathcal{B}_c = (\tilde{\mathcal{M}}_B, \Sigma, \delta_c, (M_0, \alpha(M_0)))$ be the corresponding BRG for current-state opacity and $\mathcal{M}_0$ be the intruder's knowledge about the initial marking. The *extended observer* of the BRG is a DFA $\mathcal{V} = (Q, \Sigma, \delta, q_0)$, where $Q \subseteq 2^{\tilde{\mathcal{M}}_B} \times 2^{\tilde{\mathcal{M}}_B}$ and $q_0 = (\hat{\mathcal{X}}_0, \mathcal{X}_0)$ with $\hat{\mathcal{X}}_0 = \{(M, \alpha(M)) | M \in \mathcal{M}_0 \setminus \{M_0\}\}$ and $\mathcal{X}_0 = \{(M_0, \alpha(M_0))\}$. The transition function δ is defined as follows: for $e \in E$ and $(\hat{\mathcal{X}}_i, \mathcal{X}_i) \in Q$, if $\exists (M, \alpha(M)) \in \mathcal{X}_i : e$ is defined at $(M, \alpha(M))$, then $\delta((\hat{\mathcal{X}}_i, \mathcal{X}_i), e) = (\hat{\mathcal{X}}_j, \mathcal{X}_j))$, where $\hat{\mathcal{X}}_j = \{x' \in \tilde{\mathcal{M}}_B | \exists x \in \hat{\mathcal{X}}_i : x' \in \delta_c(x, e)\}$ and $\mathcal{X}_j = \{x' \in \tilde{\mathcal{M}}_B | \exists x \in \mathcal{X}_i : x' \in \delta_c(x, e)\}$. $\diamond$

In plain words, the extended observer characterizes the possible current markings estimated by the intruder. It is first initialized with the uncertainty of the initial marking. To verify current-state opacity we only need to consider the language $\mathcal{L}(G, M_0)$ generated by the LPN, since a word w which can only be generated by some false initial marking will not occur in the actual evolution of the system. The set $\mathcal{X}$ denotes the intruder's estimation with knowledge of the initial marking M_0, while the set $\hat{\mathcal{X}}$ denotes additional estimated markings introduced by uncertainties about the initial marking $\mathcal{M}_0 \setminus \{M_0\}$. Therefore, given an observation $w \in \mathcal{L}(G, M_0)$, the intruder's estimation of the current state is $\hat{\mathcal{X}} \cup \mathcal{X}$, where $\delta(q_0, w) = (\hat{\mathcal{X}}, \mathcal{X})$. As a particular case, if $\mathcal{M}_0 = \{M_0\}$, then the intruder knows exactly the initial marking and all the states of the corresponding extended observer have $\hat{\mathcal{X}} = \emptyset$. Therefore, the complexity of constructing the extended observer is $\mathcal{O}(4^{|\mathcal{M}_B|})$.

Theorem 6.4. Let $G = (N, M_0, \Sigma, \ell)$ be a bounded LPN system whose unobservable subnet is acyclic, $S \subseteq R(N, M_0)$ be a secret and $\mathcal{V} = (Q, \Sigma, \delta, q_0)$ be the corresponding extended observer. The LPN system is current-state opaque wrt S iff for all states $(\hat{\mathcal{X}}, \mathcal{X}) \in Q$, $\exists (M, \alpha(M)) \in \hat{\mathcal{X}} \cup \mathcal{X}$: $\alpha(M) = 1$.

Proof: Let $(\hat{\mathcal{X}}, \mathcal{X})$ be the state reachable by firing a sequence w, i.e., $\delta(q_0, w) = (\hat{\mathcal{X}}, \mathcal{X})$. According to Definition 6.8, the set $\hat{\mathcal{X}} \cup \mathcal{X}$ corresponding to $(\hat{\mathcal{X}}, \mathcal{X})$ is a subset of $\mathcal{M}_B \times \{0, 1\}$ whose markings belong to $\mathcal{C}(w)$. If $\exists (M, \alpha(M)) \in \hat{\mathcal{X}} \cup \mathcal{X}$: M is weakly exposable, for observation w there exists a marking $M' \in \mathcal{U}(M)$ such that $M' \in ex(S)$, i.e., $\mathcal{C}(w) \cap ex(S) \neq \emptyset$. Since this is true for all states of Q, i.e., for all observations $w \in \mathcal{L}(G, M_0)$, the LPN is current-state opaque. Assume there is a state $(\hat{\mathcal{X}}, \mathcal{X})$ of Q reachable by w and $\forall (M, \alpha(M)) \in \hat{\mathcal{X}} \cup \mathcal{X}$: M is not weakly exposable. Therefore, by Theorem 6.1, we have $\mathcal{C}(w) \cap ex(S) = \emptyset$. We conclude that the LPN is not current-state opaque. $\square$

Example 6.11. Let us consider the LPN in Fig. 6.9. The BRG of the LPN wrt secret $S = \{M \in \mathbb{N}^6 | M(p_3) + M(p_5) \leq 0\}$ is shown in Fig.6.10. Assume that the uncertainty about the

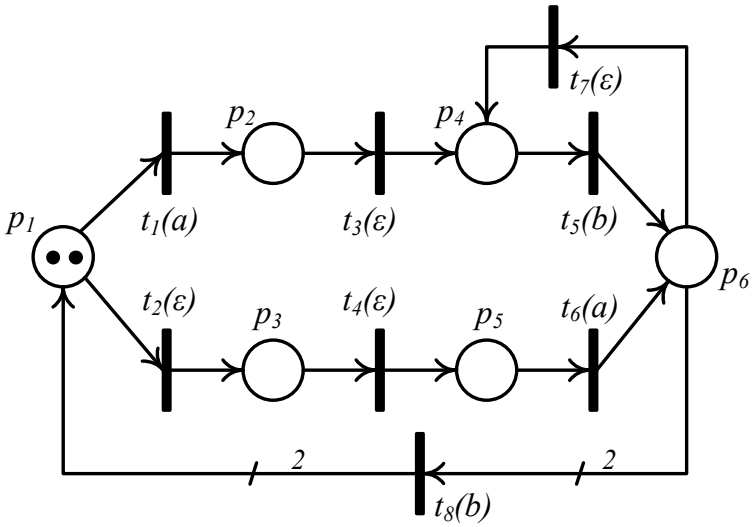

Fig. 6.9 LPN whose initial marking is not exactly known by the intruder.

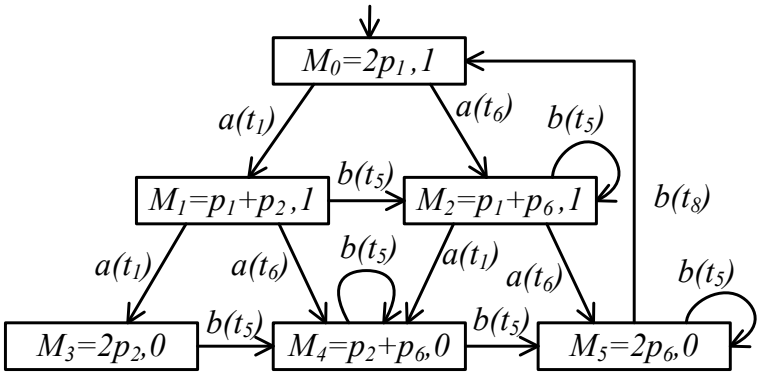

Fig. 6.10 BRG for CSO of the LPN in Example 6.11.

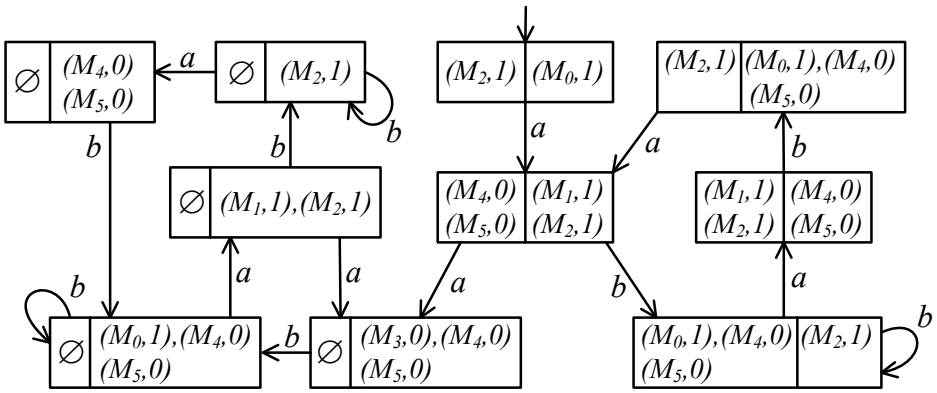

Fig. 6.11 The extended observer of the BRG in Fig. 6.10.

initial marking is $\mathcal{M}_0 = \{M_0, M_2\}$. The extended observer, as in Definition 6.8, is shown in Fig.6.11. There are states $(\emptyset; \{(M_3, 0), (M_4, 0), (M_5, 0)\})$ and $(\emptyset; \{(M_4, 0), (M_5, 0)\})$ where all basis markings in $\hat{\mathcal{X}} \cup \mathcal{X}$ satisfy $\alpha(\cdot) = 0$, and therefore, the LPN is not current-state opaque wrt S under $\mathcal{M}_0$. $\diamond$

6.4 Verification of Initial-State Opacity

According to the definition of ISO (see Definition 4.6), we have the following fact:

Fact 2: An LPN system is ISO wrt a secret $S \subseteq R(N, M_0)$ if and only if

$$\forall w \in \mathcal{L}(G, M_0) : \mathcal{I}(w) \not\subseteq S.$$

In other words, to verify ISO of an LPN, we need to check if $\mathcal{I}(w) \not\subseteq S$ holds for all $w \in \mathcal{L}(G, R(N, M_0))$. For a bounded LPN, this can be done by constructing the *initial-state estimator* (see Section 2.1.2) of its RG.

Let G be a bounded LPN and $\mathcal{A}^e = (\overline{X}, \Sigma, \delta_e, \overline{x}_0)$ be the initial-state estimator of the RG. According to the property of the initial-state estimator described in Section 2.1.2 of Chapter2, if $\exists w' \in \Sigma^*$ and $\overline{x} \in \overline{X}$ such that $\overline{x} = \delta_e(\overline{x}_0, w')$, then in the LPN we have $\mathcal{I}(w) = \overline{x}$, where w is the reverse of w'. Clearly, we have the following corollary.

Corollary 6.3. Given a bounded LPN $G = (N, M_0, \Sigma, \ell)$ and a secret $S \subseteq R(N, M_0)$, let $\mathcal{A}^e = (\overline{X}, \Sigma, \delta_e, \overline{x}_0)$ be the initial-state estimator of the RG. G is initial-state opaque wrt S iff $\forall \overline{x} \in \overline{X}$, $\overline{x} \cap ex(S) \neq \emptyset$ holds.

Therefore, by constructing the initial-state estimator of the RG, the complexity of verifying initial-state opacity in bounded Petri nets is $\mathcal{O}(2^{|R(N,M_0)|})$.

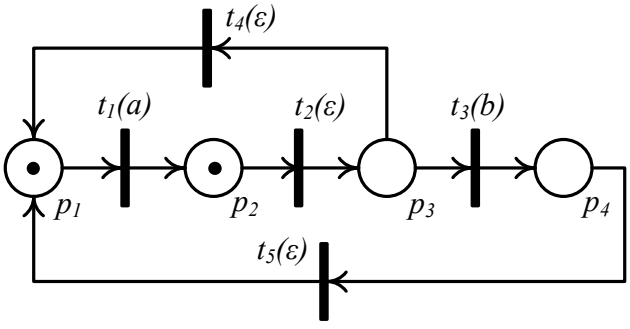

Fig. 6.12 LPN where t_3 is labeled by b.

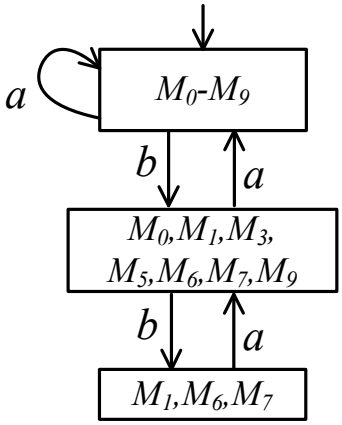

Fig. 6.13 Initial-state estimator of the RG.

Example 6.12. Consider the LPN in Fig. 6.12 whose only difference wrt Fig. 6.1 is the label assigned to transition t_3 (b rather than a). The initial-state estimator of its RG is shown in Fig. 6.13. Consider a secret $S_1 = \{M_1, M_5, M_6, M_7, M_9\}$ and an observation $w = bbaa$. The state reached by $w' = aabb$, the reverse of w, in the estimator is $\{M_1, M_6, M_7\}$, i.e., $\mathcal{I}(bbaa) = \{M_1, M_6, M_7\}$. Since $\mathcal{I}(bbaa) \cap ex(S_1) = \emptyset$, the LPN is not initial-state opaque wrt S_1.

Consider another secret $S_2 = \{M_1, M_7\}$. Then the LPN is initial-state opaque wrt S_2, since $\forall w \in \mathcal{L}(G, R(N, M_0)), \mathcal{I}(w) \cap ex(S_2) \neq \emptyset$. ◇

In the rest of this section an efficient approach to verifying initial-state opacity is proposed based on BRG analysis. Given an LPN $G = (N, M_0, \Sigma, \ell)$ and an observation $w \in \mathcal{L}(G, R(N, M_0))$, we denote by

$$\mathcal{I}_b(w) = \mathcal{I}(w) \cap \mathcal{M}_B$$

the set of basis markings generating w.

Proposition 6.6. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic and $S \subseteq R(N, M_0)$ be a secret. If G is initial-state opaque wrt S, then $\forall w \in \mathcal{L}(G, R(N, M_0)), \mathcal{I}_b(w) \cap wex(S) \neq \emptyset$ holds, where $wex(S)$ is the set of weakly exposable markings.

Proof: Since G is initial-state opaque wrt S, $\forall w \in \mathcal{L}(G, S)$, there exists an exposable marking $M \in ex(S)$ such that $w \in \mathcal{L}(G, M)$. Moreover, according to Theorem 6.1, $\exists M_b \in \mathcal{M}_B : M \in \mathcal{U}(M_b)$. Therefore, $w \in \mathcal{L}(G, M_b)$ and $M_b \in wex(S)$, i.e., $M_b \in \mathcal{I}_b(w)$. □

The following example shows that Proposition 6.6 provides a necessary but not sufficient condition for initial-state opacity.

Example 6.13. Consider the LPN system and the secret S_1 in Example 6.12. According to Table 6.1 (it applies to both the nets in Figs. 6.1 and 6.12), we have that $\forall M_b \in \mathcal{M}_B$, $\alpha(M_b) = 1$, i.e., all basis markings are weakly exposable. Clearly, $\forall w \in \mathcal{L}(G, R(N, M_0))$, $\mathcal{I}_b(w) \cap wex(S) \neq \emptyset$ holds. However, according to the result in Example 6.12, the LPN is not initial-state opaque wrt S_1. ◇

The reason why Proposition 6.6 is not a sufficient condition is that for an observation w, the possible initial markings that could generate w is generally not the union of all unobservable reach of the possible initial basis markings, i.e., $\mathcal{I}(w) \subseteq \bigcup_{M_b \in \mathcal{I}_b(w)} \mathcal{U}(M_b)$. Therefore $\mathcal{I}(w) \subseteq S$ does not imply that $\bigcup_{M_b \in \mathcal{I}_b(w)} \mathcal{U}(M_b) \subseteq S$. This is different from the case of the

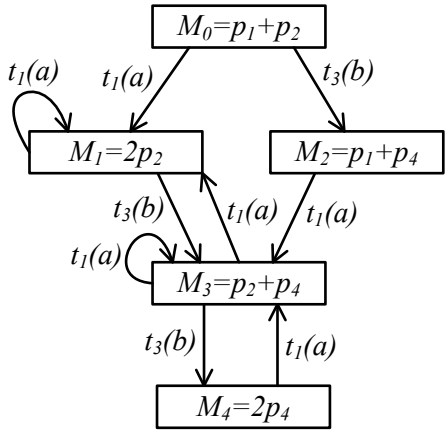

Fig. 6.14 BRG in Example 6.14.

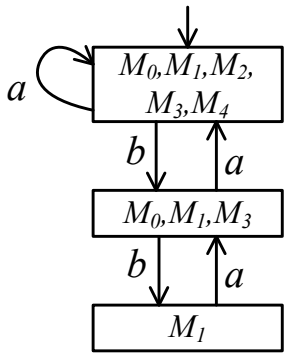

Fig. 6.15 Initial-state estimator of the BRG in Fig. 6.14.

current-state opacity problem since $\mathcal{C}(w) = \bigcup_{M_b \in \mathcal{M}_b(w)} \mathcal{U}(M_b)$. However, we show that if Assumption A3 is satisfied, initial-state opacity can be necessarily and sufficiently verified by checking if each $\mathcal{I}_b(w)$ contains at least one basis marking that does not belong to the secret.

Proposition 6.7. Let $G = (N, M_0, \Sigma, \ell)$ be an LPN system whose unobservable subnet is acyclic and S be a secret satisfying Assumption A3. G is initial-state opaque wrt S iff $\forall w \in \mathcal{L}(G, R(N, M_0)), \mathcal{I}_b(w) \cap ex(S_B) \neq \emptyset$ holds.

Proof: ($\Rightarrow$) Assume that $\forall w \in \mathcal{L}(G, R(N, M_0)), \mathcal{I}_b(w) \cap ex(S_B) \neq \emptyset$ holds. By $\mathcal{I}_b(w) \subseteq \mathcal{I}(w)$ and $ex(S_B) \subseteq ex(S)$, $\mathcal{I}(w) \cap ex(S) \neq \emptyset$, G is initial-state opaque wrt S.

($\Leftarrow$) Now assume that G is initial-state opaque wrt S. By Fact 2, $\forall w \in \mathcal{L}(G, R(N, M_0))$, we have $\mathcal{I}(w) \cap ex(S) \neq \emptyset$. Let $M \in \mathcal{I}(w) \cap ex(S)$. Under Assumption A3, there exists $M_b \in \mathcal{M}_B : M \in \mathcal{U}(M_b)$ and $M_b \in ex(S_B)$ (otherwise it would contradict $M \in ex(S)$). Therefore, $w \in \mathcal{L}(G, M_b)$ and $M_b \in \mathcal{I}(w) \cap ex(S_B)$, i.e., $\mathcal{I}_b(w) \cap ex(S_B) \neq \emptyset$. □

As a result, initial-state opacity of G can be verified by constructing the initial-state estimator of its BRG $\mathcal{B} = (\mathcal{M}_B, \Sigma, \delta)$ whose complexity is $\mathcal{O}(2^{|\mathcal{M}_B|})$. Since the size of the BRG will never be larger than the RG and it may be much smaller especially when unobservable transitions exist, the proposed approach is practically more efficient.

Example 6.14. Consider again the LPN system in Example 6.12. The BRG and the corresponding initial-state estimator are shown in Figs. 6.14 and 6.15. Let $S = \{M_0, M_2, M_5, M_8, M_9\}$ be the secret that satisfies Assumption A3. Then $S_B = \{M_0, M_2\}$ and $ex(S_B) = \{M_1, M_3, M_4\}$. According to Proposition 6.7, G is initial-state opaque wrt S since no state of the estimator either coincides with S_B or is strictly contained in it. ◇

Note that Assumption A3 is only necessary for the *only if* part of Proposition 6.7 as clarified by the following example.

Example 6.15. Consider the LPN system and secret S_2 in Example 6.12. All markings in S_2 are weakly exposable. We have $S_{2B} = \{M_1\}$ and $ex(S_{2B}) = \{M_0, M_2 - M_4\}$. Based on the initial-state estimator of the BRG shown in Fig. 6.15, we have $\mathcal{I}_b(bb) = \{M_1\}$, i.e., $\exists w : \mathcal{I}_b(w) \cap ex(S_B) = \emptyset$. However, according to the result in Example 6.12, the LPN is initial-state opaque wrt S_2. $\diamond$

6.4.1 Relaxation of Assumption A3

Different from the case discussed in Section 6.3.3, Assumption A3 cannot be relaxed in Proposition 6.7 by simply removing the weakly exposable markings from the secret. In this subsection, we propose a method to relax Assumption A3 by appropriately modifying the BRG definition. The new BRG is called *modified basis reachability graph* (MBRG).

Let us consider the case where Assumption A3 does not hold. Then S can be partitioned into $S' \cup S''$, where $S' = wex(S) \cap S \neq \emptyset$ and $S'' = S \setminus S'$ (Clearly, if Assumption A3 is satisfied, $S' = \emptyset$). The system may be initial-state opaque wrt S even if $\mathcal{I}_b(w) \subseteq S$, since there may exist some marking $M \in (\mathcal{I}(w) \setminus \mathcal{I}_b(w)) \cap ex(S)$. We write $\mathcal{Q} = \bigcup_{M \in S'} \mathcal{U}(M) \cap ex(S)$ to denote the unobservable reach of all markings in S'. The following proposition shows that to decide if the system is initial-state opaque, we need to check if $\mathcal{I}(w) \cap \mathcal{Q} \neq \emptyset$ holds

Proposition 6.8. Let w be an observation in an LPN system G whose unobservable subnet is acyclic, $S \subseteq R(N, M_0)$ be a secret, and $\mathcal{I}_b(w) \subseteq S$. Then $\mathcal{I}(w) \nsubseteq S$ iff $\mathcal{I}(w) \cap \mathcal{Q} \neq \emptyset$.

Proof: ($\Rightarrow$) Assume $\mathcal{I}(w) \cap \mathcal{Q} \neq \emptyset$. Since $\mathcal{Q} \subseteq ex(S)$, $\mathcal{I}(w) \cap ex(S) \neq \emptyset$, i.e., $\mathcal{I}(w) \nsubseteq S$.

($\Leftarrow$) Assume $\mathcal{I}(w) \nsubseteq S$. Since $\mathcal{I}_b(w) \subseteq S$, there exists a marking $M \in (\mathcal{I}(w) \setminus \mathcal{I}_b(w)) \cap ex(S)$. Let $M_b \in \mathcal{M}_B$ be the basis marking such that $M \in \mathcal{U}(M_b)$. Since $\mathcal{I}_b(w) \subseteq S$, we have $M_b \in S'$ and $M \in \mathcal{Q}$, i.e., $\mathcal{I}(w) \cap \mathcal{Q} \neq \emptyset$. $\square$

In simple words, when Assumption A3 is not satisfied, by checking if either $\mathcal{I}_b(w) \cap ex(S) \neq \emptyset$ or $\mathcal{I}(w) \cap \mathcal{Q} \neq \emptyset$ holds, initial-state opacity can be verified. Let $\mathcal{Q}_{min} \subseteq \mathcal{Q}$ be the subset of $\mathcal{Q}$ with the minimal cardinality satisfying the following property: for any $M' \in \mathcal{Q}$, there exists $M \in \mathcal{Q}_{min}$ such that $M' \in \mathcal{U}(M)$. Obviously $\mathcal{Q}_{min}$ is unique.

Proposition 6.9. Let w be an observation in a bounded LPN system G whose unobservable subnet is acyclic and $S \subseteq R(N, M_0)$ be a secret. Then $\mathcal{I}(w) \cap \mathcal{Q} \neq \emptyset$ iff $\mathcal{I}(w) \cap \mathcal{Q}_{min} \neq \emptyset$.

Proof: ($\Rightarrow$) Assume $\mathcal{I}(w) \cap \mathcal{Q}_{min} \neq \emptyset$. Since $\mathcal{Q}_{min} \subseteq \mathcal{Q}$, $\mathcal{I}(w) \cap \mathcal{Q} \neq \emptyset$.

($\Leftarrow$) Assume $\mathcal{I}(w) \cap \mathcal{Q} \neq \emptyset$. Let $M' \in \mathcal{I}(w) \cap \mathcal{Q}$. There exists $M \in \mathcal{Q}_{min}$ such that $M' \in \mathcal{U}(M)$. Therefore, $M \in \mathcal{I}(w)$, i.e., $\mathcal{I}(w) \cap \mathcal{Q}_{min} \neq \emptyset$. □

Proposition 6.9 shows that we do not need to consider all markings in $\mathcal{Q}$ but only a minimal subset of them. Given a bounded LPN system whose unobservable subnet is acyclic, we propose Algorithm 6 to compute $\mathcal{Q}_{min}$. Once $\mathcal{Q}_{min}$ is obtained, a method to verify initial-state opacity by using the MBRG is proposed.

Algorithm 6 Computation of $\mathcal{Q}_{min}$

Input: A bounded LPN system $G = (N, M_0, \Sigma, \ell)$ whose unobservable subnet N' is acyclic, and a secret S.

Output: $\mathcal{Q}_{min}$

1: $\mathcal{Q}_{min} := \emptyset$;
2: Compute S', the set of weakly exposable markings in S;
3: **while** $S' \neq \emptyset$, **do**
4: select a marking $M \in S'$;
5: construct the reachability graph of $\langle N', M\rangle$, denoted as $\mathcal{R}(N', M) = (\mathcal{U}(M), T_u, \delta, M)$;
6: $\mathcal{Q}_{temp} := \mathcal{U}(M) \cap ex(S)$.
7: **for all** $M_j \in \mathcal{Q}_{temp}$, **do**
8: **if** $\nexists M_i \in \mathcal{Q}_{temp} : M_j \in \delta(M_i, \sigma_u)$ is defined, where $\sigma_u \in T_u^*$, **then**
9: $\mathcal{Q}_{min} := \mathcal{Q}_{min} \cup \{M_j\}$;
10: **end if**
11: **end for**
12: $S' := S' \setminus \mathcal{U}(M)$;
13: **end while**

In Algorithm 6, $\mathcal{Q}_{min}$ is initialized at the empty set. Given a marking $M \in S'$, the reachability graph of $\langle N', M\rangle$ is denoted as an automaton $\mathcal{R}(N', M) = (\mathcal{U}(M), T_u, \delta, M)$: the state space of $\mathcal{R}(N', M)$ is the unobservable reach of M, the initial state is M, and the event set is the set of the unobservable transitions T_u. Since N' is acyclic, the reachable markings can be computed by solving the state equation $M' = M + C_u \cdot y$, where $y \in \mathbb{N}^{n_u}$, and there is no cycle in the reachability graph $\mathcal{R}(N', M)$. We compute the set of exposable markings that are initial vertices of paths in $\mathcal{R}(N', M)$ (Steps 7-11). Finally, since some markings in S' can be reached from some existing markings in $\mathcal{Q}_{min}$, they are removed from S' to further reduce the computation load in forthcoming iterations.

Example 6.16. Consider again the LPN in Example 6.12 and $S = \{M_1, M_7\}$. All markings in S are weakly exposable. The reachability graph of $\langle N', M_1\rangle$ is shown in Fig. 6.16, and secret markings are in shadowed boxes. We have $S' = \{M_1, M_7\}$. By Algorithm 6, after

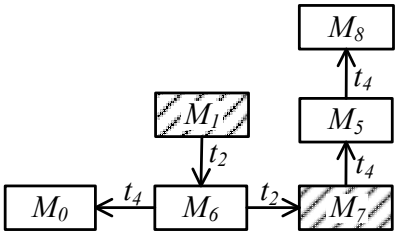

Fig. 6.16 Reachability Graph $\mathcal{R}(N', M_1)$.

the first iteration, $\mathcal{Q}_{temp} = \{M_0, M_5, M_6, M_8\}$ and $\mathcal{Q}_{min} = \{M_6\}$ since one can readily verify that all nonsecret markings M_0, M_5, and M_8 in $\mathcal{U}(M_1)$ can be reached from M_6 by firing unobservable transitions. Furthermore, M_7 is removed from S' by Step 12 since $M_7 \in \mathcal{U}(M_1)$ and it is not necessary to check the unobservable reach of M_7. Then Algorithm 6 outputs $\mathcal{Q}_{min} = \{M_6\}$. $\diamond$

In the following, Algorithm 7 is presented to construct the modified BRG (MBRG) $\mathcal{B}' = (\mathcal{M}_{B'}, \Sigma, \delta')$ of a given bounded LPN system whose unobservable subnet is acyclic. The method to construct the MBRG is analogous to the method to construct BRG, however, the nodes are initialized by markings in $\{M_0\} \cup \mathcal{Q}_{min}$. Although the MBRG is larger than the BRG, the MBRG is still much smaller than RG. We use $\mathcal{M}_{B'}$ to denote the *extended basis markings* that appear in the MBRG. Correspondingly, we denote by $\mathcal{I}_{b'}(w) = \mathcal{I}(w) \cap \mathcal{M}_{B'}$ the set of markings in $\mathcal{M}_{B'}$ generating w, $S_{B'} = S \cap \mathcal{M}_{B'}$ the set of markings in $\mathcal{M}_{B'}$ that belong to the secret, and $ex(S_{B'}) = ex(S) \cap \mathcal{M}_{B'}$ the set of markings in $\mathcal{M}_{B'}$ that do not belong to the secret.

Proposition 6.10. Let G be an LPN system whose unobservable subnet is acyclic, and $S \subseteq R(N, M_0)$ be a secret. G is initial-state opaque wrt S iff $\forall w \in \mathcal{L}(G, R(N, M_0))$, $\mathcal{I}_{b'}(w) \cap ex(S_{B'}) \neq \emptyset$ holds.

Proof: ($\Rightarrow$) Assume that $\forall w \in \mathcal{L}(G, R(N, M_0)), \mathcal{I}_{b'}(w) \cap ex(S_{B'}) \neq \emptyset$. Since $\mathcal{I}_{b'}(w) \subseteq \mathcal{I}(w)$ and $ex(S_{B'}) \subseteq ex(S)$, $\mathcal{I}(w) \cap ex(S) \neq \emptyset$ holds and, by Fact 2, G is initial-state opaque wrt S.

($\Leftarrow$) Now assume that G is initial-state opaque wrt S. According to Fact 2, $\forall w \in \mathcal{L}(G, R(N, M_0))$, we have $\mathcal{I}(w) \cap ex(S) \neq \emptyset$, i.e., $\exists M \in \mathcal{I}(w) \cap ex(S)$. Since M must be in the unobservable reach of a basis marking M_b. If $M_b \in ex(S)$ then the proof is concluded. If $M_b \in S$, then M_b is weakly exposable, i.e., $M_b \in S'$. By Algorithm 6 there must exist a marking $M' \in \mathcal{Q}_{min} \subseteq \mathcal{M}_{B'}$ such that M is reachable from M' by firing only unobservable transitions, which indicates that $M' \in \mathcal{I}_{b'}(w) \cap ex(S_{B'})$. $\square$

Algorithm 7 Construction of the MBRG

Input: A bounded LPN system $G = (N, M_0, \Sigma, \ell)$ whose unobservable subnet N' is acyclic, and a secret S.

Output: The modified BRG $\mathcal{B}' = (\mathcal{M}_{B'}, \Sigma, \delta')$.

1: Construct the BRG $\mathcal{B} = (\mathcal{M}_B, \Sigma, \delta, M_0)$ by using Algorithm 5.
2: Compute set $\mathcal{Q}_{min}$ by using Algorithm 6.
3: $\mathcal{M}_{B'} := \mathcal{M}_B \cup \mathcal{Q}_{min}$, $\delta' := \delta$.
4: Tag all $M \in \mathcal{M}_B$ "old".
5: **while** states in $\mathcal{M}_{B'}$ with no tag exist, **do**
6: select a state $M \in \mathcal{M}_{B'}$ with no tag;
7: **for all** t s.t. $\ell(t) \in \Sigma$ and $Y_{min}(M, t) \neq \emptyset$, **do**
8: **for all** $y_u \in Y_{min}(M, t)$, **do**
9: $M' := M + C_u \cdot y_u + C(\cdot, t)$;
10: **if** $M' \notin \mathcal{M}_B$, **then**
11: $\mathcal{M}_{B'} := \mathcal{M}_{B'} \cup \{M'\}$;
12: assign no tag to M';
13: **end if**
14: $\delta'(M, \ell(t)) := \delta'(M, \ell(t)) \cup \{M'\}$;
15: **end for**
16: tag node M "old";
17: **end for**
18: **end while**
19: Remove all tags.

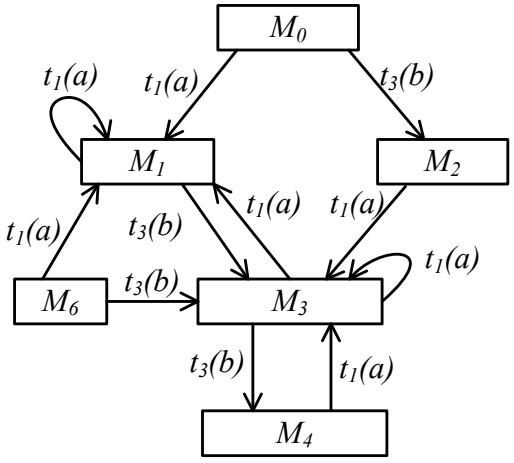

Fig. 6.17 MBRG in Example 6.17.

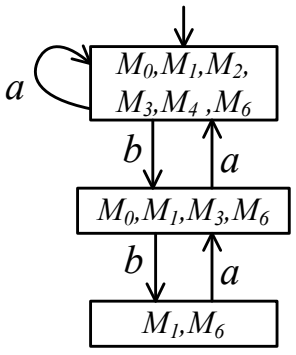

Fig. 6.18 Initial-state estimator of the MBRG in Fig. 6.17.

Proposition 6.10 shows that if Assumption A3 is not satisfied, by constructing the initial-state estimator of the MBRG, initial-state opacity of G can be verified. In this case, the complexity increases to $\mathcal{O}(2^{|\mathcal{M}_{B'}|})$.

Example 6.17. Since secret $S = \{M_1, M_7\}$ does not satisfy Assumption A3, we construct its MBRG (shown in Fig. 6.17) by Algorithm 7. We have $\mathcal{M}_{B'} = \{M_0 - M_4, M_6\}$, $S_{B'} = \{M_1\}$

and $ex(S_{B'}) = \{M_0, M_2 - M_4, M_6\}$. The corresponding initial-state estimator is shown in Fig. 6.18. Since, $\forall w \in \mathcal{L}(G, R(N, M_0)), \mathcal{I}_{b'}(w) \cap ex(S_{B'}) \neq \emptyset$ holds, the LPN is initial-state opaque wrt S. $\diamond$

6.5 Numerical Examples

To compare the approaches of using BRG and RG to verify the state-based opacity properties, a series of numerical examples are presented. Based on the proposed approaches in this work, we developed a MATLAB tool [71] to compute the BRG, the current-state basis observer, the initial-state estimator, and to determine current-state opacity of a bounded LPN. In the following, numerical results are obtained by using the tool.

We still consider the simple LPN G in Fig. 6.1 but the initial marking in place p_2 is a parameter $k \in \{1, 2, \cdots\}$. Therefore, here we consider not a single LPN but a family of nets parameterized by the initial marking. Based on the structure of the LPN, the number of its reachable markings is

$$|R(N, M_0)| = \frac{1}{6}(k+4)(k+3)(k+2). \tag{6-2}$$

We still let t_1 and t_3 be the observable transitions. Then the number of basis markings is

$$|\mathcal{M}_B| = 2k + 3. \tag{6-3}$$

Based on Eqs. (6-2) and (6-3), Fig. 6.19 shows the variation of $|R(N, M_0)|$ and $|\mathcal{M}_B|$ with respect to k. The numerical values for some specific k's together with the computational times are reported in Table 6.2, where Columns 2 and 4 illustrate the number of reachable markings $|R(N, M_0)|$ and basis markings $|\mathcal{M}_B|$, respectively. The corresponding time costs are presented in Columns 3 and 5, respectively. The table shows that when the initial marking of p_2 is larger than or equal to 60, the RG cannot be computed within 8 hours and we use "o.t." to denote the computation is out of time. On the contrary, the BRG can still be constructed in a short time.

For the verification of current-state opacity, let $\ell(t_1) = a$, $\ell(t_3) = a$, and $S = \{M \in \mathbb{N}^4 | M(p_1) + M(p_4) \geq 2\}$, i.e., $W = [\ -1 \quad 0 \quad 0 \quad -1\ ]$ and $K = -2$. For the verification of initial-state opacity, let $\ell(t_1) = a$, $\ell(t_3) = b$. Results are summarized in Tables 6.3 and 6.4, respectively. Columns 2 and 4 present the numbers of states $|\mathcal{X}_{or}|$ and $|\mathcal{X}_{ob}|$ (resp. $|\mathcal{X}_{er}|$ and $|\mathcal{X}_{eb}|$) corresponding to the observers (resp. estimators) of the RG and the BRG. The computation time T-or and T-ob (resp. T-er and T-eb) are shown in Columns 3 and 5, respectively. Note that the computational time for the observer and estimator does not

Table 6.2 Number of (basis) markings and time cost

k	$\|R(N, M_0)\|$	T-r	$\|\mathcal{M}_B\|$	T-b
8	220	7.2×10^{-1}	19	4.1×10^{-2}
10	364	2.1×10^{0}	23	5.0×10^{-2}
20	2024	6.3×10^{1}	43	8.0×10^{-2}
40	13244	1.1×10^{3}	83	1.1×10^{-1}
60	o.t.	o.t.	123	3.6×10^{-1}
80	o.t.	o.t.	163	5.3×10^{-1}
100	o.t.	o.t.	203	7.6×10^{-1}
120	o.t.	o.t.	243	9.7×10^{-1}

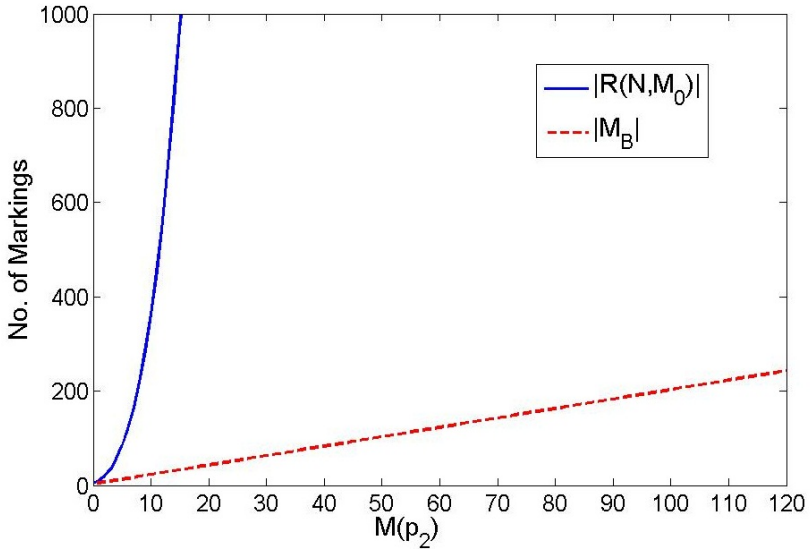

Fig. 6.19 The sizes of $|R(N, M_0)|$ and $|\mathcal{M}_B|$ changing with k.

increase fast with respect to k. However, the observer and the estimator of the RG cannot be constructed since the RG is not obtained for $k \geq 60$.

From Table 6.3, we notice that the number of states of the observer computed using RG, when computable, is identical to the number of states of the observer relative to the BRG. It can be easily proved that this is a general result validating the effectiveness of the proposed result. Same conclusions can be drawn with regard to the estimator. As a result, we conclude that the proposed approaches are practically efficient especially for large-size Petri nets. The reader can use the MATLAB tool we have developed, which is available on the web [71] to test the proposed approach on other nets.

Two remarks should be done concerning the above numerical examples. The first one relates to initial-state opacity verification, and the other is about the MBRG. When the initial marking (i.e., the value of k) changes, a given secret may not satisfy Assumption A3. Therefore, for initial-state opacity we cannot provide results as a function of parameter k while keeping the secret constant. In simple words, a column analogous to the last column of Table

Table 6.3 Number of states of the observers and the time cost

k	$\lvert\mathcal{X}_{or}\rvert$	T-or	$\lvert\mathcal{X}_{ob}\rvert$	T-ob	CSO
8	11	2.0×10^0	11	1.6×10^{-2}	Y
10	13	4.2×10^0	13	2.2×10^{-2}	Y
20	23	5.5×10^1	23	6.8×10^{-2}	Y
40	43	3.5×10^3	43	2.1×10^{-1}	Y
60	o.t.	o.t.	63	4.5×10^{-1}	Y
80	o.t.	o.t.	83	7.7×10^{-1}	Y
100	o.t.	o.t.	103	8.7×10^{-1}	Y
120	o.t.	o.t.	123	1.7×10^0	Y

Table 6.4 Number of states of the estimators and the time cost

k	$\lvert\mathcal{X}_{er}\rvert$	T-er	$\lvert\mathcal{X}_{eb}\rvert$	T-eb
8	10	1.8×10^0	10	1.1×10^{-1}
10	12	3.6×10^0	12	1.1×10^{-1}
20	22	6.0×10^1	22	2.8×10^{-1}
40	42	3.7×10^3	42	1.7×10^0
60	o.t.	o.t.	62	3.7×10^0
80	o.t.	o.t.	82	6.5×10^0
100	o.t.	o.t.	102	9.8×10^0
120	o.t.	o.t.	122	1.4×10^1

6.3 cannot be obtained. It would be also interesting to compare the size of the MBRG and the RG for different values of k as we did for the BRG. However, this cannot be done since the structure of MBRG depends not only on the initial marking but also on the secret.

6.6 Verification of Strict Language Opacity

The verification of language-based opacity in Petri nets has never been discussed before, the only way to solve the problem in bounded Petri nets was to construct the reachability graph and apply the approach in [6]. Corollary 4.1 implies that if a secret $S \subseteq T^*$ has the normality property, i.e., there exists a subset $\hat{T} \subseteq T$ such that S is normal wrt $L(N, M_0)$ and $\hat{T}$, language opacity wrt S can be verified by strict language opacity wrt $\hat{T}$ and $P_{\hat{T}}(S)$. However, given a secret $S \subseteq T^*$, finding a set $\hat{T}$ that guarantees its normality is still an open problem. Therefore, the remaining of this chapter is focused on verifying strict language opacity in a bounded LPN system $G = (N, M_0, \Sigma, \ell)$. Before formalizing the problem addressed in this section, we first introduce the following assumptions:

A1) The T_u-induced subnet of G is acyclic.

A2) The set of transitions of which the intruder cares is the set of observable transitions, i.e., $\hat{T} = T_o$.

A3) The secret is the set of all firable transition sequences in a bounded labeled Petri net system whose set of transitions is T_o, but excluding ε.

These assumptions would bring some properties on which we could build our algorithm for verifying strict language opacity. In practice, the computational load can be greatly reduced as shown in Section 6.6.2, since by Assumptions A1 and A2 the order of the unobservable transitions can be abstracted using minimal explanations and basis markings. Note that there is no assumption on the labeling function of the system. Namely, the same label (including ε) can be assigned to different transitions. Assumption A3 indicates that the secret can be described by the generated language of a bounded free-labeled Petri net but excluding the empty word. If ε is in the secret, the system is not strictly language opaque since initially it is at the state in which no observable transition has fired and the intruder can conclude that without ambiguity. It is true that only a special class of secrets satisfy Assumption A3 and thus the application of the proposed approach is not general. However, Assumption 3 leads to a computationally efficient verification procedure. Moreover, in practice we believe that these assumptions, even if quite restrictive, allow to represent some real problems.

We denote $G^S = (N^S, M_0^S, T^S, \ell^S)$, where $T^S \subseteq T_o$, the labeled Petri net system describing the secret, i.e., $S = L(N^S, M_0^S) \setminus \{\varepsilon\}$. The labeling function of G^S is identical to that of G, i.e., given a transition $t \in T^S$, $\ell^S(t) = \ell(t)$. The problem is stated as follows.

Problem Statement: Given a bounded LPN system G and a bounded LPN system G^S describing the secret $S = L(N^S, M_0^S) \setminus \{\varepsilon\} \subseteq T_o^*$ satisfying Assumptions A1 to A3, determine whether G is strictly language opaque wrt T_o and S or not.

6.6.1 Construction of the Verifier

In this section we introduce a structure, called *verifier*, to efficiently verify strict language opacity.

If the plant G and the net G^S describing the secret are bounded, the verifier is a DFA, denoted as $\mathcal{V} = (X, T^S, \delta, x_0)$. A state $x \in X$ of $\mathcal{V}$ is a 3-tuple $(M^S, \mathcal{M}^G, \mathcal{M})$, where M^S is a marking of G^S, $\mathcal{M}^G$ and $\mathcal{M}$ are subsets of basis markings $\mathcal{M}_B$ in G, i.e., $M^S \in R(N^S, M_0^S)$, $\mathcal{M}^G \subseteq \mathcal{M}_B$ and $\mathcal{M} \subseteq \mathcal{M}_B$. The i-th element (for $i = 1, 2, 3$) of x is denoted as $x(i)$. The set of events is the set of transitions of G^S. The initial state of the verifier is $x_0 = (M_0^S, \{M_0\}, \emptyset)$. Given a set T of transitions, a label $e \in \Sigma \cup \{\varepsilon\}$ and a marking M, we denote

$$T(M) = \{t \in T | M[t\rangle\}$$

the *set of transitions enabled at* M; we denote

$$T_e = \{t \in T | \ell(t) = e\}$$

the *set of transitions whose label are* e; finally, we denote

$$T_e(M) = T(M) \cap T_e$$

the *set of transitions enabled at* M *and labeled with* e. Algorithm 8 illustrates the construction of the verifier.

Given a state x in the verifier such that $\delta(x_0, \sigma) = x$, $x(1)$ is the marking in G^S reachable from M_0^S by firing σ, i.e., $M_0^S[\sigma\rangle x(1)$; $x(2)$ is the set of basis markings in G that are reachable by firing a transition sequence σ' whose projection on T_o belongs to S, i.e., $P_{T_o}(\sigma') \in S$; finally $x(3)$ is the set of basis markings in G that are reachable by firing a transition sequence σ'' whose projection on T_o is different from σ but generates the same observation, i.e., $P_{T_o}(\sigma'') \neq \sigma$ and $\ell(\sigma'') = \ell(\sigma)$. More precisely, given two states $x_1 = (M_1^S, \mathcal{M}_1^G, \mathcal{M}_1)$, $x_2 = (M_2^S, \mathcal{M}_2^G, \mathcal{M}_2)$ and an event $t \in T^S$ of $\mathcal{V}$, $\delta(x_1, t) = x_2$ implies that M_2^S is the marking reachable from M_1^S by firing t in G^S (Step 6), $\mathcal{M}_2^G$ is the set of basis markings reachable from markings in $\mathcal{M}_1^G$ by firing $t \in T^S(x(1))$ and the corresponding minimal explanations in G (Steps 7 to 14), and $\mathcal{M}_2$ is the union of two sets: the set of markings reachable from markings in $\mathcal{M}_1^G$ by firing transitions in $t' \in T_e \setminus T_e^S(x(1))$ and the corresponding minimal explanations (Steps 19 to 26) in G, and the set of markings reachable from $\mathcal{M}_1$ by firing transitions in T_e and the corresponding minimal explanations (Steps 27 to 34) in G, where $e = \ell(t)$ and $T_e^S(x(1))$ is the set of transitions labeled with e and enabled at marking $x(1)$ in G^S. If such a transition t is never enabled in G, then a new node would not be created (Steps 15 to 17).

The main idea behind Algorithm 8 is to compute a sort of parallel composition between G^S and G, where synchronization is performed wrt T_o. As a result, the generated language of $\mathcal{V}$ is equal to $P_{T_o}(L(N, M_0)) \cap L(N^S, M_0^S)$. Moreover, this enables one to understand if to a transition sequence in the secret it corresponds a transition sequence in the system such that its projection on T_o is not in the secret, and it generates the same observation. The properties of the verifier are formally presented as follows.

Proposition 6.11. Let $G = (N, M_0, \Sigma, \ell)$ be a bounded LPN satisfying Assumption A1, and $G^S = (N^S, M_0^S, T^S, \ell^S)$ be the LPN describing the secret S and satisfying Assumptions A2 and A3. Let $\mathcal{V} = (X, T^S, \delta, x_0)$ be the verifier constructed by using Algorithm 8. Given a

Algorithm 8 Construction of the verifier

Input: A bounded LPN system $G = (N, M_0, \Sigma, \ell)$ and a bounded LPN system $G^S = (N^S, M_0^S, \Sigma, \ell)$.

Output: Verifier $\mathcal{V} = (X, T^S, \delta, x_0)$

$x_0 := (M_0^S, \{M_0\}, \emptyset)$;
$X := \{x_0\}$;
for all $x \in X$ with no tag, **do**
  for all $t \in T^S(x(1))$, **do**
    $\mathcal{M}^G := \emptyset$, $\mathcal{M} := \emptyset$;
    $M^S := x(1) + C^S(\cdot, t)$;
    for all $M \in x(2)$ **do**
      for all $t \in T^S(x(1))$ **do**
        for all $y_u \in Y_{min}(M, t)$ **do**
          $M' := M + C_u \cdot y_u + C(\cdot, t)$;
          $\mathcal{M}^G := \mathcal{M}^G \cup \{M'\}$;
        end for
      end for
    end for
    if $\mathcal{M}^G = \emptyset$, **then**
      Break;
    end if
    $e := \ell(t)$;
    for all $M \in x(2)$, **do**
      for all $t' \in T_e \setminus T_e^S(x(1))$, **do**
        for all $y_u \in Y_{min}(M, t')$, **do**
          $M' := M + C_u \cdot y_u + C(\cdot, t')$
          $\mathcal{M} := \mathcal{M} \cup \{M'\}$;
        end for
      end for
    end for
    for all $M \in x(3)$, **do**
      for all $t' \in T_e$, **do**
        for all $y_u \in Y_{min}(M, t')$, **do**
          $M' := M + C_u \cdot y_u + C(\cdot, t')$;
          $\mathcal{M} := \mathcal{M} \cup \{M'\}$;
        end for
      end for
    end for
    if $(M^S, \mathcal{M}^G, \mathcal{M}) \notin X$, **then**
      $X := X \cup \{(M^S, \mathcal{M}^G, \mathcal{M})\}$;
    end if
    $\delta(x, t) := (M^S, \mathcal{M}^G, \mathcal{M})$;
  end for
  tag x "checked";
end for

state $x \in X$ and $\sigma \in L(\mathcal{V})$ such that $\delta(x_0, \sigma) = x$, the following implication holds:

$$x(2) \neq \emptyset \Leftrightarrow \exists \sigma' \in L(N, M_0) : P_{T_o}(\sigma') \in S.$$

Proof: Let $\delta(x_0, \sigma) = x$ and $\sigma = t_{i1}t_{i2}\cdots t_{ik} \in T^{S*}$. According to Algorithm 8, $x(2)$ is the set of markings reachable from M_0 by firing the sequence $\hat{\sigma} = \sigma_{u1}t'_{i1}\sigma_{u2}t'_{i2}\cdots\sigma_{uk}t'_{ik}$ in G, where σ_{uj} is a minimal explanation of t'_{ij}, $t'_{ij} \in \hat{T}$ and $\ell(t'_{ij}) = \ell(t_ij)$. Therefore, $P_{T_o}(\hat{\sigma}) \in S$. By Corollary 6.1, a sequence $\sigma' = \sigma'_{u1}t'_{i1}\sigma'_{u2}t'_{i2}\cdots\sigma'_{uk}t'_{ik}$ such that $M_0[\sigma'\rangle$ and $P_{T_o}(\sigma') = P_{T_o}(\hat{\sigma})$ exists (where $\sigma'_{ui} \in T_u^*$) if and only if $x(2) \neq \emptyset$. □

Proposition 6.12. Let $G = (N, M_0, \Sigma, \ell)$ be a bounded LPN system satisfying Assumption A1, and $G^S = (N^S, M_0^S, T^S, \ell^S)$ be the LPN system describing the secret S and satisfying Assumptions A2 and A3. Let $\mathcal{V} = (X, T^S, \delta, x_0)$ be the verifier constructed by using Algorithm 8. Given a state $x \in X \setminus \{x_0\}$ and $\sigma \in L(\mathcal{V})$ such that $\delta(x_0, \sigma) = x$, the following implication holds:

$$x(3) \neq \emptyset \Leftrightarrow \exists \sigma' \in L(N, M_0) : P_{T_o}(\sigma') \notin S \wedge \ell(\sigma') = \ell(\sigma).$$

Proof: This is proven by induction.

(Basis step) For $\sigma \in L(\mathcal{V})$ with length 1. Let $\delta(x_0, t) = x$ and $e = \ell(t)$. Since $x_0(3) = \emptyset$, if and only if $x(3) \neq \emptyset$, there exists a sequence $\sigma_u t'$ such that $M_0[\sigma_u t'\rangle M \in x(3)$, where $t' \in T_e \setminus T_e^S(x(2))$, and $\sigma_u \in \Sigma_{min}(M_0, t')$. Namely, $P_{T_o}(\sigma_u t') \notin S$ and $\ell(t') = \ell(t)$.

(Inductive step) Assume that for $\sigma_k \in L(\mathcal{V})$ with length k, the result is valid. We prove that it is also true for $\sigma_{k+1} \in L(\mathcal{V})$ with length $k+1$. Let $\sigma_{k+1} = \sigma_k t$ and $e = \ell(t)$.

Consider $\delta(x_0, \sigma_k) = x_1$ and $\delta(x_1, t) = x_2$. According to Algorithm 8 and Corollary 6.1, $x_2(3) \neq \emptyset$ if and only if one of the following conditions holds:

1. there exists $M \in x_1(2)$ such that $M[\sigma_u t'\rangle$, where $t' \in T_e \setminus T_e^S(x_1(1))$ and $\sigma_u \in \Sigma_{min}(M, t')$;
2. there exists $M \in x_1(3)$ such that $M[\sigma_u t'\rangle$, where $t' \in T_e$ and $\sigma_u \in \Sigma_{min}(M, t')$.

Assume condition 1) holds. Since $x_1(2) \neq \emptyset$, by Proposition 6.11, there exists $\sigma \in L(N, M_0)$ such that $P_{T_o}(\sigma) \in S$, $\ell(\sigma) = \ell(\sigma_k)$ and $M_0[\sigma\rangle M$. Since $t' \in T_e \setminus T_e^S(x_1(1))$, $P_{T_o}(\sigma\sigma_u t') \notin S$ but $\ell(\sigma\sigma_u t') = \ell(\sigma_{k+1})$.

Assume condition 2) holds. Since $x_1(3) \neq \emptyset$, there exists $\sigma \in L(N, M_0)$ such that $P_{T_o}(\sigma) \notin S$ and $M_0[\sigma\rangle M$. Therefore, $P_{T_o}(\sigma\sigma_u t') \notin S$ but $\ell(\sigma\sigma_u t') = \ell(\sigma_{k+1})$. Thus, this concludes the proof. □

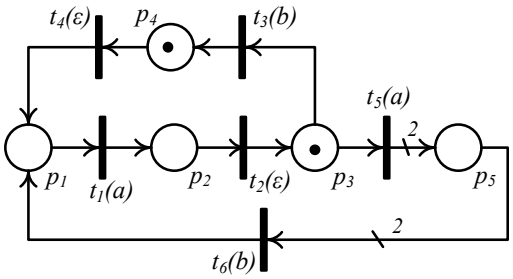

Fig. 6.20 LPN system whose T_u-induced net is acyclic in Example 6.18.

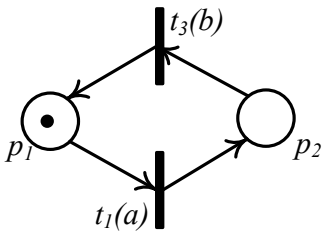

Fig. 6.21 LPN system that models the secret in Example 6.18.

Theorem 6.5. Let $G = (N, M_0, \Sigma, \ell)$ be a bounded LPN system satisfying Assumption A1, and $G^S = (N^S, M_0^S, T^S, \ell^S)$ be the LPN system describing the secret S and satisfying Assumptions A2 and A3. Let $\mathcal{V} = (X, T^S, \delta, x_0)$ be the verifier constructed by using Algorithm 8. G is strictly language opaque wrt T_o and S, if and only if $\forall x \in X \setminus \{x_0\}, x(3) \neq \emptyset$ holds.

Proof: Follows from Propositions 6.11 and 6.12. □

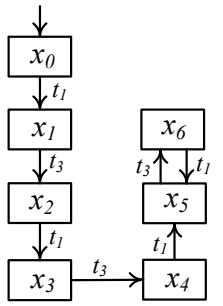

Fig. 6.22 Verifier constructed in Example 6.18.

Table 6.5 States of the verifier in Fig. 6.22.

X	$(M^S, \mathcal{M}^G, \mathcal{M})$
x_0	$(p_1, \{p_3 + p_4\}, \emptyset)$
x_1	$(p_2, \{p_3 + p_2\}, \{2p_5 + p_4\})$
x_2	$(p_1, \{p_4 + p_2\}, \{p_1 + p_4\})$
x_3	$(p_2, \{2p_2\}, \{p_4 + 2p_5, p_2 + p_4\})$
x_4	$(p_1, \{p_2 + p_4\}, \{p_4 + p_1, 2p_4\})$
x_5	$(p_2, \{2p_2\}, \{p_4 + p_2\})$
x_6	$(p_1, \{p_2 + p_4\}, \{2p_4\})$

Example 6.18. Consider the LPN in Fig. 6.20 modeling the plant, and the LPN in Fig. 6.21 describing the secret $S = \{t_1, (t_1t_3)^n, (t_1t_3)^n t_1\}$ with $n = 1, 2, 3, \ldots$. By Algorithm 8, the verifier is constructed in Fig. 6.22. Table 6.5 presents the state space of the verifier. By Theorem 6.5, since in no state of the verifier the third entry is empty, the LPN is strictly language opaque wrt T_o and S. ◇

6.6.2 Computational Complexity Analysis

In this section we compare the computational complexity (with respect to the number of states) of the proposed approach with a previous approach in the literature.

Let n_s, n_b and n_r be the number of reachable markings of G^S, basis markings of G and reachable markings of G, respectively. The number of states of the verifier $\mathcal{V}$ constructed by Algorithm 8 is bounded by

$$|X|_{Vmax} = n_s \times 2^{n_b} \times 2^{n_b}. \tag{6-4}$$

The notion of basis markings enables one to avoid enumerating all transition sequences whose projection on T_o does not belong or belongs to the secret. Since $n_b \leq n_r$ and in many cases n_b is much smaller than n_r as shown in [72], using basis markings provides significant advantages.

Let us compare the proposed approach with other methods in the literature. As pointed out at the beginning of Section 6.6, there is no method for language-based opacity analysis in Petri nets. However, some approaches in the automata framework can be used. More precisely, based on Proposition 4.2, given an arbitrary secret, the problem of verifying strict language opacity wrt $\hat{T}$ and $\hat{S}$ can be reduced to verifying language opacity wrt $P_{\hat{T}}^{-1}(\hat{S})$. Therefore, the method of verifying language opacity in [6] can also be used to verify strict language opacity in Petri nets. Note that an LPN system G is said to be language opaque wrt a secret S is equivalent to saying $L(N, M_0)$ is language-based opaque wrt S in the automata formalism. In such a case, the reachability graph of the net should be constructed and used as the automaton model. Given two automata $\mathcal{A}$ and $\mathcal{A}_S$ whose number of states are n_1 and n_2, respectively, let B be the automaton constructed to verify if $L_m(\mathcal{A})$ is language-based opaque wrt $L_m(\mathcal{A}_S)$. The number of states of B is

$$|X|_{Bmax} = 2^{n_1 \times n_2} \times 2^{n_1}. \tag{6-5}$$

Let $\mathcal{A}$ and $\mathcal{A}_S$ be the reachability graphs of G and G^S, respectively, $L_m(\mathcal{A}) = L(N, M_0)$, and $L_m(\mathcal{A}_S) = L(N^S, M_0^S) \setminus \{\varepsilon\} = S$. To construct the automaton whose accepted language is $P_{T_o}^{-1}(S)$, one just needs to add self loops of transitions in $T \setminus T_o$ on each state. We still denote $\mathcal{A}_S$ the obtained automaton to avoid introducing further notation. Namely, verifying if G is strictly language opaque wrt T_o and S is reduced to verifying if $L_m(\mathcal{A})$ is language-based opaque wrt $L_m(\mathcal{A}_S)$. By Eq. (6-5), the number of states of automaton B is

$$|X|_{Bmax} = 2^{n_r \times n_s} \times 2^{n_r}. \tag{6-6}$$

Compared with Eq. (6-4), the proposed approach is shown to be more efficient. We also point out that in practice for large-sized bounded Petri nets reachability sets may not be computable even if they are finite [72] due to the state explosion problem. However, to construct the verifier there is no need to compute all reachable markings but a subset of basis markings.

Moreover, as pointed out in the discussion about Proposition 4.1, language opacity can be semi-decided by strict language opacity. Therefore, compared with the automaton-based approach, using the proposed approach to verify language opacity is more efficient. Note

that given an arbitrary secret $S \subseteq T^*$, its projection on T_o may not be a prefix-closed bounded free-labeled Petri net language, i.e., Assumption A3 may not be satisfied. Therefore, the proposed approach cannot be used. However, for bounded LPNs the previous method provides a necessary and sufficient condition of language opacity wrt a given secret which is a regular language.

6.7 Conclusion

This chapter addresses current-state, initial-state and strict language opacity properties in labeled Petri nets. In the first part of the chapter we show that the notion of BRG can be used to verify current-state opacity by constructing the observer of the BRG. This approach has several advantages in terms of computational and space complexity. When the intruder has uncertainty about the initial marking, an extended observer can be used whose initial marking is a subset of the reachability set and the generated language is identical to the language generated by the system. In the second part of the chapter we show that under certain assumptions, initial-state opacity can be verified by constructing the initial-state estimator of the BRG. The modified basis reachability graph is introduced to verify initial-state opacity in the general case. Finally, through constructing the verifier a novel approach to verifying strict language opacity of bounded Petri nets is developed under the assumption that the set of transitions in which the intruder is interested is identical to the set of observable transitions and the secret is a prefix-closed bounded free-labeled Petri net language.

The work of this chapter has been published/accepted as:

Y. Tong, Z. W. Li, C. Seatzu, A. Giua. "Verification of Current-State Opacity Using Petri Nets", In *Proceedings of the 34th American Control Conference* (ACC'15), 2015: 1935-1940.

Y. Tong, Z. W. Li, C. Seatzu, A. Giua. "Verification of Initial-State Opacity in Petri Nets", In *Proceedings of the 54th IEEE Conference on Decision and Control* (CDC'15), 2015: 344-349.

Y. Tong, Z. Y. Ma, Z. W. Li, C. Seatzu, A. Giua, "Verification of Language-Based Opacity in Petri Nets Using Verifier", the 35th American Control Conference (ACC'16), 2016: 757-763.

Y. Tong, Z. W. Li, C. Seatzu, A. Giua, "Verification of State-Based Opacity Using Petri Nets", *IEEE Transactions on Automatic Control*, Vol. 62, No. 6, **2017**. Early Access available online. DOI: 10.1109/TAC.2016.2620429.

Chapter 7 Supervisory Enforcement of Current-State Opacity with Incomparable Observations

7.1 Introduction

In this chapter we tackle the opacity enforcement problem in DESs using supervisory control theory. Given a system that is not current-state opaque with respect to a given secret, our purpose in this chapter is to design a maximally permissive supervisor that restricts the behavior of the system to ensure that the controlled system is current-state opaque. There has been some related work on the design of supervisors to enforce opacity properties [29, 37, 39, 40, 44].

We point out that all the aforementioned works are carried out in the framework of finite automata and rely on Ramadge and Wonham's basic theory of supervisory control for DES [73]. Note that the objective of opacity enforcement is not concerned with liveness since opacity properties focus on a set of indiscernible runs from the perspective of the intruder instead of individual runs. What distinguishes our work from the existing works consists in three aspects.

- No containment relation is assumed between the sets E_I, E_S of events observable by the intruder and by the supervisor, respectively. We call this general setting *incomparable observations*. In this sense, the problem considered here is more general than the one in [29, 37, 39, 40, 44].

- We also relax the assumption made in [36–39, 74] that all controllable events E_C should be observable.

- Finally, we define $\mathcal{A}$-*opacity* of a language. We show that if a controlled system $Sup/\mathcal{A}$ is current-state opaque then its generated language $L(Sup/\mathcal{A})$ is $\mathcal{A}$-opaque but the converse may not hold. However, if the intruder does not know the supervisor, $L(Sup/\mathcal{A})$ being G-opaque is sufficient to guarantee current-state opacity of $Sup/\mathcal{A}$.

To be more clear, comparison between the proposed approach and previous ones [37, 40, 74] is summarized in Table 7.1. All the approaches are developed for deterministic finite automata but under different assumptions. The last row of Table 7.1 presents their computational complexity, where X is the set of states of the system and E_C is the set of controllable events.

Table 7.1 Comparison between the proposed approach and previous approaches.

Works	[37]	[40]	[74]	This Chapter
Assumptions	$E_I \subseteq E_S$ (or $E_S \subseteq E_I$) $E_C \subseteq E_S$	$E_I \subseteq E_S$	$E_C \subseteq E_S$	None
Does the intruder know the supervisor?	Yes	Yes	No	No
Complexity	$\mathcal{O}(\lvert X\rvert \times 2^{\lvert X\rvert})$	$\mathcal{O}(2^{2(\lvert X\rvert+\lvert E_C\rvert)})$	$O(2^{2^{2\lvert X\rvert}})$	$\mathcal{O}(2^{2(\lvert X\rvert\times 2^{\lvert X\rvert}+\lvert E_C\rvert)})$

In this chapter, first a structure called *augmented I-observer* is constructed. The augmented I-observer of a system is a deterministic finite automaton, where each state contains the current-state estimate of the intruder. Based on the augmented I-observer, evolutions of the system that satisfy current-state opacity can be characterized. Then we show that the current-state opacity enforcement problem can be reduced to the basic supervisory control problem under partial observation [45]. Note that the maximally permissive supervisor enforcing current-state opacity may not be unique. Thus we obtain a set of *locally optimal* supervisors where the adverb "locally" points out that the behavior of the controlled system under each of them is not strictly included in another. Finally, we show that based on the proposed approach it is possible to solve current-state opacity enforcement problem assuming the intruder does not know to the supervisor. To summarize, three are the main contributions of the chapter.

- The definition of $\mathcal{A}$-opacity of a language that enables us to formalize the opacity enforcement problem under the assumption that the intruder has no knowledge (or at most a partial knowledge) of the supervisor.

- The definition of a novel finite structure, the augmented I-observer, that enables one to relax the assumptions $E_S \subseteq E_I$ (or $E_I \subseteq E_S$) and $E_C \subseteq E_S$.

- The demonstration that based on the notion of $\mathcal{A}$-opacity and the augmented I-observer, the current-state opacity enforcement problem can be reduced to the basic supervisory control problem under partial observation, which is different from the approaches in literature. Then, locally optimal supervisors are achieved using appropriate supervisory control techniques.

This chapter improves the results presented in our paper [74] to a more general setting, by removing the assumption that all events controllable by the supervisor should be observable.

In addition, under the same assumptions, the proposed approach has lower complexity than the approach in [74].

The rest of this chapter is organized as follows. Basic notions on supervisory control theory are recalled in Section 7.2. Section 7.3 recalls the definition of current-state opacity, and the current-state opacity enforcement problem is formalized. In Section 7.4, a method for the synthesis of a locally optimal supervisor is proposed. The computational complexity of the proposed approach is analyzed in Section 7.5. Finally, this chapter is concluded.

7.2 Supervisory Control Theory

Before we formally present the problem addressed in this chapter, we recall supervisory control theory presented in [73]. Given a system modeled by a DFA $\mathcal{A} = (X, E, \delta, x_0)$, the goal of supervisory control is to design a supervisor such that the controlled system satisfies a set of constraints represented by a language $K \subseteq L(\mathcal{A})$ (we call it a *specification language*). The supervisor observes a subset E_S of the events in E and is able to control a subset of events $E_C \subseteq E$. After the supervisor observes a string generated by the system it tells the system the set of events that are allowed next so that the system will not violate the specification. According to the theory in [73], a supervisor is denoted as $Sup = (Y, E_S, \delta_s, y_0, \Psi)$, where (Y, E_S, δ_s, y_0) is a DFA and

$$\Psi : Y \to \{E' \subseteq E | E_{UC} \subseteq E'\}$$

specifies the set of events enabled by the supervisor. Fig. 7.1 illustrates the paradigm of supervisory control under partial observation. Let $\sigma \in L(\mathcal{A})$ be the string generated by the system and $w_s = P_S(\sigma)$ be the corresponding observation of the supervisor. Then the set of events enabled by the supervisor is $\Psi(y)$, where $y = \delta_s(y_0, w_s)$. System $\mathcal{A}$ under the control of a suitable supervisor Sup is denoted as $Sup/\mathcal{A}$, and it satisfies $L(Sup/\mathcal{A}) \subseteq K$.

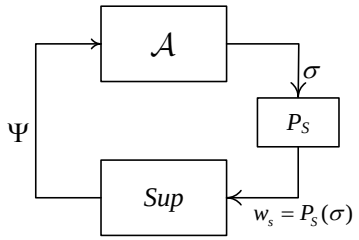

Fig. 7.1 Supervisory control under partial observation.

Definition 7.1. [73] Given a DFA $\mathcal{A}$, a set of controllable events E_C, and a language $K \subseteq L(\mathcal{A})$, K is said to be *controllable* (wrt $L(\mathcal{A})$ and E_C) if

$$\overline{K}E_{UC} \cap L(\mathcal{A}) \subseteq \overline{K},$$

where $E_{UC} = E \setminus E_C$. $\diamond$

In other words, K is controllable if in $L(\mathcal{A})$ no string that is a prefix of K exits from the prefix closure of K when followed by an uncontrollable event. It is known that controllability is preserved under arbitrary unions and consequently the supremal controllable sublanguage of a given language exists.

Definition 7.2. [73] Given a DFA $\mathcal{A}$, a set of controllable events E_C, a set of observable events E_S, and a language $K \subseteq L(\mathcal{A})$, K is said to be *observable* (wrt $L(\mathcal{A})$, E_S and E_C) if for all $\sigma, \sigma' \in \overline{K}$ and all $e \in E_C$ such that $\sigma e \in L(\mathcal{A})$, $\sigma' e \in \overline{K}$ and $P_S(\sigma) = P_S(\sigma')$, $\sigma e \in \overline{K}$ holds. $\diamond$

Roughly speaking, observability requires that the observation of the supervisor (i.e., the projection on E_S) provides sufficient information to decide after the occurrence of a controllable event whether the resultant string is still in $\overline{K}$. Unlike controllability, observability is however not preserved under union, therefore the supremal observable sublanguage of a given language may not exist. However maximal observable sublanguages exist, but are not usually unique.

Theorem 7.1. [73] Let $K \subseteq L(\mathcal{A})$ be a prefix-closed nonempty language, E_C the set of controllable events and E_S the set of observable events. There exists a supervisor Sup such that $L(Sup/\mathcal{A}) = K$ if and only if K is controllable and observable.

Definition 7.3. Given a system $\mathcal{A}$, a set of controllable events E_C, a set of observable events E_S by the supervisor, and a specification language $K \subseteq L(\mathcal{A})$, the *supervisory control and observation problem* (SCOP) consists in finding a locally optimal supervisor Sup such that:

1. $L(Sup/\mathcal{A}) \subseteq K$

2. $L(Sup/\mathcal{A})$ is maximal, i.e., for any other supervisor Sup',

$$L(Sup'/\mathcal{A}) \subseteq K \Rightarrow L(Sup/\mathcal{A}) \not\subset L(Sup'/\mathcal{A}). \quad \diamond$$

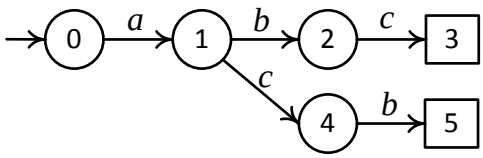

Fig. 7.2 System $\mathcal{A}$ where $E_S = \{a\}$, $E_C = \{a, b, c\}$ and states 3 and 5 should be unreachable.

A SCOP involves the system[1] $\mathcal{A}$, the set E_S of events observable by the supervisor, the set E_C of events controllable by the supervisor, and the specification language K. To be concise, we call this problem SCOP($\mathcal{A}, E_S, E_C, K$).

Since the supremal observable sublanguage may not exist, there may not be the supremal controllable and observable sublanguage of a given language. Consequently, there may be multiple solutions to a SCOP and they are called "locally optimal" because under the control of the corresponding supervisors, the behaviors of the controlled system are incomparable.

The SCOP has been considered in the literature and many different methods have been proposed to solve it [75–80]; in the chapter we briefly introduce the approach recently presented in [79].

The authors of [79] propose a structure, called *total controller*, based on which all locally optimal supervisors of the SCOP can be computed. Given a SCOP($\mathcal{A}, E_S, E_C, K$) with $K = L(H)$, it is assumed, without loss of generality, that $H = (X_H, E, \delta_H, x_{H,0})$ is a strict sub-automaton[2] of $\mathcal{A}$. In other words, the language specification K of a SCOP is reduced to a state specification: a state $x \in X$ is legal iff $x \in X_H$, i.e., $\sigma \notin K$ with $x = \delta(x_0, \sigma)$. We denote $F = X \setminus X_H$ the set of forbidden states. Clearly, the controlled system $Sup/\mathcal{A}$ would also be a strict sub-automaton of $\mathcal{A}$. In this section, such an approach is introduced through a numerical example.

Consider the system $\mathcal{A} = (X, E, \delta, x_0)$ in Fig. 7.2, where $E_S = \{a\}$ and $E_C = \{a, b, c\}$. The set of forbidden states is $F = \{3, 5\}$. The approach proposed by Yin and Lafortune [79] can be summarized as follows. First, construct a finite structure called a *total controller*, which enumerates all possible control policies of the system. In the total controller there are two types of states: Y-states $\mathcal{Y} \subseteq X$ in rounded boxes and Z-states $\mathcal{Z} = (Z, I)$ in rectangles, where $Z \subseteq X$ and I is a control decision, i.e., it contains the set of events enabled by the supervisor. The initial state of the total controller is $\mathcal{Y}_0 = \{x_0\}$. Y-states are driven to Z-states by control decisions. At each Y-state $\mathcal{Y}$, we enumerate al-

[1] Properly speaking, the SCOP concerns the language $L(\mathcal{A})$

[2] If H is not a strict subautomaton of $\mathcal{A}$, the algorithm in [81] can be used to transform both of them to $\mathcal{A}'$ and H', respectively, such that the H' is a strict subautomaton of $\mathcal{A}'$.

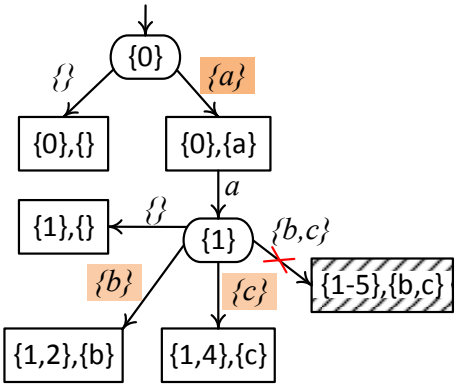

Fig. 7.3 Total controller of $\mathcal{A}$ in Fig. 7.2. Removing the state in the dashed box, the all inclusive controller is obtained.

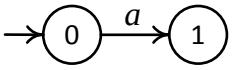

Fig. 7.4 The automaton structure of the two optimal supervisors: for Sup_1, $\Psi(0) = \{a\}$ and $\Psi(1) = \{b\}$; for Sup_2, $\Psi(0) = \{a\}$ and $\Psi(2) = \{c\}$.

l control decisions[3], and then the successor Z-state corresponding to a control decision is computed: Z is the set of states reachable from $\mathcal{Y}$ by firing unobservable events enabled by the control decision and I is the control decision. For instance, in Fig. 7.3, from Y-state $\{1\}$, for control decision $\{b\}$ the Z-state reached is $(\{1,2\},\{b\})$ and for control decision $\{b,c\}$ the Z-state reached is $(\{1-5\},\{b,c\})$. Z-states are driven to Y-states by observable events $e \in E_S$ that are defined at a state in Z and enabled by the control decision I. The successor Y-state is the set of states reachable from a state in Z after the occurrence of e. For instance, from Z-state $(\{0\},\{a\})$ event a is enabled at 0 and is allowed by the control decision, therefore the Y-state reached is $\{1\}$.

After the total controller is constructed, removing all the Y-states and Z-states that contain a forbidden state (i.e., 3 and 5 in this case) and the related arcs, the *all inclusive controller* is obtained. In Fig. 7.3 $(\{1-5\},\{b,c\})$ is such a state and should be removed. The all inclusive controller models all the control policies that enforce the specification language. Finally, after each Y-state we pick a control decision that is not a strict subset of any other decisions. A combination of those local maximal control decisions corresponds to an optimal supervisor.

It has been proven that the time complexity of the approach proposed in [79] to solving the SCOP is $\mathcal{O}(|X||E|2^{|X|+|E_C|})$. In Fig. 7.3, each local maximal control decision is colored. There are two optimal supervisors Sup_1 and Sup_2 (see Fig. 7.4) and the behav-

[3]For the system in Fig. 7.2, there is no need to enumerate all control decisions when Y-state is $\{0\}$ or $\{1\}$. Indeed, from state 0, observable event a would never occur before b and c, therefore all other control policies are equivalent to $\{a\}$ or $\{\}$. From state 1, event a would never be executed. Therefore, control policies containing a are redundant.

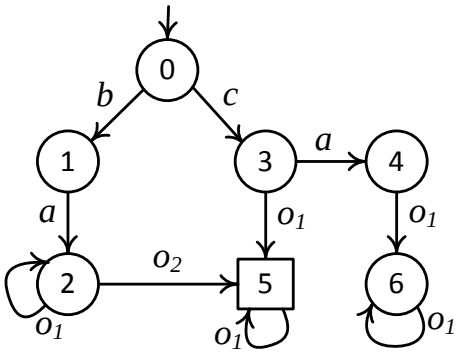

Fig. 7.5 System $\mathcal{A}$ that is not CSO wrt $S = \{5\}$ and $E_I = \{o_2\}$ in Example 7.1.

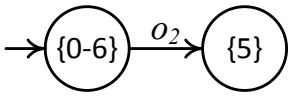

Fig. 7.6 Observer of the system in Fig. 7.5 for the intruder.

iors of the controlled system under different supervisors are $L(Sup/\mathcal{A}_1) = \{\varepsilon, a, ab\}$ and $L(Sup/\mathcal{A}_2) = \{\varepsilon, a, ac\}$, respectively.

7.3 Problem Formalization

We first recall the definition of CSO in Chapter 4 and then introduce a related notion: *$\mathcal{A}$-opaque language*, which would be helpful to formalize the opacity enforcement problem.

Definition 7.4. Given a system $\mathcal{A} = (X, E, \delta, x_0)$, a secret $S \subseteq X$, and a set E_I of events observable by the intruder, the system is said to be *current-state opaque* (CSO) wrt S and E_I if $\forall \sigma \in L(\mathcal{A})$ such that $\delta(x_0, \sigma) \in S$,

$$\exists \sigma' \in L(\mathcal{A}) : P_I(\sigma') = P_I(\sigma) \text{ and } \delta(x_0, \sigma') \notin S,$$

where $P_I : E^* \to E_I^*$ is the natural projection from E to E_I. $\diamond$

Given a word $w \in E_I^*$, we denote $\mathcal{C}_I(w) = \{x \in X | \exists \sigma \in L(\mathcal{A}) : \delta(x_0, \sigma) = x, P_I(\sigma) = w\}$ the set of states consistent with w. Thus, $\mathcal{A}$ is CSO wrt S and E_I if and only if $\forall w \in P_I(L(\mathcal{A})), \mathcal{C}(w) \not\subseteq S$ holds.

Example 7.1. Consider the system in Fig. 7.5. Let $E_I = \{o_2\}$ and $S = \{5\}$ (the secret state is in a box). The corresponding observer for the intruder is shown in Fig. 7.6. Since there exists $w = o_2$ such that $\mathcal{C}_I(w) = \{5\} \subseteq S$, the system is not current-state opaque wrt S and E_I. $\diamond$

Let us introduce the following notion of opacity that is related to a sublanguage of the generated language of the system and is useful to formalize the result of the work.

Definition 7.5. Given a system $\mathcal{A} = (X, E, \delta, x_0)$, a secret $S \subseteq X$ and a set E_I of events observable by the intruder, a sublanguage $L \subseteq L(\mathcal{A})$ is said to be $\mathcal{A}$-opaque (wrt S and E_I) if $\forall \sigma \in L$ such that $\delta(x_0, \sigma) \in S$,

$$\exists \sigma' \in L(\mathcal{A}) : \delta(x_0, \sigma') \notin S, \quad P_I(\sigma) = P_I(\sigma'). \qquad \diamond$$

In words, a language L is $\mathcal{A}$-opaque if for any string σ in L leading to a secret state there is another string in the generated language of $\mathcal{A}$ that produces the same observation as $P_I(\sigma)$ but does not lead to a secret state. Clearly, by Definitions 7.4 and 7.5 Corollary 7.1 follows.

Corollary 7.1. Given a system $\mathcal{A} = (X, E, \delta, x_0)$, a secret $S \subseteq X$ and a set E_I of events observable by the intruder, $\mathcal{A}$ is current-state opaque wrt S and E_I if and only if $L(\mathcal{A})$ is $\mathcal{A}$-opaque.

In other words, CSO of a system $\mathcal{A}$ is equivalent to $\mathcal{A}$-opacity of its generated language.

Proposition 7.1. Given a system $\mathcal{A}$, a secret $S \subseteq X$, a set E_I of events observable by the intruder, and two $\mathcal{A}$-opaque languages $L_1, L_2 \subseteq L(\mathcal{A})$, then it holds:

i) $L_1 \cup L_2$ is $\mathcal{A}$-opaque;

ii) $\forall L \subseteq L_1$, L is $\mathcal{A}$-opaque.

Proof: i) By assumption, L_i (with $i = 1, 2$) is $\mathcal{A}$-opaque. By Definition 7.5, for all $\sigma \in L_i, \mathcal{C}_I(P_I(\sigma)) \nsubseteq S$. Therefore, for all $\sigma \in L_1 \cup L_2, \mathcal{C}_I(P_I(\sigma)) \nsubseteq S$, i.e., $L_1 \cup L_2$ is $\mathcal{A}$-opaque. ii) Given a subset L of L_i, for all $\sigma \in L, \mathcal{C}_I(P_I(\sigma)) \nsubseteq S$, i.e., L is $\mathcal{A}$-opaque. $\square$

Therefore, the $\mathcal{A}$-opacity property of a language is closed under union, and the supremal $\mathcal{A}$-opaque sublanguage of a given language exists. Any sublanguage of a $\mathcal{A}$-opaque language is still $\mathcal{A}$-opaque.

Proposition 7.2. Let $Sup/\mathcal{A}$ be the controlled system of $\mathcal{A} = (X, E, \delta, x_0)$ under a supervisor Sup, $E_I \subseteq E$ the set of events observable by the intruder, and $S \subseteq X$ the secret. Given a language $L \subseteq L(\mathcal{A})$, if L is $Sup/\mathcal{A}$-opaque wrt S and E_I, then L is $\mathcal{A}$-opaque wrt S and E_I.

Proof: Let $Sup/\mathcal{A} = (X', E', \delta', x'_0)$. By the results in Section 7.2, $Sup/\mathcal{A}$ is a strict sub-automaton of $\mathcal{A}$. Namely, $X' \subseteq X$, $E' \subseteq E$, $x'_0 = x_0$, and for $\sigma \in E^*$, if $\delta'(x'_0, \sigma) = x$ in $Sup/\mathcal{A}$, then $\delta(x_0, \sigma) = x$ in $\mathcal{A}$. Assume L is $Sup/\mathcal{A}$-opaque. Then for all $\sigma \in L$ such that $\delta(x_0, \sigma) \in S$, there exists $\sigma' \in L(Sup/\mathcal{A})$ such that $\delta'(x'_0, \sigma') \notin S$ and $P_I(\sigma) = P_I(\sigma')$. Since $L(Sup/\mathcal{A}) \subseteq L(\mathcal{A})$, there also exists $\sigma' \in L(\mathcal{A})$ such that $\delta(x_0, \sigma') \notin S$ and $P_I(\sigma) = P_I(\sigma')$. Therefore, L is $\mathcal{A}$-opaque wrt S and E_I. □

Therefore, if $L(Sup/\mathcal{A}) \subseteq L(\mathcal{A})$ is $Sup/\mathcal{A}$-opaque (i.e., $Sup/\mathcal{A}$ is CSO) then $L(Sup/\mathcal{A})$ is also $\mathcal{A}$-opaque. Note that the converse of Proposition 7.2 is not true. In other words, even if $L(Sup/\mathcal{A})$ of a controlled system $Sup/\mathcal{A}$ is $\mathcal{A}$-opaque wrt S and E_I, the controlled system $Sup/\mathcal{A}$ may not be CSO wrt S and E_I. Therefore, CSO of $Sup/\mathcal{A}$ generally is a stronger requirement than $L(Sup/\mathcal{A})$ being $\mathcal{A}$-opaque.

Example 7.2. Consider the system $\mathcal{A}$ in Fig. 7.5 and its controlled system $Sup_2/\mathcal{A}$ in Fig. 7.13. Let $S = \{5\}$, $E_I = \{o_1, o_2\}$, $E_S = \{o_1\}$, and $E_C = \{a, b, c\}$. Clearly, $L(Sup_2/\mathcal{A})$ is $\mathcal{A}$-opaque wrt S and E_I but not $Sup_2/\mathcal{A}$-opaque. Namely, $Sup_2/\mathcal{A}$ is not CSO wrt S and E_I. Indeed, when the intruder observes o_1, if it knows the structure of $Sup_2/\mathcal{A}$, its estimate would be $\mathcal{C}_I(o_1) = \{5\} \subseteq S$, i.e., $Sup_2/\mathcal{A}$ is not CSO; on the contrary, if the intruder does not know the structure of $Sup_2/\mathcal{A}$, its estimate would be $\mathcal{C}_I(o_1) = \{2, 5, 6\} \nsubseteq S$, i.e., the intruder is not able to discover the secret. ◇

Example 7.2 also shows that if the intruder knows the supervisor Sup, to guarantee that the intruder does not discover the secret, $L(Sup/\mathcal{A})$ should be $Sup/\mathcal{A}$-opaque. On the contrary, if the intruder does not know the supervisor Sup, it is sufficient that $L(Sup/\mathcal{A})$ is $\mathcal{A}$-opaque. In the latter case, enforcing CSO on $\mathcal{A}$ is equal to synthesizing a supervisor Sup of $\mathcal{A}$ such that $L(Sup/\mathcal{A})$ is $\mathcal{A}$-opaque, which clearly is a weaker condition than $Sup/\mathcal{A}$ being CSO.

Note that $\mathcal{A}$-opacity of $L(Sup/\mathcal{A})$ may guarantee CSO of $Sup/\mathcal{A}$ also in some cases in which the intruder knows there is a supervisor acting on the system but has not sufficient information to determine it exactly. Suppose the intruder knows there is a supervisor and has some information on E_S and E_C but not precise. Then the intruder may synthesize an estimate supervisor Sup' on $\mathcal{A}$ such that $L(Sup'/\mathcal{A})$ is $Sup'/\mathcal{A}$-opaque. However, if $L(Sup/\mathcal{A})$ is $Sup'/\mathcal{A}$-opaque, then the intruder is still not able to discover the secret.

Example 7.3. Consider Example 7.2 again. Suppose now the intruder knows there is a supervisor and believes the supervisor can observe $E'_S = \{a, o_1\}$, and can control $E'_C = \{b, c\}$, which are different from what the supervisor really can observe and control. The estimate

Table 7.2 Observable and controllable events in Example 7.4.

Events	E_I	E_S	E_C
o_1	$\times$	$\surd$	$\times$
o_2	$\surd$	$\times$	$\times$
a	$\times$	$\times$	$\surd$
b	$\times$	$\times$	$\surd$
c	$\times$	$\times$	$\surd$

supervisor synthesized based on E'_S and E'_C is Sup' which disables event b when observing nothing. Consider the supervisor Sup_2 defined in Example 7.2. It is easy to see that $L(Sup_2/\mathcal{A})$ is $Sup'/\mathcal{A}$-opaque wrt S and E_I. Therefore, under the control of Sup_2 the intruder is still not able to infer the secret. $\diamond$

For simplicity, in the remainder of the chapter it is directly assumed that the intruder does not know a supervisor controlling the plant to enforce opacity. Introducing such an assumption enables us to solve opacity enforcement using supervisory control in an efficient way. Meanwhile, imposing such an assumption is reasonable and meaningful. Indeed, this is realistic in many practical situations. Furthermore, it is interesting from a theoretical point of view since it provides some insights into tackling more general and complicated problems. The problem we want to solve in this work can be formalized as follows.

Definition 7.6. Given a system $\mathcal{A} = (X, E, \delta, x_0)$, a secret $S \subseteq X$, a set E_I of events observable by the intruder, a set E_S of events observable by the supervisor, and a set E_C of controllable events, synthesize a locally optimal supervisor Sup such that

1. $L(Sup/\mathcal{A})$ is $\mathcal{A}$-opaque wrt S and E_I;
2. For any other supervisor Sup' such that $Sup'/\mathcal{A}$ is $\mathcal{A}$-opaque wrt S and E_I it holds
$$L(Sup/\mathcal{A}) \not\subset L(Sup'/\mathcal{A}).$$ $\diamond$

A CSOEP involves the system $\mathcal{A}$, the set E_I of events observable by the intruder, the secret S, the set E_S of events observable by the supervisor and the set E_C of events controllable by the supervisor. To be concise, we call this problem CSOEP($\mathcal{A}, E_I, S, E_S, E_C$). A solution to the CSOEP is called a *locally optimal supervisor*.

Example 7.4. Consider again the system in Fig. 7.5. From Example 7.1 we know that the system is not current-state opaque wrt $S = \{5\}$ and $E_I = \{o_2\}$. Now we want to design a locally optimal supervisor Sup, so that $L(Sup/\mathcal{A})$ is $\mathcal{A}$-opaque. The sets of events observable\controllable by the intruder and the supervisor are shown in Table 7.2. In this

case, E_I and E_S are not comparable, i.e., neither $E_I \subseteq E_S$ nor $E_S \subseteq E_I$ holds, and not all controllable events are observable, i.e., $E_C \not\subseteq E_S$. $\diamond$

Proposition 7.3. There exists a solution to the CSOEP if and only if there exists a prefix-closed language $K \subseteq L(\mathcal{A})$ such that

1. K is controllable (wrt $L(\mathcal{A})$ and E_C) and observable (wrt $L(\mathcal{A})$, E_S and E_C);

2. K is $\mathcal{A}$-opaque (wrt S and E_I);

3. For any other controllable, observable and $\mathcal{A}$-opaque language $K' \subseteq L(\mathcal{A})$, $K \not\subset K'$.

Proof: By Theorem 7.1, the first item is a necessary and sufficient condition for the existence of a supervisor that restricts the behaviour of the system to K. Items 2 and 3 correspond to items 1 and 2, respectively, of Definition 7.3 that formalize the requirements that a supervisor has to satisfy for a locally optimal solution to the CSOEP. $\square$

Thus, to solve the CSOEP we have to compute a prefix-closed maximal controllable, observable and $\mathcal{A}$-opaque sublanguage of $L(\mathcal{A})$. It is known that the supremal observable sublanguage may not exist. Therefore such a maximal controllable, observable and $\mathcal{A}$-opaque sublanguage, if it exists, may not be unique. In other words, there may exist a set of locally optimal supervisors.

In the next section, we introduce a structure, called *augmented I-observer*, based on which the supremal $\mathcal{A}$-opaque sublanguage can be characterized and the optimal supervisors can be designed.

7.4 Synthesis of Locally Optimal Supervisors

To design locally optimal supervisors, we have to characterize a maximal controllable and observable behavior of the system such that the secret will never be leaked. To do this, we need to first characterize the supremal $\mathcal{A}$-opaque sublanguage of the system as the specification language K, and then compute a maximal controllable and observable sublanguage of K. Indeed, by Proposition 7.1 if a language is $\mathcal{A}$-opaque, any sublanguage of it is still $\mathcal{A}$-opaque. Unfortunately, the absence of specific containment relationships between sets E_I and E_S makes the solution via a single structure, as in [37, 40], tricky. In the following we provide an example where the approach in [37] fails since none of the containment relationships $E_I \subseteq E_S$ or $E_S \subseteq E_I$ holds.

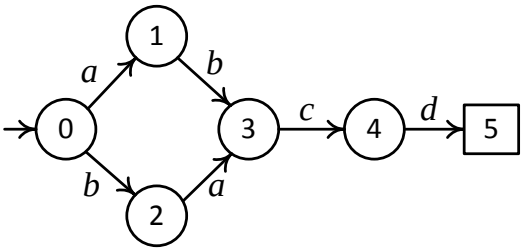

Fig. 7.7 System $\mathcal{A}$ that is not CSO wrt $S = \{5\}$ and $E_I = \{a, d\}$.

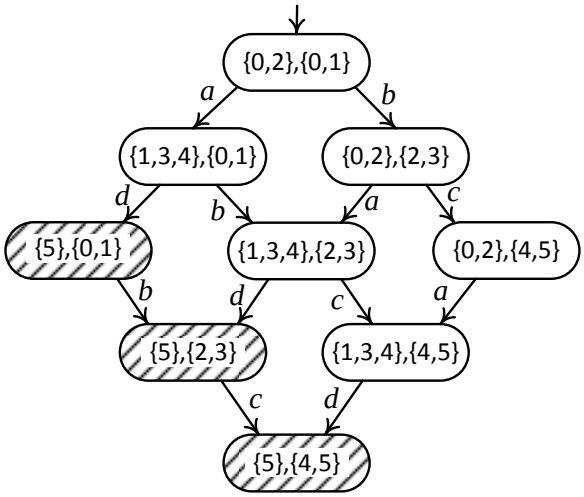

Fig. 7.8 Parallel composition N of the observers for the intruder and the supervisor.

Example 7.5. Consider the system in Fig. 7.7. Let $E_I = \{a, d\}$, $E_S = \{b, c\}$, $E_C = \{c\}$, and $S = \{5\}$. Obviously, the system is not opaque wrt S and E_I since when the intruder observes ad it unambiguously knows that the current state is 5. According to [37], observers of the system for the intruder and the supervisor should be constructed first. Then we have to compute the parallel composition N of these two observers to characterize the behavior that would leak the secret and that should be forbidden (see Fig. 7.8, states in shadow should be unreachable).

Finally, by computing the observer (wrt E_S) of the parallel composition structure the optimal supervisor can be obtained. Without the assumption $E_I \subseteq E_S$ or $E_S \subseteq E_I$, the parallel composition between the observers would introduce event sequences (e.g., $\sigma = abd$) not belonging to $P_o(L(\mathcal{A}))$, where $P_o : E^* \to (E_I \cup E_S)^*$. In the case at hand, being $E_o = E$, it is $P_o(L(\mathcal{A})) = L(\mathcal{A})$. As a result, the behavior of the system would be over restricted. For instance, sequence ab does not leak the secret. However, it should be disabled: N tells that after uncontrollable event d occurs, sequence abd will lead to a state in shadow. Therefore, the obtained supervisor would not be optimal, or even no such an opacity enforcing supervisor exists (as in the case at hand). Note that assuming $E_C = \{o_1\} = E_S$

the approach in [37] coincidentally works for Example 7.4 though neither $E_I \subseteq E_S$ nor $E_S \subseteq E_I$ holds. $\diamond$

In this work, we show that locally optimal supervisors for the CSOEP can be designed in two phases, without assuming $E_I \subseteq E_S$, or $E_S \subseteq E_I$, or $E_C \subseteq E_S$. First, by introducing a structure, called *augmented I-observer*, the supremal $\mathcal{A}$-opaque sublanguage can be computed. Then, applying the method recalled in Section 7.2 to the augmented I-observer, the locally optimal supervisors can be designed. The augmented I-observer of system $\mathcal{A} = (X, E, \delta, x_0)$ is a DFA denoted as $\mathcal{A}_g = (Q, E, \delta_g, q_0)$. A state $q \in Q$ of $\mathcal{A}_g$ is a pair (C_I, x), where $C_I \subseteq X$ and $x \subseteq X$. The initial state of the augmented I-observer is $q_0 = (R_\varepsilon(x_0), x_0)$. Note that when computing $R_\varepsilon(x)$ and $R_e(x)$ of a given state x, all events not in E_I are regarded as ε. Algorithm 9 illustrates the construction of the augmented I-observer.

Now we explain the main ideas behind Algorithm 9. The initial state of the augmented I-observer is $q_0 = (R_\varepsilon(x_0), x_0)$, i.e., the pair (set of states estimated by the intruder when observing nothing, initial state of the system). Given a state $q = (C_I, x) \in Q$ and an event $e \in E$ that is defined at x in $\mathcal{A}$, using Algorithm 9, the generic state $\delta_g(q, e) = q' = (C_I', x')$ in the augmented I-observer is computed as follows. C_I' is updated to the new intruder estimate when event e is observed by the intruder; otherwise, $C_I = C_I'$. State x' is reached by the occurrence of e at x in $\mathcal{A}$. If q' is a new state, it is added to Q, otherwise Q does not change. The set $F = \{q = (C_I, x) \in Q | C_I \subseteq S\}$ is the set of states of $\mathcal{A}_g$ where the estimate of the intruder is a subset of the secret.

The maximum number of states of the augmented I-observer is $|X| \times 2^{|X|}$. Clearly, the construction of the augmented I-observer is completely different from the parallel composition in [37] and the parallel observer proposed in [74].

Example 7.6. Consider the problem in Example 7.4. Using Algorithm 9, the augmented I-observer is constructed and shown in Fig. 7.9, where states in F are in dashed boxes. $\diamond$

Proposition 7.4. Let $\mathcal{A} = (X, E, \delta, x_0)$ be a system, E_I the set of events observable by the intruder, and S the secret. The augmented I-observer $\mathcal{A}_g = (Q, E, \delta_g, q_0)$ constructed using Algorithm 9 has the following properties:

i) $L(\mathcal{A}_g) = L(\mathcal{A})$;

ii) $\{\sigma \in L(\mathcal{A}_g) | \delta_g(q_0, \sigma) \in F\} = \{\sigma \in L(\mathcal{A}) | \mathcal{C}_I(P_I(\sigma)) \subseteq S\}$.

Algorithm 9 Computation of the augmented I-observer

Input: A system $\mathcal{A} = (X, E, \delta, x_0)$, the sets of events E_I and the secret S.

Output: The corresponding augmented I-observer $\mathcal{A}_g = (Q, E, \delta_g, q_0)$ and the subset F of Q.

$q_0 := (R_\varepsilon(x_0), x_0)$ and assign no tag to it;
$Q := \{q_0\}$;
if $R_\varepsilon(x_0) \subseteq S$, **then**
    $F := \{q_0\}$;
else
    $F := \emptyset$;
end if
while $q = (C_I, x) \in Q$ with no tag exists, **do**
    for all $e \in E$ such that $\delta(x, e)!$, **do**
        if $e \in E_I$, **then**
            $C'_I := \bigcup_{x \in C_I} R_e(x)$;
        else
            $C'_I := C_I$;
        end if
        $x' := \delta(x, e)$;
        $q' := (C'_I, x')$;
        if $q' \notin Q$ **then**
            $Q := Q \cup \{q'\}$;
        end if
        if $C'_I \subseteq S$, **then**
            $F := F \cup \{q'\}$;
        end if
        $\delta_g(q, e) := q'$;
    end for
    Tag q "old";
end while
Remove all tags;
Output $\mathcal{A}$.

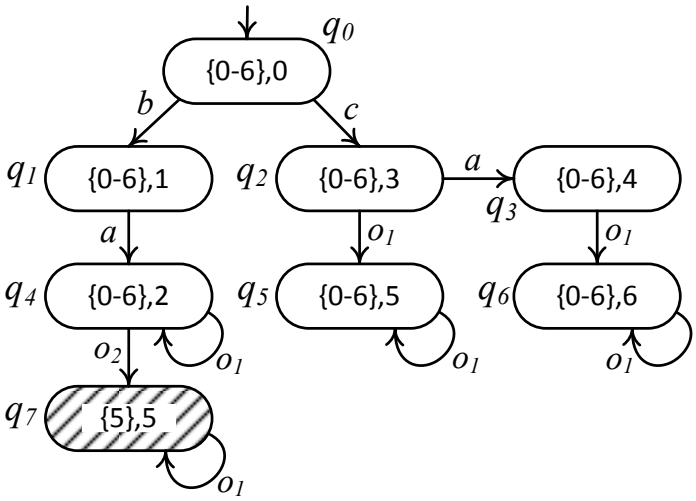

Fig. 7.9 Augmented I-observer $\mathcal{A}_g$ of the system in Example 7.6, where states in F are in dashed boxes.

Proof:

i) The statement follows from the fact that Steps 9 and 15 of Algorithm 9 consider all the events (and only them) that are defined at each state of $\mathcal{A}$.

ii) Let $q = (C_I, x) = \delta_g(q_0, \sigma)$. By Steps 3 to 7, and 20 to 22 of Algorithm 9, $C_I = \mathcal{C}_I(P_I(\sigma))$ holds. Therefore, $\delta_g(q_0, \sigma) \in F$ if and only if $\mathcal{C}_I(P_I(\sigma)) \subseteq S$.

□

Moreover, by Steps 3 to 5 and 20 to 22 of Algorithm 9, there exists $\sigma \in L(\mathcal{A})$ such that $\mathcal{C}_I(P_I(\sigma)) \subseteq S$, if and only if $F \neq \emptyset$. Therefore, we have the following corollary showing that the augmented I-observer can also be used to verify current-state opacity.

Corollary 7.2. Given a system $\mathcal{A}$, a secret S and the sets of events E_I and E_S, let $\mathcal{A}_g = (Q, E, \delta_g, q_0)$ be the augmented I-observer. $\mathcal{A}$ is current-state opaque wrt S and E_I if and only if $F = \emptyset$.

Proof: Follows from Steps 3 to 5 and 20 to 22 of Algorithm 9, Proposition 7.4 and Definition 7.4. □

The following proposition shows how it is possible to compute the supremal $\mathcal{A}$-opaque sublanguage of $\mathcal{A}$ using the augmented I-observer.

Proposition 7.5. The supremal $\mathcal{A}$-opaque sublanguage of $L(\mathcal{A})$ is

$$K = \{\sigma \in L(\mathcal{A}_g) | \delta_g(q_0, \sigma) \notin F\}.$$

Proof: First, we prove that K is opaque. Let $\sigma \in K, \delta_g(q_0, \sigma) = q = (C_I, C_S)$. Since $q \notin F$, $C_I \nsubseteq S$, i.e., $\mathcal{C}_I(w) \nsubseteq S$, where $w = P_I(\sigma)$. Therefore, K is opaque. Now we show that K is the "largest" opaque sublanguage of $L(\mathcal{A})$ and for any other opaque language $L \subseteq L(\mathcal{A})$, L is contained in K. Let $\sigma \in L$ and $q = \delta_g(q_0, \sigma) = (C_I, C_S)$. Since L is opaque, $\mathcal{C}_I(P_I(\sigma)) \nsubseteq S$, i.e., $C_I \nsubseteq S$, $q \notin F$ and $\sigma \in K$. Therefore, L is a subset of K and K contains all opaque sublanguages of $\mathcal{A}$. □

Therefore, by means of the augmented I-observer we can compute the supremal opaque sublanguage of $\mathcal{A}$, and by Propositions 7.1 and 7.3, the CSOEP can be solved by computing a maximal sublanguage of K that is prefix-closed, controllable and observable. The following theorem states that the CSOEP($\mathcal{A}, E_I, S, E_S, E_C$) is equivalent to the SCOP($\mathcal{A}, E_S, E_C, K$), i.e., based on the augmented I-observer locally optimal supervisors can be synthesized to enforce current-state opacity to a system $\mathcal{A}$.

Theorem 7.2. The set of solutions to the CSOEP($\mathcal{A}, E_I, S, E_S, E_C$) coincides with the set of solutions to the SCOP($\mathcal{A}_g, E_S, E_C, K$), where $\mathcal{A}_g$ is the augmented I-observer of $\mathcal{A}$ and $K = \{\sigma \in L(\mathcal{A}_g) | \delta_g(q_0, \sigma) \notin F\}$.

Proof: We prove this theorem by showing that the CSOEP($\mathcal{A}, E_I, S, E_S, E_C$) and SCOP($\mathcal{A}_g, E_S, E_C, K$) define the same supervisory control problem. By Proposition 7.1, we know that any sublanguage of a $\mathcal{A}$-opaque language is still $\mathcal{A}$-opaque. By Proposition 7.5, it is known that K is the supremal $\mathcal{A}$-opaque sublanguage of $\mathcal{A}$. Therefore, condition 1 in Definition 7.6 can be rephrased as "$L(Sup/\mathcal{A}) \subseteq K$", same as condition 1 in Definition 7.3. Moreover, $L(\mathcal{A}) = L(\mathcal{A}_g)$. Therefore, the CSOEP($\mathcal{A}, E_I, S, E_S, E_C$) and the SCOP($\mathcal{A}_g, E_S, E_C, K$) define the same supervisory control problem, and thus they share the same set of solutions. Namely, if Sup is a locally optimal supervisor of SCOP($\mathcal{A}_g, E_S$, E_C, K), then Sup is also a locally optimal supervisor of CSOEP($\mathcal{A}, E_I, S, E_S, E_C$), and vice versa. □

In other words, the CSOEP($\mathcal{A}, E_I, S, E_S, E_C$) can be solved by synthesizing a locally optimal supervisor of $\mathcal{A}_g$ with F being the set of forbidden states.

Example 7.7. By Theorem 7.2, the CSOEP in Example 7.4 is reduced to the problem of finding a locally optimal supervisor Sup for $\mathcal{A}_g$ such that state q_7 of $\mathcal{A}_g$ is not reachable in the controlled system. Applying the approach recalled in Section 7.2, first we construct the total controller in Fig. 7.10 and then, after removing all the states that contain forbidden state 7 (i.e., q_7 in $\mathcal{A}_g$), we obtain the all inclusive controller. In this case, removing the states in the dashed boxes in Fig. 7.10, the all inclusive controller is obtained. For simplicity, in the

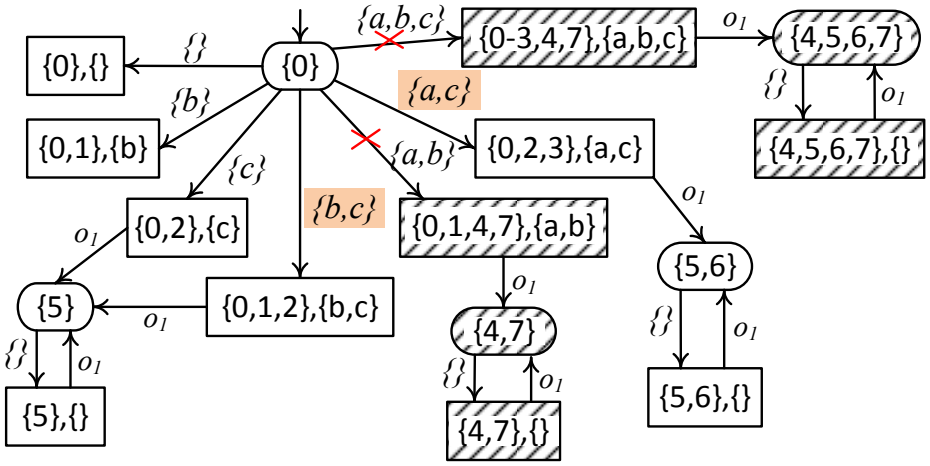

Fig. 7.10 Total controller of $\mathcal{A}$ in Fig. 7.9.

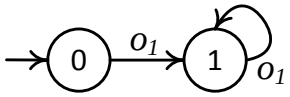

Fig. 7.11 Supervisors of the CSOEP in Example 7.4.

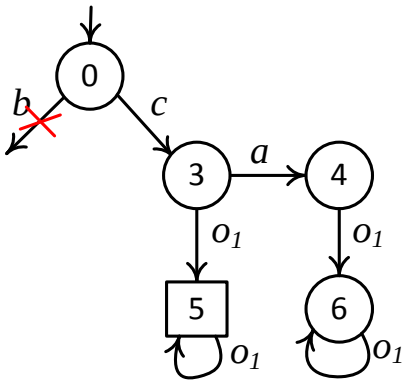

Fig. 7.12 $\mathcal{A}$ under the control of Sup_1.

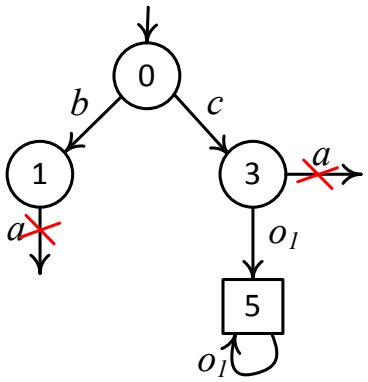

Fig. 7.13 $\mathcal{A}$ under control of Sup_2.

diagrams, we use i (with $i = 0, 1, \ldots, 7$) to denote state q_i in of the augmented I-observer and omit all uncontrollable events in the control decisions, e.g., decision $\{\}$ represents $\{o_1, o_2\}$, and so forth. Finally, at each step we choose a local maximal control decision and all locally optimal supervisors are computed. There are two locally optimal supervisors: Sup_1 and Sup_2 with the same automaton structure shown in Fig. 7.11. For Sup_1, $\Psi(0) = \{a, c, o_1, o_2\}$ and $\Psi(1) = \{o_1, o_2\}$; for Sup_2, $\Psi(0) = \{b, c\}$ and $\Psi(1) = \{o_1, o_2\}$. The controlled system under Sup_1 and Sup_2 is shown in Figs. 7.12 and 7.13, respectively. $\diamond$

7.5 Computational complexity analysis

According to the previous analysis, in the worst case the number of states of the augmented I-observer is $|X| \times 2^{|X|}$, where X is the set of states of $\mathcal{A}$. Since the complexity of solving the SCOP is $\mathcal{O}(|Q||E|2^{|Q|+|E_C|})$, where Q is the set of states of the augmented I-observer, the worst-case complexity of solving the CSOEP is $\mathcal{O}(|X|\times 2^{|X|}|E|2^{|X|\times 2^{|X|}+|E_C|})$, i.e., double exponential in the number of states of $\mathcal{A}$. It is clear that one exponential order comes from the construction of the augmented I-observer and the other one comes from the method adopted in this chapter to solve the SCOP.

We point out that in some cases (e.g., finding a near optimal supervisor [75, 82], online synthesizing the supervisor [78]), the complexity of solving the SCOP may decrease and consequently so would be the complexity of solving CSOEP.

Assuming the intruder has no knowledge of the supervisor, the proposed approach can solve the same problems in [37, 40, 74] with the same or lower complexity: exponential or double exponential, respectively. Consider the problem in [37] where $E_I \subseteq E_S = E$, $E_C \subseteq E_S$. The augmented I-observer contains all observations of the supervisor (i.e., $P_S(L(\mathcal{A})) = L(\mathcal{A}_g)$). Therefore, the augmented I-observer can be used to synthesize the supervisor directly. Moreover, due to $E_C \subseteq E_S$ the complexity of the proposed approach reduces to $\mathcal{O}(|X| \times 2^{|X|})$ same as the complexity of the approach in [37]. On the other hand, the complexity of solving the problem in [74] (where E_S and E_I are incomparable but $E_C \subseteq E_S$) using the proposed approach is $\mathcal{O}(2^{(|X|\times 2^{|X|})})$, lower than that of the approach in [74]. In addition, if either $E_S \subseteq E_I$ (or $E_I \subseteq E_S$) or $E_C \subseteq E_S$ holds, the supervisory synthesis problem considered in the chapter cannot be solved using the approaches in [39, 40, 44, 74].

7.6 Conclusion

In this chapter, we proposed a novel approach to solve the problem of current-state opacity enforcement in discrete event systems using finite automata. By constructing the augmented I-observer, all the strings that will leak the secret can be characterized. Based on the augmented I-observer, current-state opacity can be checked and a synthesis algorithm was provided to design locally optimal supervisors, without assuming the existence of containment relationships between E_I and E_S, or between E_C and E_S.

Preliminary work of this chapter has been published as:

Y. Tong, Z. Y. Ma, Z. W. Li, C. Seatzu, A. Giua. "Supervisory Enforcement of Current-State Opacity with Uncomparable Observations", In *Proceedings of the 13th International Workshop on Discrete Event Systems* (WODES'16), 2016: 313-318.

Chapter 8 Conclusion and Future Work

In the last chapter of the thesis conclusions for observation structures and opacity problems are drawn. In addition, directions on future research are also pointed out.

8.1 Concluding Remarks on Observation Structures

In the thesis, we have defined two classes of Petri net generators: labeled Petri nets with outputs (LPNOs) and adaptive labeled Petri nets (ALPNs), which extend the modeling power of Petri nets. Besides all the features of labeled Petri nets, LPNOs and ALPNs are capable of modeling systems with more complex observation structures. LPONs have an output function that provides an observation that is an arbitrary function of markings while the labeling function of an ALPN depends on the current marking, namely, the observation of an event depends on the evolution of the system.

The notion of observation equivalence is proposed to compare the modeling power. Two Petri net generators are observation equivalent if for any sequence of transitions the corresponding estimates of transition sequences in the two generators are identical. In other words, their observation structures provide the same information for reconstructing the evolution of the system. We show that among all Petri net generators LPNOs and ALPNs have the highest modeling power while for bounded systems, ALPNs are even more powerful than LPNOs. The conversion from other Petri net generators to ALPNs is meaningful, since if a method is applicable to ALPNs then it is also applicable to any other Petri net generator.

Finally, we presented a general procedure to convert an LPNO into an observation equivalent ALPN. Considering in literature there is a lot of methodology of system analysis using labeled Petri nets (LPNs), we also proposed an algorithm of converting the LPNO into an observation equivalent LPN, if there exists one.

8.2 Concluding Remarks on Opacity Problems

In the thesis we investigated opacity problems in discrete event systems modeled with labeled Petri net systems. The intruder knows the structure of the system but partially observes the behavior of the system. Based on its knowledge of the system and its observation, the intruder wants to know whether or not the secret has occurred. The system is said to be opaque if for any secret behavior there is another non-secret one that produces the same

observation.

We formalized the notion of current-state opacity (CSO), initial-state opacity (ISO) and language opacity (LO) in LPNs. In addition, a new notion of language-based opacity, called strict language opacity (SLO), is proposed to describe the situation where the intruder is interested in a subset of transitions. We showed that SLO generalizes the notion of LO and characterizes the class of secrets for which SLO and LO are identical. We clarified that B-initial-state opacity (B-ISO) defined in [5] generalizes the notion of ISO in Petri nets and automata.

We showed that the ISO verification problem can be reduced to the B-ISO verification problem, which is proven undecidable in LPNs. Thus, the ISO verification problem is not decidable either. We proved that the CSO verification problem is undecidable since it can be reduced to another undecidable problem — the Petri net language containment problem. Meanwhile, reducing the LO verification problem to the CSO verification problem, the LO verification problem is proven undecidable. Since in bounded LPNs, CSO verification, ISO verification and LO verification problems are decidable, we then focused on developing efficient and effective methods of verifying CSO, ISO and SLO in bounded LPNs.

We defined *exposable* and *weakly exposable* markings. Based on the notion of basis markings, if and only if for any observation there exists at least one weakly basis marking that is consistent with the observation, then the LPN system is current-state opaque. Therefore, to check CSO there is no need to construct the RG and its observer but the basis reachability graph, which is in general smaller than the RG, and its observer. If the secret is described by a set of generalized mutual exclusion constraints (GMECs), then CSO can be verified by solving a set of integer linear programming problems (ILPPs). Meanwhile, if the incidence matrix of the LPN is a totally unimodular, then the ILPPs can be relaxed to linear programming problems (LPPs). In particular, if no weakly exposable marking is contained in the secret, then CSO can be efficiently verified without solving LPPs. With constructing the extended observer, the proposed approach to verifying CSO can be applied to the case where the intruder only knows that the initial marking belongs to a set of markings. A sufficient and necessary condition is proposed to verify ISO. If no weakly exposable marking is in the secret, then the initial-state estimator of the BRG can be used to verify ISO. Otherwise, the modified BRG has to be constructed rather than the BRG. Finally, the verifier is proposed to analyze strict language opacity in bounded LPNs. Given the nets modeling the plant and the secret, the verifier synchronizes the plant and the secret with respect to observable transitions. It keeps track of both the sequences belonging and not belonging

to the secret. In particular, thanks to the notion of minimal explanations, to characterize the sequences there is no need to enumerate all of them. Therefore, the construction of the reachability graph is avoided. Compared with other methods in literature, the proposed approach is of lower complexity. Based on the proposed algorithms, a MATLAB toolbox is developed to verify CSO and ISO.

Given a system that is modeled with a DFA and that is not CSO with respect to a given secret, we proposed a method of synthesizing the locally optimal supervisors such that the controlled system is CSO. We considered the case where the intruder and the supervisor have incomparable observations and the sets of events observable and controllable by the supervisor is also incomparable. The notion of $\mathcal{A}$-opacity of a language is defined. It enables us to formalize the CSO enforcement problem assuming that intruder does not know or partially knows the supervisor. A finite structure, called *augmented I-observer*, is constructed to compute the supremal $\mathcal{A}$-opaque sublanguage of the system. It is shown that, the CSO enforcement problem of the system can be reduced to the supervisory control and observation problem of the augmented I-observer. Then using the methods in literature, the locally optimal supervisors are computed.

8.3 Future Work

The work in the thesis also points out several potential research directions. First, LPNOs and ALPNs provide an intuitive way to model systems with various kinds of sensors. Converting the LPNOs to ALPNs provides a way to analyze the system if the obtained ALPN is also an LPN. However, the converting procedure is of high complexity since one needs to compute the observer of the RG, and to solve the vertex coloring problem, which is NP-complete. Therefore, it would be interesting to develop a systematic way to analyze the system based on the information provided by the labeling function and output functions directly.

For the opacity verification problem, we only considered the problem in logical Petri net models, i.e., labeled Petri nets (LPNs), and there is no time factor or probability. Clearly, this is not the case in practice. Therefore, extending the notion of opacity to LPNOs/ALPNs, timed/stochastic Petri nets and Petri nets with probability would be another direction of our future research. For LPNOs and ALPNs, there are two types of observations. The problem would be how to extend the notion of opacity in the new Petri net models and how to efficiently verify opacity without converting LPNOs/ALPNs back to LPNs. For timed Petri nets, the intruder may refine its estimate taking the time factor into account.

Therefore, the problem would be more complicated. In Petri nets with probability, the firing of a transition has its probability. If we assume the intruder also knows the firing probability of all transitions, then its estimate would be with confidence. A system is opaque if for every observation, the occurrence of nonsecret behavior has higher probability than the occurrence of secret behavior.

The proposed approach to enforcing CSO does not consider the case where the intruder knows the supervisor. Otherwise, the intruder would make its estimate based on the structure of the controlled system rather than the original plant. In that case, the controlled system may not be opaque any more. Therefore, the supervisor has to take the knowledge of the intruder into account and further modify the controller, which leads to a game situation. In fact, such a scenario is quite common in practice because the two players could be military forces of two countries, or commercial competitors, or network hackers, etc. It would be meaningful to investigate the opacity enforcement problem considering different levels of intruder's knowledge.

Last but not the least, most work on opacity is carried out in a centralized way. On the contrary, it is more often that there is a group of intruders that may cooperate together to crack the secret, and/or the system is composed by modules distributed in different areas. Therefore, it would be also meaningful to study opacity problems in decentralized/distributed structures.

Reference

[1] BUSI N, GORRIERI R. A survey on non-interference with Petri nets[G] // Lectures on Concurrency and Petri Nets. [S.l.] : Springer, 2004 : 328 – 344.

[2] HADJ-ALOUANE N, LAFRANCE S, LIN F, et al. On the verification of intransitive noninterference in mulitlevel security[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 2005, 35(5) : 948 – 958.

[3] SHMATIKOV V. Probabilistic analysis of an anonymity system[J]. Journal of Computer Security, 2004, 12(3) : 355 – 377.

[4] MAZARÉ L. Using unification for opacity properties[C] // . 2004 : 165 – 176.

[5] BRYANS J, KOUTNY M, RYAN P. Modelling opacity using Petri nets[J]. Electronic Notes in Theoretical Computer Science, 2005, 121 : 101 – 115.

[6] LIN F. Opacity of discrete event systems and its applications[J]. Automatica, 2011, 47(3) : 496 – 503.

[7] JACOB R, LESAGE J, FAURE J. Overview of discrete event systems opacity: Models, validation, and quantification[J]. Annual Reviews in Control, 2016, 41 : 135 – 146.

[8] CASSANDRAS C, LAFORTUNE S. Introduction to Discrete Event Systems[M]. [S.l.] : Springer, 2008.

[9] GIUA A, SEATZU C. Observability of Place/Transition Nets[J]. IEEE Trans. Autom. Control, 2002, 47(9) : 1424 – 1437.

[10] GIUA A, SEATZU C, BASILE F. Observer-based state-feedback control of timed Petri nets with deadlock recovery[J]. IEEE Trans. Autom. Control, 2004, 49(1) : 17 – 29.

[11] GIUA A, SEATZU C, CORONA D. Marking estimation of Petri nets with silent transitions[J]. IEEE Trans. Autom. Control, 2007, 52(9) : 1695 – 1699.

[12] CABASINO M P, GIUA A, SEATZU C. Diagnosis using labeled Petri nets with silent or undistinguishable fault events[J]. IEEE Trans. Sys. Man Cybern., Syst., 2013, 43(2) : 345 – 355.

[13] USHIO T, ONISHI I, OKUDA K. Fault detection based on Petri net models with faulty behaviors[C] // Proc. IEEE Conference on Systems, Man, and Cybernetics. 1998 : 113 – 118.

[14] BOURIJ A, KOENIG D. An original Petri net state estimation by a reduced Luenberger observer[C] // Proc. American Control Conference : Vol 3. 1999 : 1986 – 1989.

[15] CHUNG S L. Diagnosing PN-based models with partial observable transitions[J]. International Journal of Computer Integrated Manufacturing, 2005, 18(2-3) : 158 – 169.

[16] RU Y, HADJICOSTIS C. Fault diagnosis in discrete event systems modeled by partially observed

Petri nets[J]. Discrete Event Dynamic Systems, 2009, 19(4) : 551 – 575.

[17] LEFEBVRE D. Diagnosis with Petri nets according to partial events and states observation[C] // IFAC Fault Detection, Supervision and Safety of Technical Processes : Vol 8. 2012 : 1244 – 1249.

[18] LEFEBVRE D. On-line fault diagnosis with partially observed Petri nets[J]. IEEE Trans. on Autom. Control, 2014, 59(7) : 1919 – 1924.

[19] SABOORI A, HADJICOSTIS C. Notions of security and opacity in discrete event systems[C] // Proceedings of the 46th IEEE Conference on Decision and Control. 2007 : 5056 – 5061.

[20] SABOORI A, HADJICOSTIS C N. Verification of initial-state opacity in security applications of DES[C] // Proceedings of the 9th International Workshop on Discrete Event Systems. 2008 : 328 – 333.

[21] SABOORI A, HADJICOSTIS C N. Verification of K-Step Opacity and Analysis of Its Complexity[J/OL]. IEEE Transactions on Automation Science and Engineering, 2011, 8(3) : 549 – 559. http://dx.doi.org/10.1109/TASE.2011.2106775.

[22] WU Y, LAFORTUNE S. Comparative analysis of related notions of opacity in centralized and coordinated architectures[J]. Discrete Event Dynamic Systems, 2013, 23(3) : 307 – 339.

[23] CASSEZ F, DUBREIL J, MARCHAND H. Synthesis of opaque systems with static and dynamic masks[J]. Formal Methods in System Design, 2012, 40(1) : 88 – 115.

[24] BRYANS J, KOUTNY M, MAZARÉ L, et al. Opacity generalised to transition systems[J]. International Journal of Information Security, 2008, 7(6) : 421 – 435.

[25] CASSEZ F. The dark side of timed opacity[G] // Advances in Information Security and Assurance. [S.l.] : Springer, 2009 : 21 – 30.

[26] SABOORI A, HADJICOSTIS C. Probabilistic current-state opacity is undecidable[C] // Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems. 2010 : 477 – 483.

[27] JACOB R, LESAGE J J, FAURE J M. Overview of Discrete Event Systems Opacity: models, validation, and quantification[J]. Annual Reviews in Control, 2016, 41 : 135 – 146.

[28] SABOORI A, HADJICOSTIS C. Verification of initial-state opacity in security applications of discrete event systems[J]. Information Sciences, 2013, 246 : 115 – 132.

[29] BADOUEL E, BEDNARCZYK M, BORZYSZKOWSKI A, et al. Concurrent secrets[J]. Discrete Event Dynamic Systems, 2007, 17(4) : 425 – 446.

[30] FALCONE Y, MARCHAND H. Enforcement and validation (at runtime) of various notions of opacity[J]. Discrete Event Dynamic Systems, 2015, 25(4) : 531 – 570.

[31] CASSEZ F, DUBREIL J, MARCHAND H. Dynamic observers for the synthesis of opaque systems[C] // International Symposium on Automated Technology for Verification and Analysis.

2009 : 352 – 367.

[32] ZHANG B, SHU S L, LIN F. Polynomial algorithms to check opacity in discrete event systems[C] // Proceedings of the 24th Chinese Control and Decision Conference. 2012 : 763 – 769.

[33] MA Z, LI Z, GIUA A. Design of Optimal Petri Net Controllers for Disjunctive Generalized Mutual Exclusion Constraints[J]. IEEE Transactions on Automatic Control, 2015, 60(7) : 1774 – 1785.

[34] CHEN Y, LI Z, BARKAOUI K, et al. On the enforcement of a class of nonlinear constraints on Petri nets[J]. Automatica, 2015, 55 : 116 – 124.

[35] CABASINO M P, GIUA A, SEATZU C. Diagnosability of Discrete-Event Systems Using Labeled Petri Nets[J/OL]. IEEE Transactions on Automation Science and Engineering, 2014, 11(1) : 144 – 153. http://dx.doi.org/10.1109/TASE.2013.2289360.

[36] TAKAI S, OKA Y. A formula for the supremal controllable and opaque sublanguage arising in supervisory control[J]. SICE Journal of Control, Measurement, and System Integration, 2008, 1(4) : 307 – 311.

[37] DUBREIL J, DARONDEAU P, MARCHAND H. Supervisory control for opacity[J]. IEEE Transactions on Automatic Control, 2010, 55(5) : 1089 – 1100.

[38] BEN-KALEFA M, LIN F. Supervisory control for opacity of discrete event systems[C/OL] // Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing. 2011 : 1113 – 1119. http://dx.doi.org/10.1109/Allerton.2011.6120292.

[39] SABOORI A, HADJICOSTIS C N. Opacity-enforcing supervisory strategies via state estimator constructions[J]. IEEE Transactions on Automatic Control, 2012, 57(5) : 1155 – 1165.

[40] YIN X, LAFORTUNE S. A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems[C] // Proceedings of the 2015 American Control Conference. 2015 : 377 – 383.

[41] WU Y C, LAFORTUNE S. Synthesis of insertion functions for enforcement of opacity security properties[J]. Automatica, 2014, 50(5) : 1336 – 1348.

[42] WU Y C, LAFORTUNE S. Synthesis of opacity-enforcing insertion functions that can be publicly known[C] // 2015 IEEE 54th Annual Conference on Decision and Control. 2015 : 3506 – 3513.

[43] WU Y C, LAFORTUNE S. Synthesis of optimal insertion functions for opacity enforcement[J]. IEEE Transactions on Automatic Control, 2016, 61(3) : 571 – 584.

[44] DUBREIL J, DARONDEAU P, MARCHAND H. Opacity enforcing control synthesis[C] // Proceedings of the 9th International Workshop on Discrete Event Systems,. 2008 : 28 – 35.

[45] CASSANDRAS C, LAFORTUNE S. Introduction to discrete event systems[M]. [S.l.] : Springer, 2008.

[46] MURATA T. Petri nets: Properties, analysis and applications[J]. Procedings of the IEEE, 1989,

77(4) : 541 – 580.

[47] SIPSER M. Introduction to the Theory of Computation : Vol 2[M]. [S.l.] : Thomson Course Technology Boston, 2006.

[48] GAREY M R, JOHNSON D S. Computers and intractability : Vol 29[M]. [S.l.] : wh freeman New York, 2002.

[49] CABASINO M, GIUA A, SEATZU C. Fault detection for discrete event systems using Petri nets with unobservable transitions[J]. Automatica, 2010, 46(9) : 1531 – 1539.

[50] FANTI M, MANGINI A, UKOVICH W. Fault Detection by Labeled Petri Nets in Centralized and Distributed Approaches[J/OL]. IEEE Transactions Automation Science Engineering, 2013, 10(2) : 392 – 404. http://dx.doi.org/10.1109/TASE.2012.2203596.

[51] CASSEZ F, TRIPAKIS S. Fault Diagnosis with Dynamic Observers[C] // Proceedings of 9th International Workshop on Discrete Event Systems. 2008 : 212 – 217.

[52] USHIO T, TAKAI S. Supervisory control of discrete event systems modeled by Mealy automata with nondeterministic output functions[C] // 2009 American Control Conference. 2009 : 4260 – 4265.

[53] WANG W, GIRARD A R, LAFORTUNE S, et al. On codiagnosability and coobservability with dynamic observations[J]. IEEE Trans. Autom. Control, 2011, 56(7) : 1551 – 1566.

[54] CARVALHO L K, BASILIO J C, MOREIRA M V. Robust diagnosis of discrete event systems against intermittent loss of observations[J]. Automatica, 2012, 48(9) : 2068 – 2078.

[55] DIESTEL R. Graph theory[M]. New York : Springer, 2006.

[56] JENSEN T R, TOFT B. Graph coloring problems : Vol 39[M]. [S.l.] : John Wiley & Sons, 2011.

[57] LIN F, WONHAM W M. On observability of discrete-event systems[J]. Information sciences, 1988, 44(3) : 173 – 198.

[58] REUTENAUER C. The mathematics of Petri nets[M]. [S.l.] : Prentice-Hall, Inc., 1990.

[59] CABASINO M, GIUA A, SEATZU C. Fault detection for discrete event systems using Petri nets with unobservable transitions[J]. Automatica, 2010, 46(9) : 1531 – 1539.

[60] CABASINO M, GIUA A, POCCI M, et al. Discrete event diagnosis using labeled Petri nets. An application to manufacturing systems[J]. Control Engineering Practice, 2011, 19(9) : 989 – 1001.

[61] GIUA A, SEATZU C, CORONA D. Marking Estimation of Petri Nets With Silent Transitions[J]. IEEE Transactions on Automatic Control, 2007, 52(9) : 1695 – 1699.

[62] CABASINO M, GIUA A, SEATZU C. Diagnosability of discrete-event systems using labeled Petri nets[J]. IEEE Transactions on Automation Science and Engineering, 2014, 11(1) : 144 – 153.

[63] MA Z, TONG Y, LI Z W, et al. Basis Marking Representation of Petri Net Reachability Spaces and Its Application to the Reachability Problem[J/OL]. IEEE Transactions on Automatic Control

(to appear), 2017, 62(3). http://dx.doi.org/10.1109/TAC.2016.2574120.

[64] GIUA A, DICESARE F, SILVA M. Generalized mutual exclusion contraints on nets with uncontrollable transitions[C] // Proc. of the 1992 IEEE International Conference on Systems, Man and Cybernetics. 1992 : 974 – 979 vol.2.

[65] MOODY J, ANTSAKLIS P. Petri net supervisors for DES with uncontrollable and unobservable transitions[J]. IEEE Transactions on Automatic Control, 2000, 45(3) : 462 – 476.

[66] IORDACHE M V, ANTSAKLIS P J. Petri net supervisors for disjunctive constraints[C] // Proceedings of the 2007 American Control Conference. 2007 : 4951 – 4956.

[67] YE J, LI Z, GIUA A. Decentralized Supervision of Petri Nets With a Coordinator[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2015, 45(6) : 955 – 966.

[68] MA Z Y, LI Z W, GIUA A. Design of Optimal Petri Net Controllers for Disjunctive Generalized Mutual Exclusion Constraints[J]. IEEE Transactions on Automatic Control, 2015, 60(7) : 1774 – 1785.

[69] SCHRIJVER A. Theory of linear and integer programming[M]. [S.l.] : John Wiley & Sons, 1998.

[70] MAHULEA C, SEATZU C, CABASINO M P, et al. Fault diagnosis of discrete-event systems using continuous Petri nets[J]. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 2012, 42(4) : 970 – 984.

[71] TONG Y. Matlab Tool for Verification of State-Based Opacity[J], .

[72] CABASINO M P, GIUA A, MARCIAS L, et al. A comparison among tools for the diagnosability of discrete event systems[C] // Proceedings of the 2012 IEEE International Conference on Automation Science and Engineering. 2012 : 218 – 223.

[73] RAMADGE P J G, WONHAM W M. The control of discrete event systems[J]. Proceedings of the IEEE, 1989, 77(1) : 81 – 98.

[74] TONG Y, MA Z, LI Z, et al. Supervisory enforcement of current-state opacity with uncomparable observations[C/OL] // Proceedings of the 13th International Workshop on Discrete Event Systems. 2016 : 313 – 318. http://dx.doi.org/10.1109/WODES.2016.7497865.

[75] HEYMANN M, LIN F. On-line control of partially observed discrete event systems[J]. Discrete Event Dynamic Systems, 1994, 4(3) : 221 – 236.

[76] HADJ-ALOUANE N, LAFORTUNE S, LIN F. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation[J]. Discrete Event Dynamic Systems, 1996, 6(4) : 379 – 427.

[77] RU Y, CABASINO M P, GIUA A, et al. Supervisor synthesis for discrete event systems under partial observation and arbitrary forbidden state specifications[J]. Discrete Event Dynamic Systems, 2014, 24(3) : 275 – 307.

[78] CAI K, ZHANG R, WONHAM W. Relative observability of discrete-event systems and its supremal sublanguages[J]. IEEE Transactions on Automatic Control, 2015, 60(3) : 659 – 670.

[79] YIN X, LAFORTUNE S. Synthesis of Maximally Permissive Supervisors for Partially-Observed Discrete-Event Systems[J]. IEEE Transactions on Automatic Control, 2016, 61(5) : 1239 – 1254.

[80] YIN X, LAFORTUNE S. A Uniform Approach for Synthesizing Property-Enforcing Supervisors for Partially-Observed Discrete-Event Systems[J/OL]. IEEE Transactions on Automatic Control, 2016, 61(8) : 2140 – 2154. http://dx.doi.org/10.1109/TAC.2015.2484359.

[81] CHO H, MARCUS S I. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation[J]. Mathematics of Control, Signals, and Systems (MCSS), 1989, 2(1) : 47 – 69.

[82] USHIO T. On-line control of discrete event systems with a maximally controllable and observable sublanguage[J]. IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, 1999, 82(9) : 1965 – 1970.

[83] RAMADGE P J, WONHAM W M. The control of discrete event systems[J]. Proceedings of the IEEE, 1989, 77(1) : 81 – 98.

[84] RAMADGE P J, WONHAM W M. Supervisory control of a class of discrete event processes[J]. SIAM journal on control and optimization, 1987, 25(1) : 206 – 230.

[85] SAMPATH M, SENGUPTA R, LAFORTUNE S, et al. Diagnosability of discrete-event systems[J]. IEEE Trans. Autom. Control, 1995, 40(9) : 1555 – 1575.

[86] SAMPATH M, SENGUPTA R, LAFORTUNE S, et al. Failure diagnosis using discrete-event models[J]. IEEE Trans. Control Syst. Technol, 1996, 4(2) : 105 – 124.

[87] LIN F, WONHAM W M. On observability of discrete-event systems[J]. Information sciences, 1988, 44(3) : 173 – 198.

[88] BASILE F, CHIACCHIO P, TOMMASI G D. An efficient approach for online diagnosis of discrete event systems[J]. IEEE Trans. Autom. Control, 2009, 54(4) : 748 – 759.

[89] GENC S, LAFORTUNE S. Distributed diagnosis of place-bordered Petri nets[J]. IEEE Trans. Autom. Sci. Eng., 2007, 4(2) : 206 – 219.

[90] FANTI M P, SEATZU C. Fault diagnosis and identification of discrete event systems using Petri nets[C] // 9th International Workshop on Discrete Event Systems (WODES), 2008.. 2008 : 432 – 435.

[91] RAMÍREZ-TREVIÑO A, RIVERA-RANGEL I, LÓPEZ-MELLADO E. Observability of discrete event systems modeled by interpreted Petri nets[J]. IEEE Trans. Robot. Autom., 2003, 19(4) : 557 – 565.

[92] RU Y, HADJICOSTIS C. Sensor selection for structurally observable in discrete event systems

modeled by partially observable Petri nets[J]. IEEE Trans. Autom. Control, 2010, 55(8) : 1751 – 1764.

[93] RU Y, HADJICOSTIS C N. State estimation in discrete event systems modeled by labeled Petri nets[C] // IEEE Conference on Decision and Control. 2006 : 6022 – 6027.

[94] LI Z, WU N, ZHOU M. Deadlock control of automated manufacturing systems based on Petri nets-A literature review[J]. IEEE Trans. Syst. Man Cybern. Part C, 2012, 42(4) : 437 – 462.

[95] LIN F. Diagnosability of discrete event systems and its applications[J]. Discrete Event Dynamic Systems, 1994, 4(2) : 197 – 212.

[96] RAMÍREZ-TREVIÑO A, RUIZ-BELTRÁN E, RIVERA-RANGEL I, et al. Online fault diagnosis of discrete event systems. A Petri net-based approach[J]. IEEE Trans. Autom. Sci. Eng., 2007, 4(1) : 31 – 39.

[97] CABASINO M P, GIUA A, LAFORTUNE S, et al. A new approach for diagnosability analysis of Petri nets using verifier nets[J]. IEEE Trans. Autom. Control, 2012, 57(12) : 3104 – 3117.

[98] GIUA A, DICESARE F. Blocking and controllability of Petri nets in supervisory control[J]. IEEE Trans. Autom. Control, 1994, 39(4) : 818 – 823.

[99] REITER M, RUBIN A. Crowds: Anonymity for web transactions[J]. ACM Transactions on Information and System Security, 1998, 1(1) : 66 – 92.

[100] SABOORI A. Verification and enforcement of state-based notions of opacity in discrete event systems[D]. [S.l.] : University of Illinois at Urbana-Champaign, 2011.

[101] SABOORI A, HADJICOSTIS C. Current-State Opacity Formulations in Probabilistic Finite Automata[J]. IEEE Transactions on Automatic Control, 2014, 59(1) : 120 – 133.

[102] SABOORI A, HADJICOSTIS C N. Opacity-enforcing supervisory strategies for secure discrete event systems[C] // Proceedings of the 47th IEEE Conference on Decision and Control. 2008 : 889 – 894.

[103] MAHULEA C, SEATZU C, CABASINO M, et al. Observer design for untimed continuous Petri nets[C] // Proceedings of the 2009 American Control Conference. 2009 : 4765 – 4770.

[104] GIUA A, DICESARE F. Decidability and closure properties of weak Petri net languages in supervisory control[J]. IEEE transactions on automatic control, 1995, 40(5) : 906 – 910.

[105] SEATZU C, SILVA M, van SCHUPPEN J. Control of Discrete-Event Systems : Vol 433[M]. [S.l.] : Springer-Verlag London, 2013.

[106] PETERSON J. Petri net theory and the modeling of systems[M]. [S.l.] : Prentice Hall PTR, 1981.

[107] BÉRARD B, MULLINS J. Verification of information flow properties under rational observation[J]. arXiv preprint arXiv:1409.0871, 2014.

[108] KOBAYASHI K, HIRAISHI K. Verification of opacity and diagnosability for pushdown system-

s[J]. Journal of Applied Mathematics, 2013, 2013.

[109] CHÉDOR S. Diagnostic, opacité et test de conformité pour des systèmes récursifs[D]. [S.l.] : Université Rennes 1, 2014.

[110] BÉRARD B, CHATTERJEE K, SZNAJDER N. Probabilistic opacity for Markov decision processes[J]. Information Processing Letters, 2015, 115(1) : 52 – 59.

[111] WU Y. Verification and Enforcement of Opacity Security Properties in Discrete Event Systems[D]. [S.l.] : The University of Michigan, 2014.

[112] CORONA D, GIUA A, SEATZU C. Marking estimation of Petri nets with silent transitions[C] // Proceedings of the 43rd IEEE Conference on Decision and Control : Vol 1. 2004 : 966 – 971.

[113] GIUA A, SEATZU C. Fault detection for discrete event systems using Petri nets with unobservable transitions[C] // Proceedings of the 44th IEEE Conference on Decision and Control, 2005 European Control Conference. 2005 : 6323 – 6328.

[114] LI Z W, ZHOU M C. Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems[J]. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, 2004, 34(1) : 38 – 51.

[115] CASSEZ F. A note on fault diagnosis algorithms[C] // Proceedings of the 48th IEEE Conference on and 28th Chinese Control Conference on Decision and Control. 2009 : 6941 – 6946.

[116] TSITSIKLIS J N. On the control of discrete-event dynamical systems[J]. Mathematics of Control, Signals and Systems, 1989, 2(2) : 95 – 107.

[117] WU Y C, LAFORTUNE S. Enforcement of opacity properties using insertion functions[C] // Decision and Control (CDC), 2012 IEEE 51st Annual Conference on. 2012 : 6722 – 6728.

Acknowledgement

This thesis represents not only the work during my doctoral studies, it is a milestone in my life time. I'm so lucky and grateful to have friendship, support, encouragement and guidance from so many people on the journey to my PhD.

First and foremost, I would like to express the deepest appreciation to my supervisors Prof. Alessandro Giua and Prof. Zhiwu Li, for their unwavering support and guidance. In particular, Prof. Giua dedicated a lot of time to guiding me. I benefited a lot, more than academically, from the experience of working with him. His kindness, work style, academic rigour and enthusiasm would affect me for a lifetime. Prof. Li gave me trust and precious opportunities of studying in the University of Cagliari. They are both my life mentors who taught me how to eagerly and humbly pursue the knowledge of, especially, but not limited to Discrete Event Systems.

Special thanks to Prof. Carla Seatzu for her patience, kindness, and our fruitful collaboration. I would like to thank Mr. Gianluca Mereu, Dr. Mauro Franceschelli, Dr. Alessandro Pilloni, and Ms. Graziana Cavone for their help and hospitality during my stay in the University of Cagliari. With them I never felt as an outsider. I would also like to thank Prof. W. M. Wonham and Prof. Kamel Barkaoui for their advice, encouragement and friendship.

I would like to greatly thank Dr. Xiuyan Zhang and Ms. Yunfang Wang for her companion. Thanks for their love, encouragement, and the shopping time we spent together. I would like to express my gratitude to Dr. Ziyue Ma and Mr. Zhou He. I thank Dr. Ziyue Ma for providing me indispensable advice, and inspiration on different aspects of my research. Mr. Zhou He and I grew together from newbies to mature researchers. With him I have never felt lonely on my PhD journey.

Last but not least, I ineffably indebted to my parents, Mr. Zhilin Tong and Mrs. Xiuhua Yao, for their constant love and support. Whenever I felt confused, happy and wanted to falter, they were always there and gave me the strongest encouragement and support but asking for nothing. Without them, I could not have focused on my study.

Biography

1. Basics

Yin TONG (female) was born in Nanchong, Sichuan in August 1989. She received Bachelor Degree in Automation from Xidian University, Xi'an, China, in 2012. Since 2012 she has been a Ph.D student of the School of Electro-Mechanical Engineering of Xidian University majored in Electro-Mechanical Engineering, and co-tutored by Prof. Dr. Zhiwu Li and Prof. Dr. Alessandro Giua. Since 2014, she has been in the program of joint supervision of thesis between Xidian University and the University of Cagliari, Italy, and also registered as a Ph.D student of the Department of Electrical and Electronic Engineering of the University of Cagliari.

2. Education Background

2008.08 $\sim$ 2012.07, Xidian University, B.S. in Automation
2012.08 $\sim$ present, Xidian University, Ph.D student in Electro-Mechanical Engineering
2014.11 $\sim$ present, the University of Cagliari, Ph.D student in Electronic and Computer Engineering

3. Academic Publications During Ph.D

3.1 Journal Publications

[1] **Y. Tong**, Z. W. Li, A. Giua. "On the Equivalence of Observation Structures for Petri Net Generators", *IEEE Transactions on Automatic Control*, Sept. 2016, 60(9): 2448-2462. (SCI: 000382686800009, EI: 16253316, JCR Q1)

[2] **Y. Tong**, Z. W. Li, C. Seatzu, A. Giua. "Verification of State-Based Opacity in Petri Nets", *IEEE Transactions on Automatic Control*, Jun. 2017, 62(6). Early access available online. DOI: 10.1109/TAC.2016.2620429. (SCI Journal, JCR Q1)

[3] **Y. Tong**, Z. W. Li, C. Seatzu, A. Giua. "Decidability of Opacity Problems in Labeled Petri Nets", *Automatica*, Jun. 2017, 80: 48-53. Early access available online. DOI: 10.1016/j.automatica.2017.01.013. (SCI Journal, JCR Q1)

[4] Z. Y. Ma, **Y. Tong**, Z. W. Li, A. Giua. "Basis Marking Representation of Petri

Net Reachability Spaces and Its Application to the Reachability Problem", *IEEE Transactions on Automatic Control*, Mar. 2016, 62(3): 1078-1093.
(SCI: 000395924300005, JCR Q1)

[5] Z. Y. Ma, **Y. Tong**. "Supervisor Synthesis in Petri Nets Based on Basis Marking Graphs", *Journal of Xidian University*, Dec. 2016, 43(6): 68-73.
(EI: 20165203176180)

3.2 Conference Publications

[1] **Y. Tong**, Z. W. Li, A. Giua. "General Observation Structures for Petri Nets", In *Proceedings of the 18th IEEE International Conference on Emerging Technologies and Factory Automation* (ETFA'13), 2013. (EI: 20135217133797)

[2] **Y. Tong**, Z. W. Li, A. Giua. "Observation Equivalence of Petri Net Generators", In *Proceedings of the 12th International Workshop on Discrete Event Systems* (WODES'14), 2014: 338 – 343. (EI: 20154501491534)

[3] **Y. Tong**, Z. W. Li, C. Seatzu, A. Giua. "Verification of Current-State Opacity Using Petri Nets", In *Proceedings of the 34th American Control Conference* (ACC'15), 2015: 1935-1940. (SCI: 20153701252212, EI: 20153701252212)

[4] **Y. Tong**, Z. W. Li, C. Seatzu, A. Giua. "Verification of Initial-State Opacity in Petri Nets", In *Proceedings of the 54th IEEE Conference on Decision and Control* (CDC'15), 2015: 344-349. (EI: 20161402197003)

[5] **Y. Tong**, Z. Y. Ma, Z. W. Li, C. Seatzu, A. Giua. "Supervisory Enforcement of Current-State Opacity with Uncomparable Observations", In *Proceedings of the 13th International Workshop on Discrete Event Systems* (WODES'16), 2016: 313-318. (SCI: 000382860200048, EI: 16104398)

[6] **Y. Tong**, Z. Y. Ma, Z. W. Li, C. Seatzu, A. Giua, "Verification of Language-Based Opacity in Petri Nets Using Verifier", the 35th American Control Conference (ACC'16), 2016: 757-763. (EI: 16194270)

3.3 Participation in Academic Programs

[1] General project of National Natural Science Foundation of China under Grant No. 61472295 entitled "Supervision and Reconfiguration in Discrete Event Systems", 2015.1-2018.12.

[2] General project of National Natural Science Foundation of China under Grant No. 61603285 entitled "Optimal Design of Petri Net Supervisors by Nonpure

Net Structures", 2017.1-2019.12.

[3] Youth fund of National Natural Science Foundation of China under Grant No. 51305325 entitled "Siphon Based Deadlock Control and Performance Analysis", 2014.1-2016.12.

[4] Youth fund of National Natural Science Foundation of China under Grant No. 61304050 entitled "On Intrinsically Live Structure of Generalized Petri Nets Modeling Flexible Manufacturing Systems", 2014.1-2016.12.

[5] Youth fund of National Natural Science Foundation of China under Grant No. 61304051 entitled "Robust Deadlock Control for Flexible Manufacturing Systems", 2014.1-2016.12.

[6] Youth fund of National Natural Science Foundation of China under Grant No. 61403296 entitled "Distributed Control of Automated Manufacturing Systems with Petri Nets", 2015.1-2017.12.

[7] Youth fund of National Natural Science Foundation of China under Grant No. 61403297 entitled "Theory and Algorithms of Output Feedback Robust Tracking Model Predictive Control", 2015.1-2017.12.

[8] Youth fund of National Natural Science Foundation of China under Grant No. 61472295 entitled "Supervision and Reconfiguration of Discrete Event Systems", 2015.1-2017.12.

[9] Youth fund of National Natural Science Foundation of China under Grant No. 61603285 entitled "On stability of dybamic reconfigurable discrete event systems based on R-TNCES", 2017.1-2019.12.