

A New Framework for Discrete-Event Systems

Other titles in Foundations and Trends® in Systems and Control

Adaptive Internal Models in Neuroscience

Mireille E. Broucke

ISBN: 978-1-68083-940-1

Finite-Time Stability Tools for Control and Estimation

Denis Efimov and Andrey Polyakov

ISBN: 978-1-68083-926-5

Generalized Coordination of Multi-robot Systems

Kazunori Sakurama and Toshiharu Sugie

ISBN: 978-1-68083-902-9

*Analysis and Control for Resilience of Discrete Event Systems: Fault
Diagnosis, Opacity and Cyber Security*

João Carlos Basilio, Christoforos N. Hadjicostis and Rong Su

ISBN: 978-1-68083-856-5

Learning-Based Control: A Tutorial and Some Recent Results

Zhong-Ping Jiang, Tao Bian and Weinan Gao

ISBN: 978-1-68083-752-0

*Synchronous Reinforcement Learning-Based Control for Cognitive Au-
tonomy*

Kyriakos G. Vamvoudakis and Nick-Marios T. Kokolakis

ISBN: 978-1-68083-744-5

A New Framework for Discrete-Event Systems

Kuize Zhang
University of Surrey
kuize.zhang@surrey.ac.uk

now

the essence of knowledge

Boston — Delft

AVA **Documents**
Document Sharing Hub

Foundations and Trends® in Systems and Control

Published, sold and distributed by:

now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:

now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

K. Zhang. *A New Framework for Discrete-Event Systems*. Foundations and Trends® in Systems and Control, vol. 10, no. 1-2, pp. 1–179, 2023.

ISBN: 978-1-63828-153-5

© 2023 K. Zhang

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, mechanical, photocopying, recording or otherwise, without prior written permission of the publishers.

Photocopying. In the USA: This journal is registered at the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923. Authorization to photocopy items for internal or personal use, or the internal or personal use of specific clients, is granted by now Publishers Inc for users registered with the Copyright Clearance Center (CCC). The 'services' for users can be found on the internet at: www.copyright.com

For those organizations that have been granted a photocopy license, a separate system of payment has been arranged. Authorization does not extend to other kinds of copying, such as that for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. In the rest of the world: Permission to photocopy must be obtained from the copyright owner. Please apply to now Publishers Inc., PO Box 1024, Hanover, MA 02339, USA; Tel. +1 781 871 0245; www.nowpublishers.com; sales@nowpublishers.com

now Publishers Inc. has an exclusive license to publish this material worldwide. Permission to use this content must be obtained from the copyright license holder. Please apply to now Publishers, PO Box 179, 2600 AD Delft, The Netherlands, www.nowpublishers.com; e-mail: sales@nowpublishers.com

Foundations and Trends® in Systems and Control

Volume 10, Issue 1-2, 2023

Editorial Board

Editors-in-Chief

Panos J. Antsaklis

University of Notre Dame
United States

Alessandro Astolfi

Imperial College London, United Kingdom
University of Rome "Tor Vergata", Italy

Elena Valcher

University of Padova
Italy



Documents
Document Sharing Hub

Editorial Scope

Topics

Foundations and Trends® in Systems and Control publishes survey and tutorial articles in the following topics:

- Control of:
 - Hybrid and Discrete Event Systems
 - Nonlinear Systems
 - Network Systems
 - Stochastic Systems
 - Multi-agent Systems
 - Distributed Parameter Systems
 - Delay Systems
- Filtering, Estimation, Identification
- Optimal Control
- Systems Theory
- Control Applications

Information for Librarians

Foundations and Trends® in Systems and Control, 2023, Volume 10, 4 issues. ISSN paper version 2325-6818. ISSN online version 2325-6826. Also available as a combined paper and online subscription.

Contents

1	Background and Motivation	4
1.1	Background	4
1.2	The First Motivation	5
1.3	The Second Motivation	7
1.4	The Third Motivation	11
1.5	Structure of the Monograph	12
1.6	Notation	12
1.7	Preliminaries on Decidability and Complexity	13
2	Labeled Finite-State Automata	15
2.1	The Models	15
2.2	Basic Tools — Concurrent Composition and Observer . . .	23
2.3	Verification of Inference-Based Properties	27
2.4	Verification of Concealment-Based Properties	47
2.5	An Open-Loop Property Enforcement Framework	61
3	A Decentralized Setting in Labeled Finite-State Automata	76
3.1	The Basic Tool — Concurrent Composition	77
3.2	Verification of Decentralized Detectability	79
3.3	Verification of Decentralized Diagnosability	90
3.4	Verification of Decentralized Predictability	98

3.5	An Open-Loop Framework for Enforcing Decentralized Versions of Strong Detectability, Diagnosability, and Predictability	102
4	Labeled Weighted Automata over Monoids	104
4.1	The Model	105
4.2	The Definitions of Strong and Weak Versions of Detectability	111
4.3	Three Basic Tools — Concurrent Composition, Observer, and Detector	115
4.4	Necessary and Sufficient Conditions for Strong and Weak Versions of Detectability	122
4.5	Illustrative Examples	130
5	Labeled Weighted Automata over the Monoid $(\mathbb{Q}^k, +)$	133
5.1	Computation of Three Basic Tools for $\mathcal{A}^{\mathbb{Q}^k}$	133
5.2	Illustrative Examples	160
5.3	Initial Exploration on an Open-Loop Framework for Enforcing Strong and Weak Versions of Detectability	165
5.4	Outlook on Diagnosability, Predictability, and Opacity	167
6	Outlook on Implementation of an Open-Loop Property Enforcement Framework in Labeled Petri nets and Labeled Timed Automata	168
	Acknowledgements	170
	References	171

A New Framework for Discrete-Event Systems

Kuize Zhang

University of Surrey, UK; kuize.zhang@surrey.ac.uk

ABSTRACT

Real-world problems are often formulated as diverse properties of different types of dynamical systems. Hence property verification and synthesis (i.e., enforcement) have been long-standing research interests. The motivations of writing this monograph lie in two aspects. First, we will develop an open-loop property enforcement framework for discrete-event systems. Second, we will propose a new model — labeled weighted automata over monoids.

The supervisory control framework initialized by Ramadge, Wonham, and Lin in the 1980s provides a closed-loop property enforcement framework for discrete-event systems which usually consist of discrete states and transitions between states caused by spontaneous occurrences of labeled (i.e., partially-observed) events. This framework can be fully realized in labeled finite-state automata (LFSAs). Plenty of theoretical and applied results under this framework have been obtained during the past three decades. However, there are several drawbacks in this framework which restrict the application of the framework to large-scale systems, e.g., all enforceable properties can be enforced in LFSAs in at least *exponential time*, showing that the enforcement algorithms in this framework do not scale well; this framework

Kuize Zhang (2023), “A New Framework for Discrete-Event Systems”, Foundations and Trends® in Systems and Control: Vol. 10, No. 1-2, pp 1–179. DOI: 10.1561/26000000028.

©2023 K. Zhang

cannot be fully realized in more complicated models such as labeled Petri nets and labeled timed automata, because supervisors/controllers in such models are generally not computable (with any complexity upper bound), which narrows the application range of this framework. In this monograph, we will develop an open-loop property enforcement framework for discrete-event systems which scales better and can be implemented in more models.

In order to implement this new framework, we develop a tool called *concurrent composition*, and use this tool to unify plenty of *inference-based* properties (e.g., detectability, diagnosability, predictability) and *concealment-based* properties (e.g., various notions of opacity) in discrete-event systems.¹ The negations of such properties are equivalently represented by the existence of special runs in the concurrent composition of two variants of a plant. Then, a property of interest can be enforced by choosing controllable events/transitions to disable in order to cut off all such runs violating the property. Our open-loop framework can be implemented in LFSAs in *polynomial time* for polynomially verifiable properties (e.g., strong detectability, diagnosability, predictability), and can also be fully realized (at least) in labeled Petri nets and labeled timed automata for decidable inference-based and concealment-based properties.

In the second aspect, we propose a new model called *labeled weighed automata over monoids* (LWAMs). LWAMs provide a natural generalization of LFSAs in the sense that each transition therein carries a weight from a monoid, the weight of a run (a sequence of consecutive transitions) is the product of the weights of the run's transitions. When weights are non-negative real numbers, they could be interpreted as the time

¹In the past, these inference-based properties were verified by using different methods, and based on two fundamental assumptions of deadlock-freeness (a plant will always run) and divergence-freeness (the running of a plant will always be eventually observed).

consumptions of the transitions' executions, so that LWAMs could be regarded as real-time systems. When weights are real vectors (hence the entries of the vectors could be negative), they can be interpreted as position deviations of a moving object along with the transitions' executions. We develop original techniques to compute three basic tools — *concurrent composition*, *observer*, and *detector* in LWAMs, and then design algorithms for verifying various notions of detectability. The research in LWAMs has just started. With these three tools, plenty of results in LFSAs obtained in the past three decades can be extended to LWAMs, including results on inference-based properties and concealment-based properties, as well as results obtained in the supervisory control framework. Our open-loop property enforcement framework, of course, can be fully implemented in LWAMs. Compared with LFSAs, LWAMs provide more accurate modeling scheme, hence have more applications. A challenging future direction lies in extending the formal verification and synthesis framework of cyber-physical systems from the core part of LFSAs-based to LWAMs-based.

1

Background and Motivation

1.1 Background

In the 1980s, P. Ramadge, W. Wonham, and F. Lin initialized the so-called *supervisory control framework* (Ramadge and Wonham, 1987; Lin and Wonham, 1988), which extends the analysis and synthesis framework from control systems (normally differential equations) to computer systems (formal languages) which are called *discrete-event systems* (DESs), where the counterparts of all kinds of (e.g., controllable or observable) subspaces in control systems are diverse sublanguages of formal languages. DESs usually consist of discrete states and transitions between states caused by spontaneous occurrences of labeled (aka partially-observed) events, and the formal languages of interest are the sets of label/output sequences generated by DESs. A transition is represented by the form $q_1 \xrightarrow{e(\sigma)} q_2$, indicating that when a DES is in state q_1 and event e occurs, the DES transitions to state q_2 ,¹ σ is the label/output of e , i.e., the observation when e occurs, particularly $\sigma = \epsilon$ ² implies e is unobservable. Hence DESs are autonomous (i.e., not

¹ q_2 need not be different from q_1 .

²As usual, ϵ denotes the empty string.

driven by external factors) and nonlinear (Wonham and Cai, 2019). The supervisory control framework provides a controller synthesis method in a *closed-loop manner*. It tracks a sequence of observed outputs generated by a given DES, does state estimation according to the outputs, and meanwhile synthesizes control policies (called a *supervisor*) to restrict the behavior of the DES such that the modified DES satisfies a property of interest that the original DES does not satisfy. Hence, one prerequisite is that the property of interest is decidable, i.e., there is an algorithm for verifying the property. The verification problem is also called the analysis problem. The synthesis problem is also called the enforcement problem, i.e., for a DES $\mathcal{S}$ and a property P , $\mathcal{S}$ does not satisfy P , one modifies $\mathcal{S}$ in order to make it enforce P .

1.2 The First Motivation

During the past three decades, plenty of interesting properties with their variants in DESs have been proposed, investigated, and applied to many different areas such as heating, ventilation, and air conditioning (HVAC), traffic networks, automated manufacturing, tracking of mobile agents in sensor networks, etc. It is exciting that so many results have been obtained and different properties have remarkably different physical meanings, e.g., detectability implies that one *can* determine the current and subsequent states using observed output sequences (Shu *et al.*, 2007), diagnosability implies that one *can* determine the past occurrences of faulty events, using the observed output sequences (Sampath *et al.*, 1995), state-based opacity implies that one *cannot* determine the visit to some secret state, also by using the observed output sequences (Saboori and Hadjicostis, 2007). However, different properties have been verified using different methods and it is not known whether there are essential differences between them, particularly from a mathematical point of view. A first motivation of writing this monograph is to *perform subtractions on* DESs, i.e., using a streamlined mathematical framework to unify as many as properties, although they have diverse physical meanings. The first target of the monograph is to unify all properties without essential differences into one mathematical framework. Note that such subtractions will not reduce the existing realms and realms of

results obtained in DESs, actually they will make the contents of DESs more *tidy*.

We develop a mathematical tool that we named *concurrent composition*³ to implement the first target. Intuitively speaking, the concurrent composition of two labeled systems $\mathcal{S}_l$ and $\mathcal{S}_r$ aggregates any pair of a run⁴ of $\mathcal{S}_l$ and a run of $\mathcal{S}_r$ with the same observation; the observations in any pair of their runs will be synchronized and their unobservable transitions will interleave. The concurrent composition will provide a unified mathematical framework for most *inference-based* properties and *concealment-based* properties. By inference-based we mean a property indicating that one can get further internal information from observations, e.g., detectability, diagnosability, predictability, etc. By concealment-based we mean a property indicating that one cannot get further internal information from observations, e.g., opacity. A preliminary work along this line in LFSAs refers to Zhang (2021a).

Because of the partially-observed feature of DESs, the properties therein can be naturally classified into the two basic categories of inference-based and concealment-based, other properties can be seen as variants of the properties in the two categories. In order to verify an inference-based property, we represent its *negation* as the existence of special runs in the corresponding concurrent composition, the rest is to check the existence of the special runs (see Section 2.3 and Zhang, 2021a). The advantages of this approach in LFSAs with respect to verification are two-fold. Firstly, it provides polynomial-time verification algorithms for most known inference-based properties except for those whose verification problems have been proven PSPACE-hard,⁵ e.g., weak detectability (Zhang, 2017), strong periodic D-detectability (Balun and Masopust, 2021b). Secondly, it does not depend on any assumption. Note that the widely-used verification algorithms in the literature for verifying inference-based properties such as the detector method for

³Its form in labeled finite-state automata (LFSAs, as in Definition 2.5) is shown in Definition 2.6.

⁴A run is a sequence of transitions in which the terminating state of each transition is the same as the starting state of its very next transition.

⁵It is widely conjectured that a PSPACE-hard problem cannot be solved by any polynomial-time algorithm (Sipser, 1996), see Page 13.

strong detectability (Shu and Lin, 2011b), the twin-plant method (Jiang *et al.*, 2001) and the verifier method (Yoo and Lafortune, 2002) for diagnosability, and the verifier method (Genc and Lafortune, 2009) for predictability, though also run in polynomial time, all depend on two fundamental assumptions of deadlock-freeness (also called liveness, which means that an automaton will always run) and divergence-freeness (i.e., an automaton has no reachable unobservable transition cycle, which means that the running of an automaton will always be eventually observed). The reason lies in the fact that these methods were used to verify the properties themselves but not their negations. See Section 2.3.4 for detailed analysis. In order to verify a concealment-based property in an LFSA $\mathcal{A}$, e.g., opacity, we first compute the concurrent composition of $\mathcal{A}$ and its *observer*⁶ (see Definition 2.7), and then check the reachability of some special state in the concurrent composition. Note that this idea of verification is directly derived from various definitions of opacity, and the derived algorithms are currently the most efficient (see Section 2.4).

In addition, with respect to verification, one major advantage of concurrent composition lies in that it can be extended to models that are more general than LFSAs, e.g., labeled weighted automata over monoids (Zhang, 2022), labeled Petri nets (Zhang and Giua, 2020b; Zhang *et al.*, 2020), and so on. The study on labeled weighted automata over monoids has just started. Welcome more and more researchers to join in this new research direction.

1.3 The Second Motivation

Before introducing a second motivation of writing this monograph, we recall the overall procedure of supervisor synthesis in the supervisory control framework (Ramadge and Wonham, 1987; Lin and Wonham, 1988; Wonham and Cai, 2019; Cassandras and Lafortune, 2008) (see Figure 1.1 for an illustration). Recall that the event set in a DES is an alphabet, i.e., a nonempty finite set E such that every sequence of elements of E has a unique decomposition of elements of E . For example,

⁶Actually the standard powerset/subset construction for determinizing a non-deterministic finite automaton with ε -transitions (Rabin and Scott, 1959; Sipser, 1996).

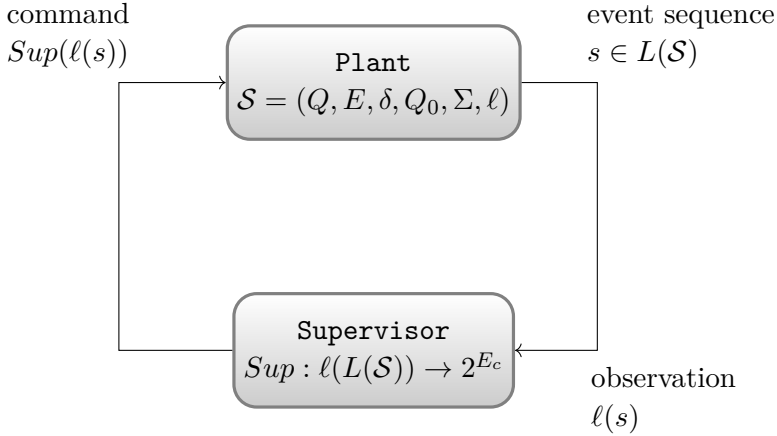


Figure 1.1: A sketch for supervisory control.

$\{a, b\}$ and $\{0, 01\}$ are alphabets, but $\{0, 00\}$ is not, because $000 = 0\ 00 = 00\ 0$. An event set must be an alphabet because this guarantees that every generated event sequence cannot have two different interpretations. Consider $\{0, 00\}$ as a counterexample, if 000 were generated, then there are two interpretations: (1) 0 was first generated and then 00 was generated, or (2) 00 was first generated and then 0 was generated. Also recall that an event set E can be partitioned into two disjoint subsets E_c and E_{uc} , denoted by $E = E_c \cup E_{uc}$, where E_c denotes the set of *controllable events* and E_{uc} the set of *uncontrollable events*. The occurrence of a controllable event can be forbidden, but the occurrence of an uncontrollable event cannot. Given a formal language L ⁷ and a DES G as one of its generators⁸ such that G does not satisfy a property P of interest, one tracks an observed output sequence γ generated by G , does state estimate SE_γ according to γ , and then uses SE_γ to compute a subset C_{SE_γ} of controllable events that are in the transitions starting from the states of SE_γ . A supervisor $S : L \rightarrow 2^{E_c}$ ⁹ is a mapping that sends an observed output sequence γ of L to $C_{SE_\gamma} (\subset E_c)$.

⁷Defined by a subset of E^* , where E^* denotes the set of finitely long strings of elements of E .

⁸The set of finitely long output sequences generated by G is equal to L .

⁹The powerset of E_c , i.e., $2^{E_c} = \{E'_c | E'_c \subset E_c\}$.

The supervisor works in this way: whenever the current-state estimate is SE_γ (no matter the current observed output sequence is γ or not, i.e., there may exist different γ, γ' such that $SE_\gamma = SE_{\gamma'}$), one dynamically disables several controllable events of C_{SE_γ} to restrict the behavior of G , so that the closed-loop system (G, S) satisfies the property P . From the procedure, one can see that if G has finitely many states, then the supervisor S usually can be fully computed even if L is infinite, because during computation one can partition L into a finite number of disjoint nonempty subsets such that two sequences $\gamma, \gamma' \in L$ belong to the same subset if and only if $SE_\gamma = SE_{\gamma'}$ (in this case, S send them to the same subset of E_c). In this sense, S is a *nondeterministic finite automaton* (see Definition 2.2). However, if G has infinitely many states, usually the supervisor S cannot be fully computed, i.e., the supervisor control cannot be fully realized. In DESs, the widely-used models such as finite automata, Petri nets, timed automata, etc., have finitely many events, because their event sets are always alphabets; however, Petri nets and timed automata may have infinitely many states. As a result, although the supervisory control framework is a methodology that owns abundant intension, it is somehow air-castle. For the models that are more complicated than finite automata, generally the supervisory control cannot be fully realized. *The second motivation of writing this monograph is to develop a new synthesis framework that is applicable to remarkably larger classes of models that can represent DESs.*

Our new property synthesis framework is open-loop, i.e., one does not supervise output sequences generated by a DES as is done in the supervisory control framework. The philosophy of our framework is as follows (see Figure 1.2 for an illustration): (1) the negation of a property of interest is equivalently represented by the existence of special runs in the corresponding concurrent composition of a variant of the plant and another variant of the plant, (2) controllable events/transitions¹⁰ are chosen to be disabled so that all these special runs violating the property will disappear. This synthesis method is quite simple and easily realizable on decidable properties of DESs, as there are only finitely

¹⁰A controllable transition is a transition whose event is controllable.

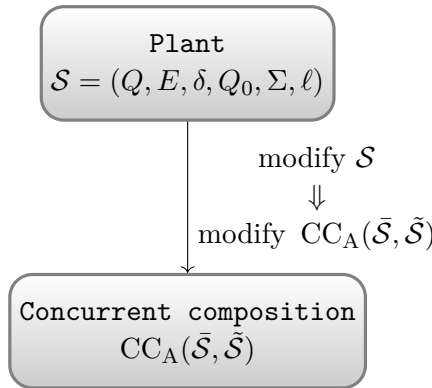


Figure 1.2: A sketch for the open-loop control.

many events. One can also easily check whether a property is enforceable: disabling all controllable events and then see whether the modified DES satisfies the property. If no, then the property of the DES cannot be enforceable in a large extent; otherwise, one can enforce the property by choosing several controllable events to disable according to specific scenarios. Of course, although for several DESs the property could be enforced after all controllable events being disabled, the remainder of such DESs might not be interesting any more because they might lose several interesting behaviors. Therefore, a better way is to choose as few as controllable events to disable. To be more refined, one can also choose concrete controllable transitions to disable instead of controllable events (in the latter coarser case, if one controllable event is disabled, then all transitions with the event will be disabled). In finite automata, there are finitely many transitions, so the synthesis method via choosing controllable transitions can be fully realized. However, for systems with infinitely many transitions such as unbounded Petri nets, the focus should be on controllable events or a finite subset of controllable transitions. To sum up, our open-loop framework will *perform additions* to DESs, because it provides a new property synthesis framework on dramatically larger classes of systems compared with the supervisory control framework. The second target of writing the monograph is to implement our open-loop property synthesis framework.

1.4 The Third Motivation

As mentioned above, DESs have LFSAs as their basic model. The supervisory control framework can be fully realized in LFSAs, but cannot be fully realized in models that are more general than LFSAs in general, because their observers are usually not computable, e.g., labeled timed automata and labeled Petri nets. A natural question to ask is: Are there a class of systems that are more general than LFSAs but the supervisory control framework can be fully realized in the class? This is almost equivalent to ask: Are there a class of systems that are more general than LFSAs but their observers are computable?

On the other hand, although LFSAs are the basic model of DESs, they do not show sufficiently accurate modeling. For example, when doing state estimation based on a sequence γ of observed labels, the time consumptions for the executions of unobservable transitions were usually assumed to be zero (Sampath *et al.*, 1995; Shu *et al.*, 2007). Timed automata are a natural generalization of finite automata in the sense that the executions of transitions are constrained by time intervals with rational endpoints. However, the observers of labeled timed automata are usually not computable.

Based on the above two points, it is very meaningful to find a class of systems that are more general than LFSAs but their observers are computable. It is challenging to do so. Consider a run $q_0 \xrightarrow{e_1/t_1} q_1 \xrightarrow{e_2/t_2} \dots \xrightarrow{e_n/t_n} q_n$, in which after each “/” there is a weight for the corresponding transition. If the weight of a transition therein is considered as its time consumption, then the weight of the run is equal to $\sum_{i=1}^n t_i$. This semantics is similar to that in timed automata. Differently, here we consider a general monoid $\mathfrak{M} = (T, \otimes, \mathbf{1})$, where $\otimes$ is an associative binary operation on T and $\mathbf{1} \in T$ is an identity element. We extend LFSAs in the sense that each of its transitions carries a weight in a monoid and the weight of a run is the product of the weights of the transitions of the run. When $\mathfrak{M}$ is specified as $(\mathbb{R}_{\geq 0}, +, 0)$, where $\mathbb{R}_{\geq 0}$ denotes the set of nonnegative real numbers as usual, the weights can represent the time consumptions of the corresponding transitions and then the extended LFSAs can represent a real-time system; while $\mathfrak{M}$ is specified as $(\mathbb{R}^n, +)$, where $\mathbb{R}^n$ denotes the set of n -dimensional

real vectors, the weights can represent position deviations along with the transitions. That is, the weights have diverse physical meanings. In Section 5, we will prove that the observers of this kind of extended LFSAs over the monoid $(\mathbb{Q}^n, +)$ are computable by developing original computing techniques, where $\mathbb{Q}^n$ denotes the set of n -dimensional rational vectors. A general theory of the extended LFSAs over monoids will be given in Section 4, where the new class of automata are called *labeled weighted automata over monoids* (LWAMs).

1.5 Structure of the Monograph

In Section 2, we show the implementation of our unified concurrent-composition framework for DESs modeled by LFSAs as well as our open-loop property synthesis framework, in a centralized setting. In Section 3 we show the implementation for LFSAs in a decentralized setting. In Section 4, we will propose the new model — LWAMs. We will formulate the basic tools of concurrent composition, observer, and detector for LWAMs, and use them to derive necessary and sufficient conditions for several strong versions of detectability and weak versions of detectability. Particularly in Section 5, for labeled weighted automata (LWAs) over the monoid $(\mathbb{Q}^n, +)$, we will develop original methods (that can be implemented algorithmically) to compute the three basic tools, and hence prove that the necessary and sufficient conditions obtained in Section 4 are algorithmically implementable. Thus, our concurrent-composition framework and open-loop property synthesis framework will be fully extended to LMAs over the monoid $(\mathbb{Q}^n, +)$. Section 6 shows a brief outlook on the implementation of the open-loop property enforcement framework in labeled Petri nets and labeled timed automata.

1.6 Notation

Symbols $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{Q}_{\geq 0}, \mathbb{R}$, and $\mathbb{R}_{\geq 0}$ denote the sets of nonnegative integers, integers, rational numbers, nonnegative rational numbers, real numbers, and nonnegative real numbers, respectively. $\mathbb{R}^n$ denotes the set of n -dimensional real column vectors. The symbol $(\cdot)^n$ also applies

to the other sets of numbers. 0_n denotes the n -dimensional column vector with all entries 0. $\llbracket m, n \rrbracket$ denotes the set of integers no less than m and no greater than n . A finite nonempty set Σ is called an *alphabet* if every sequence of elements of Σ is a unique sequence of elements of Σ . For example, $\{0, 00\}$ is not an alphabet since $000 = 0\ 00 = 00\ 0$. For an alphabet Σ , elements of Σ are called *letters*, Σ^* and Σ^ω are used to denote the set of *words/strings* (i.e., finite-length sequences of elements of Σ) over Σ including the empty word ϵ and the set of *configurations* (i.e., infinite-length sequences of elements of Σ) over Σ , respectively. $\Sigma^+ := \Sigma^* \setminus \{\epsilon\}$. For a word $s \in \Sigma^*$, $|s|$ stands for its length, and we set $|s'| = +\infty$ for all $s' \in \Sigma^\omega$. For $s \in \Sigma^+$ and $k \in \mathbb{N}$, s^k and s^ω denote the concatenations of k copies of s and infinitely many copies of s , respectively. Analogously, the concatenation of two languages L_1 and L_2 is defined by $L_1 L_2 := \{e_1 e_2 \mid e_1 \in L_1, e_2 \in L_2\}$, where $L_1, L_2 \subset \Sigma^*$. For a word (configuration) $s \in \Sigma^*(\Sigma^\omega)$, a word $s' \in \Sigma^*$ is called a *prefix* of s , denoted as $s' \sqsubset s$, if there exists another word (configuration) $s'' \in \Sigma^*(\Sigma^\omega)$ such that $s = s' s''$. In this case, s'' is called a *suffix* of s . For a set S , $|S|$ denotes its cardinality and 2^S its power set. Symbols $\subset$ and $\subsetneq$ denote the subset and strict subset relations, respectively. Symbol $_$ denotes an element that is not specified. For instance, consider $Q \times E \times Q$ with Q and E two nonempty sets, $(q, e, _) \in Q \times E \times Q$ denotes a triple of $Q \times E \times Q$ whose first entry is q , second entry is e , and third entry can be any element of Q .

1.7 Preliminaries on Decidability and Complexity

We recall basic concepts on decidability and complexity (see Hopcroft and Ullman, 1969; Sipser, 1996; Immerman, 1988, etc.). Given two sets $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{B} \subset \mathcal{A}$, a *decision problem* refers to whether there exists an *algorithm*¹¹ for determining whether a given $a \in \mathcal{A}$ belongs to $\mathcal{B}$. A decision problem is called *decidable* if an algorithm for solving this problem exists, and called *undecidable* otherwise. For example, the well-known Turing machine halting problem is undecidable. Decidable problems can be classified into different classes according the complexity

¹¹Defined by a *halting Turing machine*.

of the algorithms solving them. For example, P (resp., NP , $PSPACE$, $NPSPACE$, $EXPTIME$, $2-EXPTIME$) denotes the class of the problems solvable by polynomial-time (resp., nondeterministic polynomial-time, polynomial-space, nondeterministic polynomial-space, exponential-time, doubly exponential-time) algorithms. NL denotes the class of problems solvable by nondeterministic logarithmic-space algorithms. $coNL$, $coNP$, and $coNPSPACE$ denote the sets of problems whose complements belong to NL , NP , and $NPSPACE$, respectively. It is known that (Sipser, 1996; Immerman, 1988) $NL \subset P \subset NP \subset PSPACE \subset EXPTIME$, $P \subset coNP \subset PSPACE$, $NL = coNL$, and $PSPACE = NPSPACE = coNPSPACE$. It is also known that $NL \subsetneq PSPACE$ and $P \subsetneq EXPTIME$, but whether the rest of these containments are strict are long-standing open questions. It is widely conjectured all the other containments are strict.

A decision problem is called NP (resp., $coNP$, $PSPACE$, $EXPTIME$)-hard if every problem in NP (resp., $coNP$, $PSPACE$, $EXPTIME$) is polynomial time reducible to it. A problem is called X -complete if the problem belongs to the class X and is X -hard, where X can be P , NP , $coNP$, $PSPACE$, $EXPTIME$, etc. Hence there exists no polynomial-time algorithm for solving an NP (resp., $PSPACE$)-complete problem unless $P = NP$ (resp., $PSPACE$). There is no polynomial-time algorithm for solving an $EXPTIME$ -hard problem since $P \subsetneq EXPTIME$. A decision problem is called NL -hard if every problem in NL is logarithmic space reducible to it.

2

Labeled Finite-State Automata

Labeled finite-state automata (LFSAs) (Definition 2.5) are the most widely studied model of discrete-event systems (DESS). It was pointed out by Lin and Wonham (1988) that “Under certain restrictions, the existence problem for a supervisory controller was solved abstractly.” Plenty of related problems have been concretely solved in LFSAs later. An LFSA is obtained from a standard *nondeterministic finite automaton* (NFA) (Definition 2.2) in computation theory textbooks by (1) replacing the unique initial state by a subset of initial states, indicating that the real initial state could be any one in the subset, but is usually unknown; (2) removing the final states; and (3) adding a labeling function, showing the partially-observable feature of events.

2.1 The Models

The standard definition of *deterministic finite automaton* (DFA) is as follows, which can be found in standard computation theory textbooks, e.g., (Sipser, 1996):

Definition 2.1. A *deterministic finite automaton* (DFA) is a quintuple

$$(Q, \Sigma, \delta, q_0, F),$$

where

1. Q is a finite set of *states*,
2. Σ is an *alphabet*,
3. $\delta : Q \times \Sigma \rightarrow Q$ is the *transition function*,
4. $q_0 \in Q$ is the *initial state* (aka *start state*), and
5. $F \subset Q$ is the set of *final states* (aka *accept states*).

We extend the transition function $\delta : Q \times \Sigma \rightarrow Q$ to $\delta : Q \times \Sigma^* \rightarrow Q$ recursively as follows: for all $q \in Q$, $u \in \Sigma^*$, and $\sigma \in \Sigma$, one has $\delta(q, \epsilon) = q$, and $\delta(q, u\sigma) = \delta(\delta(q, u), \sigma)$.

A transition $q_1 \xrightarrow{a} q_2$ is interpreted as follows: when a DFA A is in state q_1 and reads letter a , A transitions to state q_2 . A *run* is a sequence $q_1 \xrightarrow{a_1} \dots \xrightarrow{a_n} q_n$ of transitions (any q_i could be equal to the initial state q_0), where $n \in \mathbb{Z}_+$ and $q_{i+1} = \delta(q_i, a_{i+1})$, $1 \leq i \leq n-1$. Particularly $n = 1$ means the run degenerates to a single state q_1 .

Let Σ be an alphabet and $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. A word $w \in \Sigma^*$ is called *accepted by DFA A* if $\delta(q_0, w) \in F$. The formal language *recognized by DFA A* is the set of all words accepted by A , i.e., $\{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$. A formal language recognized by some DFA is called *regular*.

Example 2.1. Consider the DFA A_1 as in Figure 2.1.

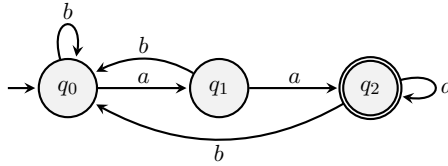


Figure 2.1: DFA A_1 , where the state with an input arrow from nowhere (i.e., q_0) is initial, states are drawn in circles, particularly the states in double circles are final, the symbols beside arrows are letters.

In A_1 , all transitions ending with the unique final state q_2 are $q_1 \xrightarrow{a} q_2$ and $q_2 \xrightarrow{a} q_2$, and the unique transition ending with q_1 is $q_0 \xrightarrow{a} q_1$. Hence all transition sequences of length 2 ending with q_2 are $q_0 \xrightarrow{a} q_1 \xrightarrow{a} q_2$,

$q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_2$, and $q_2 \xrightarrow{a} q_2 \xrightarrow{a} q_2$. Hence all words accepted by A_1 end with aa . On the contrary, for every word waa , where $w \in \Sigma^*$, no matter $\delta(q_0, w)$ is equal to q_0 , q_1 , or q_2 , A_1 accepts waa . Hence A_1 exactly accepts all words ending with aa . $\square$

Definition 2.2. A *nondeterministic finite automaton* (NFA) is also a quintuple

$$(Q, \Sigma, \delta, q_0, F),$$

where

1. Q is a finite set of *states*,
2. Σ is an *alphabet*,
3. $\delta : Q \times \Sigma \rightarrow 2^Q$ is the *transition function*,
4. $q_0 \in Q$ is the *initial state*, and
5. $F \subset Q$ is the set of *final states*.

There exists a transition $q \xrightarrow{a} q'$ if and only if $q' \in \delta(q, a)$. When an NFA A is in a state q and reads a letter a , then A can transition to any state of $\delta(q, a)$. We extend the transition function $\delta : Q \times \Sigma \rightarrow 2^Q$ to $\delta : Q \times \Sigma^* \rightarrow 2^Q$ recursively as follows: for all $q \in Q$, $u \in \Sigma^*$, and $a \in \Sigma$, one has $\delta(q, \epsilon) = \{q\}$, and $\delta(q, ua) = \bigcup_{p \in \delta(q, u)} \delta(p, a)$.

Definition 2.3. Let Σ be an alphabet and $A = (Q, \Sigma, \delta, q_0, F)$ be an NFA. A word $w \in \Sigma^*$ is called *accepted by A* if $\delta(q_0, w) \cap F \neq \emptyset$. The formal language *recognized by A* is the set of all words accepted by A , i.e., $\{w \in \Sigma^* | \delta(q_0, w) \cap F \neq \emptyset\}$.

In other words, for an NFA $A = (Q, \Sigma, \delta, q_0, F)$ over an alphabet Σ and a word $w = w_1 \dots w_n \in \Sigma^*$, where $w_1, \dots, w_n \in \Sigma$, A accepts w if and only if there exist states $q_1, \dots, q_n$ such that $q_i \in \delta(q_{i-1}, w_i)$ for all $1 \leq i \leq n$ and $q_n \in F$.

When spontaneous transitions are added into an NFA, i.e., the transitions between states without reading any letter (represented by reading ϵ), an NFA is extended to an ϵ -*nondeterministic finite automaton* (ϵ -NFA).

Definition 2.4. An ε -nondeterministic finite automaton (ε -NFA) is a quintuple

$$(Q, \Sigma_\varepsilon, \delta, q_0, F),$$

where

1. Q is a finite set of *states*,
2. Σ is an *alphabet*, $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, $\varepsilon \notin \Sigma$,
3. $\delta : Q \times \Sigma_\varepsilon \rightarrow 2^Q$ is the *transition function*,
4. $q_0 \in Q$ is the *initial state*, and
5. $F \subset Q$ is the set of *final states*.

The notions of transitions, runs, etc., of DFAs, are naturally extended to NFAs and further to ε -NFAs, where particularly, the transitions of the form $q_1 \xrightarrow{\varepsilon} q_2$ are called ε -transitions, and the runs of the form $q_1 \xrightarrow{\varepsilon} \cdots \xrightarrow{\varepsilon} q_n$ with $n \geq 1$ are called ε -runs, where a single state q_1 is also called an ε -run. The words accepted by an ε -NFA A are those that lead A to at least one final state. That is, ϵ is accepted by A if and only if $q_0 \in F$ or there is an ε -run from q_0 some final state other than q_0 ; a word $a_1 \dots a_n \in \Sigma^+$ is accepted by A if and only if there is a run $q_0 \xrightarrow{\varepsilon^*} q'_0 \xrightarrow{a_1} q_1 \xrightarrow{\varepsilon^*} q'_1 \cdots \xrightarrow{\varepsilon^*} q'_{n-1} \xrightarrow{a_n} q_n \xrightarrow{\varepsilon^*} q'_n$ with $q'_n \in F$, where $q_i \xrightarrow{\varepsilon^*} q'_i$ denotes an ε -run from q_i to q'_i , $0 \leq i \leq n$. The language recognized by an ε -NFA A is the set of words accepted by A .

By definition, the languages recognized by DFAs are a subset of those recognized by NFAs, and the languages recognized by NFAs are a subset of those recognized by ε -NFAs. A well-known result is that the language classes recognized by the three kinds of finite automata coincide, because the *powerset construction* (Rabin and Scott, 1959; Sipser, 1996) (aka *subset construction*) provides a procedure to convert an ε -NFA to its equivalent DFA¹.

Consider an ε -NFA $A_N^\varepsilon = (Q, \Sigma_\varepsilon, \delta, q_0, F)$ that recognizes a given language L . We next construct a DFA A_D from A_N^ε that also recognizes L . The idea of constructing A_D is keeping track of every word,

¹Two automata are equivalent if they recognize the same language.

meanwhile, collecting all states the word can lead to, by additionally considering all possible ε -transitions. The state set of A_D is the power set of the state set of A_N^ε , so this construction is usually called powerset construction.

Formally,

$$A_D = (Q', \Sigma, \delta', q'_0, F'), \quad (2.1)$$

where

1. $Q' = 2^Q$,
2. for all $x \in Q'$ and $a \in \Sigma$, $\delta'(x, a) = \varepsilon\text{-CLO} \left(\bigcup_{q \in x} \delta(q, a) \right)$,
3. $q'_0 = \varepsilon\text{-CLO}(\{q_0\})$,
4. $F' = \{x \in Q' \mid x \cap F \neq \emptyset\}$,

where for a subset $Q' \subset Q$ of states, its ε -closure $\varepsilon\text{-CLO}(Q')$ is defined by the set of states to which there are ε -runs from the states of Q' , $\varepsilon\text{-CLO}(Q') = \{q \in Q \mid (\exists q' \in Q')[q' \xrightarrow{\varepsilon^*} q]\}$. By the construction, one directly sees for every word $w \in \Sigma^*$, w is accepted by A_N^ε if and only if it is accepted by A_D . Hence A_N^ε and A_D recognize the same language.

Example 2.2. Consider the ε -NFA A_2 in Figure 2.2. We construct its equivalent DFA A_{2D} as in Figure 2.3 by the powerset construction. $\square$

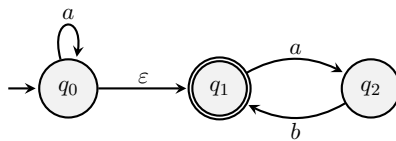


Figure 2.2: ε -NFA A_2 .

One basic tool — observer (Definition 2.7) widely used in DESs, is exactly obtained from the powerset construction.

Definition 2.5. A *labeled finite-state automaton* (LFSA) is a sextuple

$$\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell), \quad (2.2)$$

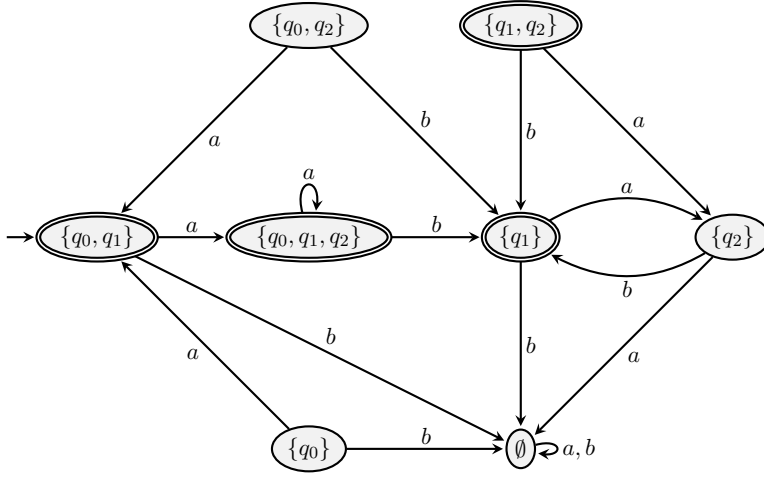


Figure 2.3: An equivalent DFA A_{2D} of the ε -NFA A_2 in Figure 2.2.

where

1. Q is a finite set of *states*,
2. E is an alphabet of *events*,
3. $\delta : Q \times E \rightarrow 2^Q$ is the *transition function* (equivalently represented by the *transition relation* $\delta \subset Q \times E \times Q$ such that $q' \in \delta(q, e)$ if and only if $(q, e, q') \in \delta$),
4. $Q_0 \subset Q$ is a set of *initial states*,
5. Σ is an alphabet of *outputs/labels*, and
6. $\ell : E \rightarrow \Sigma \cup \{\epsilon\}$ is the *labeling function*.

The transition function δ is *deterministic* if for all $q \in Q$ and $e \in E$, $|\delta(q, e)| \leq 1$. A transition $(q, e, q') \in \delta$ is interpreted as when $\mathcal{S}$ is in state q and event e occurs² $\mathcal{S}$ transitions to state q' . The event set E can be rewritten as disjoint union of *observable* event set $E_o = \{e \in E | \ell(e) \in \Sigma\}$ and *unobservable* event set $E_{uo} = \{e \in E | \ell(e) = \epsilon\}$. When an observable event occurs, its label can be observed; when an unobservable event

²Different from the interpretation of reading letter e in an NFA.

occurs, nothing can be observed. Transition function $\delta : Q \times E \rightarrow 2^Q$ is recursively extended to $\delta : Q \times E^* \rightarrow 2^Q$ as follows: for all $q \in Q$, $u \in E^*$, and $a \in \Sigma$, one has $\delta(q, \epsilon) = \{q\}$ and $\delta(q, ua) = \bigcup_{p \in \delta(q, u)} \delta(p, a)$; equivalently, transition relation $\delta \subset Q \times E \times Q$ is recursively extended to $\delta \subset Q \times E^* \times Q$ as follows: (1) for all $q, q' \in Q$, $(q, \epsilon, q') \in \delta$ if and only if $q = q'$; (2) for all $q, q' \in Q$, $s \in E^*$, and $e \in E$, one has $(q, se, q') \in \delta$, also denoted by $q \xrightarrow{se} q'$, called *transition sequence* or *run*, if and only if, $(q, s, q''), (q'', e, q') \in \delta$ for some $q'' \in Q$.

Labeling function $\ell : E \rightarrow \Sigma \cup \{\epsilon\}$ is recursively extended to $\ell : E^* \cup E^\omega \rightarrow \Sigma^* \cup \Sigma^\omega$ as $\ell(e_1 e_2 \dots) = \ell(e_1) \ell(e_2) \dots$ and $\ell(\epsilon) = \epsilon$. For all $E' \subset E$, $\ell(E') := \{\ell(e) | e \in E'\}$. Transitions $x \xrightarrow{e} x'$ with $\ell(e) = \epsilon$ (resp., $\ell(e) \neq \epsilon$) are called *unobservable* (resp., *observable*). For $q \in Q$ and $s \in E^+$, (q, s, q) is called a *(transition) cycle* if $(q, s, q) \in \delta$. An *observable (transition) cycle* is defined by a cycle with at least one observable transition. Analogously an *unobservable cycle* is defined by a cycle with no observable transition. An LFSA is called *deterministic* if $|Q_0| = 1$ and for all $q, q', q'' \in Q$ and $e \in E$, if $(q, e, q'), (q, e, q'') \in \delta$ then $q' = q''$.

A state $q' \in Q$ is *reachable* from a state $q \in Q$ if there exists $s \in E^+$ such that $q \xrightarrow{s} q'$. A subset Q' of Q is *reachable* from a state $q \in Q$ if some state of Q' is reachable from q . Similarly a state $q \in Q$ is *reachable* from a subset Q' of Q if q is reachable from some state of Q' . A state $q \in Q$ is called *reachable (in $\mathcal{S}$)* if either $q \in Q_0$ or it is reachable from some initial state. For a transition $(q, e, q') \in \delta$, a transition $(q'', e', q''') \in \delta$ is called a *predecessor* of (q, e, q') if either $q = q'''$ or q is reachable from q''' ; a transition (q'', e', q''') is called a *successor* of (q, e, q') if either $q'' = q'$ or q'' is reachable from q' .

A state $q \in Q$ is called *live* if $(q, e, q') \in \delta$ for some $e \in E$ and $q' \in Q$. $\mathcal{S}$ is called *live* or *deadlock-free* if each of its reachable states is live. $\mathcal{S}$ is called *divergence-free* or *convergent* if it contains no reachable, unobservable cycle. The two assumptions imply that at each reachable state, there exists an infinitely long event sequence whose label sequence is also of infinite length. They have been widely used in the past over two decades, since they guarantee the widely-used tools such as detector (Shu and Lin, 2011b, for verifying strong detectability), twin plant (Jiang et al., 2001, for verifying diagnosability), verifier (Yoo and Lafortune, 2002, for verifying diagnosability), and verifier (Genc and Lafortune,

2009, for verifying predictability) work. Without the two assumptions, these tools do not work. *In the monograph, our developed concurrent composition does not depend on any assumption and will take the place of all the above tools.*

The symbol

$$L(\mathcal{S}) := \{s \in E^* | (\exists q_0 \in Q_0)(\exists q \in Q)[q_0 \xrightarrow{s} q]\} \quad (2.3)$$

will denote the set of finite-length event sequences generated by $\mathcal{S}$,

$$L^\omega(\mathcal{S}) := \{e_1 e_2 \dots \in E^\omega | (\exists q_0 \in Q_0)(\exists q_1, q_2, \dots \in Q)[q_0 \xrightarrow{e_1} q_1 \xrightarrow{e_2} \dots]\} \quad (2.4)$$

will denote the set of infinite-length event sequences generated by $\mathcal{S}$. For each $\sigma \in \Sigma^*$, $\mathcal{M}(\mathcal{S}, \sigma)$ denotes the *current-state estimate*, i.e., the set of states that the system can be in when σ has just been generated, i.e.,

$$\mathcal{M}(\mathcal{S}, \sigma) := \{q \in Q | (\exists q_0 \in Q_0)(\exists s \in E^*)[(\ell(s) = \sigma) \wedge (q_0 \xrightarrow{s} q)]\}. \quad (2.5)$$

$\mathcal{L}(\mathcal{S})$ will denote the *language generated by $\mathcal{S}$* , i.e.,

$$\mathcal{L}(\mathcal{S}) := \{\sigma \in \Sigma^* | \mathcal{M}(\mathcal{S}, \sigma) \neq \emptyset\}. \quad (2.6)$$

$\mathcal{L}^\omega(\mathcal{S})$ denotes the *ω -language generated by $\mathcal{S}$* , i.e.,

$$\mathcal{L}^\omega(\mathcal{S}) := \{\sigma \in \Sigma^\omega | (\exists s \in L^\omega(\mathcal{S}))[\ell(s) = \sigma]\}. \quad (2.7)$$

For a subset $x \subset Q$ of states, its *unobservable reach* $\text{UR}(x)$ is defined by $\bigcup_{q \in x} \bigcup_{s \in (E_{uo})^*} \delta(q, s)$ (which is similar to the ε -CLO in ε -NFAs, see paragraph below (2.1), Page 19).

Example 2.3. Consider the following LFSA $\mathcal{S}_1$ shown in Figure 2.4. It is deterministic but not live (q_1 is not live). One sees $L(\mathcal{S}_1) = \{(e_1)^n, (e_1)^n e_2 | n \geq 0\}$, $L^\omega(\mathcal{S}_1) = \{(e_1)^\omega\}$, $\mathcal{L}(\mathcal{S}_1) = \{a^n, a^n b | n \geq 0\}$, $\mathcal{L}^\omega(\mathcal{S}_1) = \{a^\omega\}$, and $\mathcal{M}(\mathcal{S}_1, b) = \{q_1\}$.

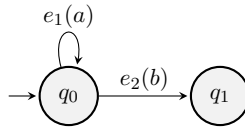


Figure 2.4: LFSA $\mathcal{S}_1$, where q_0 is the initial state (with an input arrow from nowhere), $\ell(e_1) = a$, $\ell(e_2) = b$.

□

2.2 Basic Tools — Concurrent Composition and Observer

In this section, we introduce two basic tools — concurrent composition and observer.

The concurrent-composition operator arose from characterizing the *negation* of a strong version of detectability called *eventual strong detectability* in Zhang and Giua (2020b), where the eventual strong detectability is essentially different from and strictly weaker than the fundamental notion of *strong detectability* formulated by Shu *et al.* (2007). Soon later, the concurrent composition was used to verify strong detectability of LFSAs without any assumption for the first time (Zhang and Giua, 2019). In the concurrent composition of two automata, observable transitions with the same label are synchronized, while unobservable transitions interleave.

Definition 2.6 (Zhang and Giua, 2019; Zhang and Giua, 2020b). Consider two LFSAs $\mathcal{S}^i = (Q_i, E, \delta_i, Q_{0i}, \Sigma, \ell)$, $i = 1, 2$, we define the *concurrent composition* $\text{CC}_A(\mathcal{S}^1, \mathcal{S}^2)$ of $\mathcal{S}^1$ and $\mathcal{S}^2$ by the LFSA

$$\text{CC}_A(\mathcal{S}^1, \mathcal{S}^2) = (Q', E', \delta', Q'_0, \Sigma, \ell'), \quad (2.8)$$

where

1. $Q' = Q_1 \times Q_2$,
2. $E' = E'_o \cup E'_{uo}$, where $E'_o = \{(\check{e}, \check{e}') | \check{e}, \check{e}' \in E_o, \ell(\check{e}) = \ell(\check{e}')\}$,
 $E'_{uo} = \{(\check{e}, \epsilon) | \check{e} \in E_{uo}\} \cup \{(\epsilon, \check{e}) | \check{e} \in E_{uo}\}$,
3. for all $(\check{q}_1, \check{q}'_1), (\check{q}_2, \check{q}'_2) \in Q'$, $(\check{e}, \check{e}') \in E'_o$, $(\check{e}'', \epsilon) \in E'_{uo}$, and $(\epsilon, \check{e}''') \in E'_{uo}$,
 - (a) $((\check{q}_1, \check{q}'_1), (\check{e}, \check{e}'), (\check{q}_2, \check{q}'_2)) \in \delta'$ if and only if $(\check{q}_1, \check{e}, \check{q}_2) \in \delta_1$,
 $(\check{q}'_1, \check{e}', \check{q}'_2) \in \delta_2$,
 - (b) $((\check{q}_1, \check{q}'_1), (\check{e}'', \epsilon), (\check{q}_2, \check{q}'_2)) \in \delta'$ if and only if $(\check{q}_1, \check{e}'', \check{q}_2) \in \delta_1$,
 $\check{q}'_1 = \check{q}'_2$,
 - (c) $((\check{q}_1, \check{q}'_1), (\epsilon, \check{e}'''), (\check{q}_2, \check{q}'_2)) \in \delta'$ if and only if $\check{q}_1 = \check{q}_2$, $(\check{q}'_1, \check{e}''', \check{q}'_2) \in \delta_2$,

4. $Q'_0 = Q_{01} \times Q_{02}$,
5. for all $(\check{e}, \check{e}') \in E'$, $\ell'((\check{e}, \check{e}')) := \ell(\check{e}) = \ell(\check{e}')$.

For (3a), $((\check{q}_1, \check{q}'_1), (\check{e}, \check{e}'), (\check{q}_2, \check{q}'_2)) \in \delta'$ is an observable transition of $\text{CC}_A(\mathcal{S}^1, \mathcal{S}^2)$ if and only if it left (resp., right) component $(\check{q}_1, \check{e}, \check{q}_2)$ (resp., $(\check{q}'_1, \check{e}', \check{q}'_2)$) is an observable transition of $\mathcal{S}^1$ (resp., $\mathcal{S}^2$). For (3b), $((\check{q}_1, \check{q}'_1), (\check{e}'', \epsilon), (\check{q}_2, \check{q}'_2)) \in \delta'$ is an unobservable transition of $\text{CC}_A(\mathcal{S}^1, \mathcal{S}^2)$ if and only if its left component $(\check{q}_1, \check{e}'', \check{q}_2)$ is an unobservable transition of $\mathcal{S}^1$, but in its right component, there is no state transition. (3c) is the symmetric form of (3b). Particularly if $\mathcal{S}^1 = \mathcal{S}^2$, then $\text{CC}_A(\mathcal{S}^1, \mathcal{S}^2) =: \text{CC}_A(\mathcal{S}^1)$ is called the *self-composition* of $\mathcal{S}^1$.

For an event sequence $s' \in (E')^*$, we use $s'(L)$ and $s'(R)$ to denote its left and right components, respectively. Similar notation is applied to states of Q' . In addition, for every $s' \in (E')^*$, we use $\ell(s')$ to denote $\ell(s'(L))$ or $\ell(s'(R))$, since $\ell(s'(L)) = \ell(s'(R))$. In the above construction, $\text{CC}_A(\mathcal{S}^1, \mathcal{S}^2)$ aggregates all pairs of runs of $\mathcal{S}_1$ and runs of $\mathcal{S}_2$ that produce the same label sequence. $\text{CC}_A(\mathcal{S}^1, \mathcal{S}^2)$ has at most $|Q_1||Q_2|$ states and at most $|Q_1||Q_2|(|E_{uo1}||Q_1| + |E_{uo2}||Q_2| + \sum_{\sigma \in \Sigma} |\ell^{-1}(\sigma)|^2|Q_1||Q_2|)$ transitions.

Example 2.4. An LFSA $\mathcal{S}_2$ and its self-composition $\text{CC}_A(\mathcal{S}_2)$ are shown in Figure 2.5.

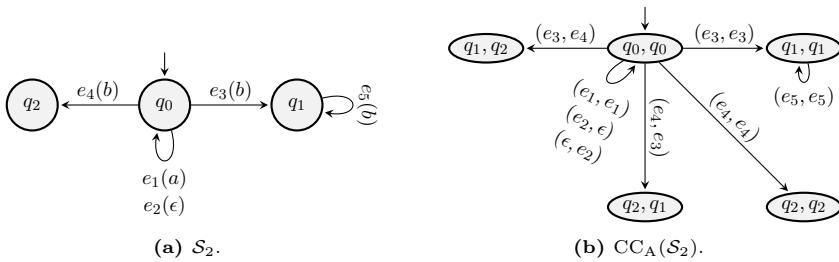


Figure 2.5: LFSA $\mathcal{S}_2$ (left) and its self-composition (right, only reachable states illustrated).

□

Remark 2.1. In Lee and Seshia (2017), the term of concurrent composition has been discussed and it means a set of all kinds of synchronous

and asynchronous compositions of fully-observed state machines. However, the concurrent composition as in Definition 2.6 is a single operator, which did not appear in Lee and Seshia (2017). As mentioned above, our concurrent composition works on partially-observed systems: observable transitions with the same label are synchronized, while unobservable transitions interleave (which is a type of asynchronous compositions). Hence our concurrent composition contains both synchronous compositions and asynchronous compositions. On the other hand, recently another definition of concurrent composition for differential privacy was proposed and studied (Vadhan and Zhang, 2022; Vadhan and Wang, 2021), but this concurrent composition is in continuous mathematics and hence remarkably different from ours. $\square$

For an LFSA $\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell)$, its observer is actually its equivalent DFA. After replacing each unobservable event of $\mathcal{S}$ by ε , we obtain an ε -NFA $\mathcal{S}^\varepsilon$. The observer of $\mathcal{S}$ is the equivalent DFA of $\mathcal{S}^\varepsilon$ defined by (2.1).

Definition 2.7 (Rabin and Scott, 1959; Hopcroft and Ullman, 1969; Sipser, 1996). Consider an LFSA $\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell)$. Its *observer* $\mathcal{S}_{\text{obs}}$ ³ is defined by the DFA

$$(Q_{\text{obs}}, \ell(E_o), \delta_{\text{obs}}, q_{0\text{ obs}}),$$

where

1. $Q_{\text{obs}} = 2^Q$,
2. $\ell(E_o) = \ell(E) \setminus \{\varepsilon\}$,
3. for all $x \in Q_{\text{obs}}$ and $a \in \ell(E_o)$, $\delta_{\text{obs}}(x, a) = \bigcup_{q \in x} \bigcup_{\substack{e_a \in E_o \\ \ell(e_a) = a \\ s \in (E_{uo})^*}} \delta(q, e_a s)$,
4. $q_{0\text{ obs}} = \bigcup_{q_0 \in Q_0} \bigcup_{s \in (E_{uo})^*} \delta(q_0, s) = \text{UR}(Q_0)$.

By definition, for all $a \in \ell(E_o)$, one has $\delta_{\text{obs}}(\emptyset, a) = \emptyset$. The size of $\mathcal{S}_{\text{obs}}$ is $O(2^{|Q|}|\ell(E_o)|)$, it costs time $O(2^{|Q|}|\ell(E_o)||Q|^2|E|)$ to compute

³The term “observer” dates back to Özveren and Willsky (1990) and Shu *et al.* (2007).

$\mathcal{S}_{\text{obs}}$: for every subset $x \subset Q$ and every label $a \in \ell(E_o)$, the time consumption of computing $\delta(x, a)$ is bounded from above by $|Q|^2|E_o| + |Q|^2|E_{uo}| = |Q|^2|E|$. When the labeling function ℓ is injective on E_o , the time consumption of computing $\delta(x, a)$ degenerates to $|Q|^2 + |Q|^2|E_{uo}| = |Q|^2(1 + |E_{uo}|)$, and the time consumption of computing $\mathcal{S}_{\text{obs}}$ degenerates to $O(2^{|Q|}|E_o||Q|^2(1 + |E_{uo}|))$. When the labeling function ℓ is injective on E_o and $\mathcal{S}$ is deterministic, the time consumption of computing $\delta(x, a)$ degenerates to $|Q| + |Q||E_{uo}| = |Q|(1 + |E_{uo}|)$, and the time consumption of computing $\mathcal{S}_{\text{obs}}$ degenerates to $O(2^{|Q|}|E_o||Q|(1 + |E_{uo}|))$.

Example 2.5. Part of the observer $\mathcal{S}_{2\text{obs}}$ of the LFSA $\mathcal{S}_2$ in Figure 2.5 is shown in Figure 2.6.

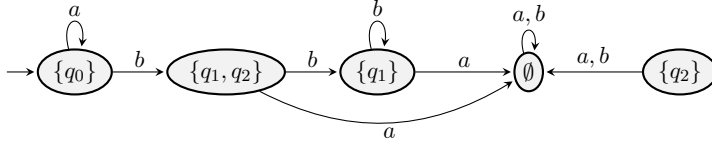


Figure 2.6: Part of observer $\mathcal{S}_{2\text{obs}}$ of the LFSA $\mathcal{S}_2$ in Figure 2.5.

□

The detector is a simplified version of the observer, and was proposed in Caines *et al.* (1988), Caines *et al.* (1991), and Shu and Lin (2011b). A detector is obtained from an observer by splitting its states into subsets of cardinality at most 2. Hence the size of a detector of an LFSA is polynomial in the size of the LFSA. By Shu and Lin (2011b), the detector was used to verify strong detectability in polynomial time under the two assumptions of liveness and divergence-freeness.

Definition 2.8. Consider an LFSA $\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell)$. Its *detector* $\mathcal{S}_{\text{det}}$ is defined by the NFA

$$(Q_{\text{det}}, \ell(E_o), \delta_{\text{det}}, q_{0\text{det}}),$$

where

1. $Q_{\text{det}} = \{q_{0\text{det}}\} \cup \{Q' \subset Q \mid 1 \leq |Q'| \leq 2\}$,
2. $\ell(E_o) = \ell(E) \setminus \{\epsilon\}$,

3. for all $x \in Q_{\text{det}}$ and $a \in \ell(E_o)$, $(x, a, y) \in \delta_{\text{det}}$ if either

$$\begin{aligned} \text{(a)} \quad & \left| \bigcup_{q \in x} \bigcup_{\substack{e_a \in E_o \\ \ell(e_a)=a \\ s \in (E_{uo})^*}} \delta(q, e_a s) \right| = 1, \quad y = \bigcup_{q \in x} \bigcup_{\substack{e_a \in E_o \\ \ell(e_a)=a \\ s \in (E_{uo})^*}} \delta(q, e_a s), \\ \text{or,} \\ \text{(b)} \quad & \left| \bigcup_{q \in x} \bigcup_{\substack{e_a \in E_o \\ \ell(e_a)=a \\ s \in (E_{uo})^*}} \delta(q, e_a s) \right| > 1, \quad y \subset \bigcup_{q \in x} \bigcup_{\substack{e_a \in E_o \\ \ell(e_a)=a \\ s \in (E_{uo})^*}} \delta(q, e_a s), \\ & \text{and } |y| = 2, \end{aligned}$$

4. $q_{0 \text{ det}} = \bigcup_{q_0 \in Q_0} \bigcup_{s \in (E_{uo})^*} \delta(q_0, s) = \text{UR}(Q_0)$.

For (3a), $x \xrightarrow{a} y$ is a transition of the observer $\mathcal{S}_{\text{obs}}$ and $|y| = 1$. For (3b), $x \xrightarrow{a} y'$ is a transition of the observer $\mathcal{S}_{\text{obs}}$ and $|y'| \geq 2$, $y' = \bigcup_{q \in x} \bigcup_{\substack{e_a \in E_o \\ \ell(e_a)=a \\ s \in (E_{uo})^*}} \delta(q, e_a s)$, y is any subset of y' of cardinality 2.

2.3 Verification of Inference-Based Properties

2.3.1 Detectability

In this subsection, we show how to use the concurrent-composition method to verify strong detectability (first studied by Shu *et al.*, 2007) and eventual strong detectability (first studied by Zhang and Giua, 2020b).

The study of the state detection problem dates back to Moore (1956) in computer science and Kalman (1963) in control science, respectively. In the former, Moore studied initial-state detection (called Gedanken-experiment) of finite-state machines which were called Moore machines later; in the latter, Kalman studied initial-state detection (called observability) of linear differential equations. The two seminal papers stimulated many research branches in computer science and control, e.g., model-based testing of all kinds of reactive systems (Gill, 1962; Broy *et al.*, 2005) in computer science and observability studies of all kinds of control systems, e.g., arranging from linear systems (Kalman, 1963; Wonham, 1985), to nonlinear systems (Sontag, 1979; Conte *et al.*,

2007; Isidori, 1995), to switched systems (Tanwani *et al.*, 2013), and also to networked systems (Kibangou *et al.*, 2016; Angulo *et al.*, 2019).

The state detection problem in DESs dates back to Ramadge (1986) and Özveren and Willsky (1990). Two widely accepted fundamental notions are *strong detectability* and *weak detectability* proposed by Shu *et al.* (2007), where the former implies that there is a delay k such that for *each* event sequence generated by an LFSA, each prefix of its output sequence of length greater than k allows reconstructing the current state. The latter relaxes the former by changing the verbatim from *each* to *some*. *Eventual strong detectability* was proposed by Zhang and Giua (2020b), which is a strong version of detectability and essentially different from strong detectability. Eventual strong detectability implies for each generated event sequence s , there is a delay k_s depending on s such that each prefix of its output sequence of length greater than k_s allows reconstructing the current state. Because the delay k_s depends on an event sequence, it is more difficult to derive a verification algorithm for eventual strong detectability than strong detectability. It even requires three tools — concurrent composition, observation automaton, and bifurcation automaton to verify eventual strong detectability of LFSAs (Zhang and Giua, 2020b). In this subsection, we simplify the verification of eventual strong detectability by only using concurrent composition. Eventual strong detectability is strictly weaker than strong detectability and strictly stronger than weak detectability. The bifurcation automaton was also used to give a linear-time verification algorithm for *instant strong detectability* of LFSAs (Zhang and Giua, 2020a) which implies that every output sequence allows reconstructing the current state. The instant strong detectability is the strongest known detectability property.

When long-term behavior is considered, we let the above conditions apply to all infinite-length generated event sequences (in this case we call the notions ω -detectability or ω -versions of detectability); when short-term behavior is considered, we let them apply to all finite-length generated event sequences (in this case we call the notions $*$ -detectability or $*$ -versions of detectability).

The following definitions of strong detectability (Definition 2.9) and weak detectability (Definition 2.10) are chosen from Shu *et al.* (2007) defined on live and divergence-free LFSAs. In our monograph, because

of the adoption of the concurrent composition, we can consider general LFSAAs.

Definition 2.9 (ω -SD). An LFSA $\mathcal{S}$ is called ω -strongly detectable if there exists a positive integer k such that for each infinite-length event sequence $s \in L^\omega(\mathcal{S})$, $|\mathcal{M}(\mathcal{S}, \sigma)| = 1$ for every prefix σ of $\ell(s)$ satisfying $|\sigma| \geq k$.

Definition 2.10 (ω -WD). An LFSA $\mathcal{S}$ is called ω -weakly detectable if either $L^\omega(\mathcal{S}) = \emptyset$ or there exists an infinite-length event sequence $s \in L^\omega(\mathcal{S})$ such that for some positive integer k , $|\mathcal{M}(\mathcal{S}, \sigma)| = 1$ for every prefix σ of $\ell(s)$ satisfying $|\sigma| \geq k$.

Definition 2.11 ($*$ -SD). An LFSA $\mathcal{S}$ is called $*$ -strongly detectable if there exists a positive integer k such that for each finite-length event sequence $s \in L(\mathcal{S})$, $|\mathcal{M}(\mathcal{S}, \sigma)| = 1$ for every prefix σ of $\ell(s)$ satisfying $|\sigma| \geq k$.

Definition 2.12 ($*$ -WD). An LFSA $\mathcal{S}$ is called $*$ -weakly detectable if there exists a finite-length event sequence $s \in L(\mathcal{S})$ such that for some positive integer k , $|\mathcal{M}(\mathcal{S}, \sigma)| = 1$ for every prefix σ of $\ell(s)$ satisfying $|\sigma| \geq k$.

The following definition of eventual strong detectability (Definition 2.13) is chosen from Zhang and Giua (2020b).

Definition 2.13 (ω -ESD). An LFSA $\mathcal{S}$ is called ω -eventually strongly detectable if for each infinite-length event sequence $s \in L^\omega(\mathcal{S})$, there exists a positive integer k_s (depending on s) such that $|\mathcal{M}(\mathcal{S}, \sigma)| = 1$ for every prefix σ of $\ell(s)$ satisfying $|\sigma| \geq k_s$.

Definition 2.14 ($*$ -ESD). An LFSA $\mathcal{S}$ is called $*$ -eventually strongly detectable if for each finite-length event sequence $s \in L(\mathcal{S})$, there exists a positive integer k_s (depending on s) such that $|\mathcal{M}(\mathcal{S}, \sigma)| = 1$ for every prefix σ of $\ell(s)$ satisfying $|\sigma| \geq k_s$.

The above $*$ -version of eventual strong detectability is not interesting, because every LFSA satisfies this property: for every $s \in L(\mathcal{S})$, one chooses $k_s = |\ell(s)| + 1$.

An exponential-time verification algorithm based on the notion of observer for weak detectability was given by Shu *et al.* (2007) based on two assumptions of liveness and divergence-freeness (see paragraph before (2.20), Page 22). Later, verifying weak detectability was proven to be PSPACE-complete (Zhang, 2017), where PSPACE-hardness was proven by reducing the PSPACE-complete DFA INTERSECTION problem (proven by Kozen, 1977) to weak detectability of LFSAs having more than one initial state and whose transition functions are deterministic in polynomial time. More recently, Masopust (2018) refined the reduction in Zhang (2017) and proved that verifying even more restrictive LFSAs is still PSPACE-hard. In Shu and Lin (2011b), a detector method was used to verify strong detectability in polynomial time, also under the two assumptions as mentioned above, where the detector is a simplified version of the observer by splitting the states of the observer into subsets of cardinality 2 (previously used by Caines *et al.*, 1988; Caines *et al.*, 1991). We refer the reader to Zhang and Giua (2020b, Remark 2) for why the detector method depends on the two assumptions and without the two assumptions the method does not work generally. See Table 2.1 for a summary.

Table 2.1: Results on complexity of verifying various notions of detectability of LFSAs, where ω -SD, ω -ESD, and ω -WD are short for ω -strong detectability, ω -eventual strong detectability, and ω -weak detectability, respectively; $\mathcal{S}$ denotes an LFSA, “d.d.” is short for “deadlock-free (i.e., live), divergence-free”, “c.” is short for “complete”.

	ω -SD	ω -ESD	ω -WD
$\mathcal{S}$	P (Zhang and Giua, 2019) NL (Zhang, 2021a)	P (Zhang and Giua, 2020b)	PSPACE (Zhang and Giua, 2020b)
d.d. $\mathcal{S}$	P (Shu and Lin, 2011b) NL-c. (Masopust, 2018)		EXPTIME (Shu <i>et al.</i> , 2007) PSPACE-c. (Zhang, 2017)

Verifying strong detectability

In order to verify strong detectability, we choose to characterize its *negation*, which is essentially different from the way of directly verifying strong detectability adopted by Shu *et al.* (2007) and Shu and Lin (2011b). By definition, the following proposition holds.

Proposition 2.1 (Zhang *et al.*, 2020). An LFSA $\mathcal{S}$ is not ω -strongly detectable (resp., not $*$ -strongly detectable) if and only if for every positive integer k there exists an infinite-length (resp., finite-length) event sequence $s \in L^\omega(\mathcal{S})$ (resp., $s \in L(\mathcal{S})$) such that $|\mathcal{M}(\mathcal{S}, \sigma)| > 1$ for some prefix σ of $\ell(s)$ satisfying $|\sigma| \geq k$.

With the notion of self-composition of an LFSA $\mathcal{S}$, we give necessary and sufficient conditions for the negations of two versions of strong detectability, without any assumption.

Theorem 2.1 (Zhang and Giua, 2019; Zhang *et al.*, 2020). An LFSA $\mathcal{S}$ is not $*$ -strongly detectable if and only if in $\text{CC}_A(\mathcal{S})$, there exists a run

$$q'_0 \xrightarrow{s'_1} q'_1 \xrightarrow{s'_2} q'_1 \xrightarrow{s'_3} q'_2 \quad (2.9)$$

satisfying

$$q'_0 \in Q'_0; q'_1, q'_2 \in Q'; s'_1, s'_2, s'_3 \in (E')^*; \ell(s'_2) \in \Sigma^+; q'_2(L) \neq q'_2(R). \quad (2.10)$$

An LFSA $\mathcal{S}$ is not ω -strongly detectable if and only if in $\text{CC}_A(\mathcal{S})$, there exists a run (2.9) satisfying (2.10) and in $\mathcal{S}$, there exists a transition cycle reachable from $q'_2(L)$.

Proof We use Proposition 2.1 to prove this theorem. We first consider $*$ -strong detectability.

“only if”: Assume $\mathcal{S}$ is not $*$ -strongly detectable. Then by Proposition 2.1, choose $k = |Q|^2 + 1$, there exists $s_k \in L(\mathcal{S})$ and $\sigma \in \Sigma^+$ such that $|\mathcal{M}(\mathcal{S}, \sigma)| > 1$, $\sigma \sqsubset \ell(s)$, and $|\sigma| \geq k$. Then in $\text{CC}_A(\mathcal{S})$, there exists a run $q'_0 \xrightarrow{s'} q'$ such that $q'_0 \in Q'_0$, $\ell(s') = \sigma$ and $q'(L) \neq q'(R)$. Since $|\sigma| > |Q|^2$ and there exist at most $|Q|^2$ distinct states in $\text{CC}_A(\mathcal{S})$, by the pigeonhole principle⁴, the run $q'_0 \xrightarrow{s'} q'$ can be rewritten as $q'_0 \xrightarrow{s'_1} q'_1 \xrightarrow{s'_2} q'_1 \xrightarrow{s'_3} q'$, where $\ell(s'_2) \in \Sigma^+$.

⁴If n items are put into m containers with $n > m$, then at least one container must contain more than one item.

“if”: Assume in $\text{CC}_A(\mathcal{S})$ there exists a run (2.9) satisfying (2.10). We choose event sequence $s := s'_1(L)(s'_2(L))^{k+1}s'_3(L) \in L(\mathcal{S})$, then $|\ell(s)| > k$ and $|\mathcal{M}(\mathcal{S}, \ell(s))| > 1$. By Proposition 2.1, $\mathcal{S}$ is not $*$ -strongly detectable.

We second consider ω -strong detectability. Because a transition cycle reachable from $q'_2(L)$ can be repeated arbitrarily often, resulting in an infinite-length run starting from $q'_2(L)$. Then based on the above argument for $*$ -strong detectability, the necessary and sufficient condition for ω -strong detectability also holds. $\square$

Example 2.6. Reconsider the LFSA $\mathcal{S}_2$ and its self-composition $\text{CC}_A(\mathcal{S}_2)$ both in Figure 2.5. In $\text{CC}_A(\mathcal{S}_2)$, one sees a run

$$(q_0, q_0) \xrightarrow{(e_1, e_1)} (q_0, q_0) \xrightarrow{(e_3, e_4)} (q_1, q_2)$$

such that $\ell((e_1, e_1)) = a$ is of positive length and $q_1 \neq q_2$. Then by Theorem 2.1, $\mathcal{S}_2$ is not $*$ -strongly detectable. In addition, in $\mathcal{S}_2$, there is a self-loop on q_1 , hence also by Theorem 2.1, $\mathcal{S}_2$ is not ω -strongly detectable. $\square$

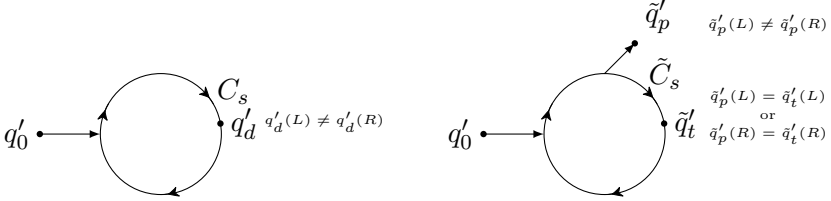
Verifying eventual strong detectability

Verifying eventual strong detectability of LFSAs is more difficult than verifying strong detectability because the time delay k depends on a generated event sequence in the former (see Definition 2.13).

Theorem 2.2 (Zhang and Zhang, 2022). An LFSA $\mathcal{S}$ is not ω -eventually strongly detectable if and only if one of the following two items holds.

- (i) In $\text{CC}_A(\mathcal{S})$ (see Figure 2.7a for an illustration), there exists a reachable, observable cycle C_s that contains a state $q'_d = (q'_d(L), q'_d(R))$ satisfying $q'_d(L) \neq q'_d(R)$.
- (ii) In $\text{CC}_A(\mathcal{S})$ (see Figure 2.7b for an illustration),
 - (a) there exists a reachable, observable cycle $\tilde{C}_s$ such that every state $q' \in \tilde{C}_s$ satisfies $q'(L) = q'(R)$,
 - (b) there exists a state $\tilde{q}'_p$ with $\tilde{q}'_p(L) \neq \tilde{q}'_p(R)$ reachable from cycle $\tilde{C}_s$,

- (c) there exists a state $\tilde{q}'_t \in \tilde{C}_s$ such that either $\tilde{q}'_p(L) = \tilde{q}'_t(L)$ or $\tilde{q}'_p(R) = \tilde{q}'_t(R)$.



(a) An illustration of item (i) in Theorem 2.2. (b) An illustration of item (ii) in Theorem 2.2.

Figure 2.7: An illustration used in the proof of Theorem 2.2.

Proof “if”:

Assume item (i) holds. Then in $\text{CC}_A(\mathcal{S})$, there is an infinite path $\pi'_1 := q'_0 \xrightarrow{s'_0} q'_d \left(\xrightarrow{s'_1} q'_d \right)^\omega$, where q'_0 is initial, $q'_d(L) \neq q'_d(R)$, $|\ell'(s'_1)| > 0$.

In $\mathcal{S}$, there is an infinite path $\pi_{1L} := q'_0(L) \xrightarrow{s'_0(L)} q'_d(L) \left(\xrightarrow{s'_1(L)} q'_d(L) \right)^\omega$ such that $\{q'_d(L), q'_d(R)\} \subset \mathcal{M}(\mathcal{S}, \ell(s'_0(L)(s'_1(L))^k))$ for all $k \geq 0$, hence $|\mathcal{M}(\mathcal{S}, \ell(s'_0(L)(s'_1(L))^k))| > 1$. Thus, $\mathcal{S}$ is not ω -eventually strongly detectable.

Assume item (ii) holds. Then in $\text{CC}_A(\mathcal{S})$, there exists a finite path $\pi' := \tilde{q}'_t \xrightarrow{\tilde{s}'} \tilde{q}'_p$, i.e., $\pi' = (\tilde{q}'_t(L), \tilde{q}'_t(R)) \xrightarrow{\tilde{s}'} (\tilde{q}'_p(L), \tilde{q}'_p(R))$ with $|\ell'(\tilde{s}')| > 0$. Since $\tilde{q}'_t \in \tilde{C}_s$, where $\tilde{q}'_t(L) = \tilde{q}'_t(R)$, and $\tilde{q}'_p(L) = \tilde{q}'_t(L)$ (or $\tilde{q}'_p(R) = \tilde{q}'_t(R)$), we can rewrite π' as $(\tilde{q}'_t(L), \tilde{q}'_t(L)) \xrightarrow{\tilde{s}'} (\tilde{q}'_t(L), \tilde{q}'_p(R))$.

Thus, in $\mathcal{S}$, there exist finite paths $q_0 \xrightarrow{s_0} \tilde{q}'_t(L) \xrightarrow{\tilde{s}'(L)} \tilde{q}'_t(L)$ and $q_0 \xrightarrow{s_0} \tilde{q}'_t(L) \xrightarrow{\tilde{s}'(R)} \tilde{q}'_p(R)$ such that $q_0 \in Q_0$ and $\{\tilde{q}'_p(L), \tilde{q}'_p(R)\} \subset \mathcal{M}(\mathcal{S}, \ell'(s_0 \tilde{s}'(L)))$. Moreover, in $\mathcal{S}$, there exists an infinite path starting at initial state q_0 , reaching state $\tilde{q}'_t(L)$ and visiting cycle $\tilde{q}'_t(L) \xrightarrow{\tilde{s}'(L)} \tilde{q}'_t(L)$ for infinitely many times, e.g., $\pi_2 := q_0 \xrightarrow{s_0} \tilde{q}'_t(L) \left(\xrightarrow{\tilde{s}'(L)} \tilde{q}'_t(L) \right)^\omega$.

For every $k \in \mathbb{Z}_+$, there exists a prefix $\pi_2^k := q_0 \xrightarrow{s_0} \tilde{q}'_t(L) \left(\xrightarrow{\tilde{s}'(L)} \tilde{q}'_t(L) \right)^k$ of path π_2 satisfying $|\ell(s_0(\tilde{s}'(L))^k)| \geq k$ and $|\mathcal{M}(\mathcal{S}, \ell(s_0(\tilde{s}'(L))^k))| = |\{\tilde{q}'_p(L), \tilde{q}'_p(R), \dots\}| > 1$. Therefore, $\mathcal{S}$ is not ω -eventually strongly detectable.

“only if”:

Suppose $\mathcal{S}$ is not ω -eventually strongly detectable and item (i) does not hold.

Then $L^\omega(\mathcal{S}) \neq \emptyset$. Moreover, in $\text{CC}_A(\mathcal{S})$, there exists at least one infinite run starting from initial states, and hence there is a reachable cycle. Since item (i) does not hold, i.e., there does not exist a reachable cycle containing a state q'_d satisfying $q'_d(L) \neq q'_d(R)$, hence in $\text{CC}_A(\mathcal{S})$, every state q' belonging to a reachable cycle satisfies $q'(L) = q'(R)$. Thus, (iia) holds.

Suppose $\mathcal{S}$ has exactly m states. Since $\mathcal{S}$ is not ω -eventually strongly detectable, there is an infinite path $\pi := q_0 \xrightarrow{s_1} q_1 \xrightarrow{s_2} \dots \xrightarrow{s_i} q_i \xrightarrow{s_{i+1}} \dots$, such that for every $i \in \mathbb{Z}_+$, $s_i \in E^*E_o(E_{uo})^*$, $|\mathcal{M}(\mathcal{S}, \ell(s_1s_2\dots s_i))| > 1$. Choose a sufficiently long prefix $\bar{\pi}$ of π as $\bar{\pi} := q_0 \xrightarrow{s_1} q_1 \xrightarrow{s_2} \dots \xrightarrow{s_k} q_k$ with $k \geq m^2 + m$. By the pigeonhole principle, at least one state appears no less than $m + 1$ times in $\{q_1, \dots, q_k\}$. We denote such a state by $\bar{q}$. Then we can rewrite $\bar{\pi}$ as $q_0 \xrightarrow{\bar{s}_1} \bar{q} \xrightarrow{\bar{s}_2} \dots \xrightarrow{\bar{s}_j} \bar{q}$, where $j \geq m + 1$, and each $\bar{s}_l$, $l \in \llbracket 1, j \rrbracket$, is the concatenation of several consecutive s_i 's as above.

For such a path $\bar{\pi}$, one has $|\mathcal{M}(\mathcal{S}, \ell(\bar{s}_1\bar{s}_2\dots\bar{s}_j))| > 1$, so there exists a finite path $\hat{q}_0 \xrightarrow{\hat{s}_1} \hat{q}_1 \xrightarrow{\hat{s}_2} \dots \xrightarrow{\hat{s}_j} \hat{q}_j$ such that $\hat{q}_0 \in Q_0$, $\hat{q}_j \neq \bar{q}$, and for every $l \in \llbracket 1, j \rrbracket$, $\ell(\bar{s}_l) = \ell(\hat{s}_l)$. See Figure 2.8 for an illustration.

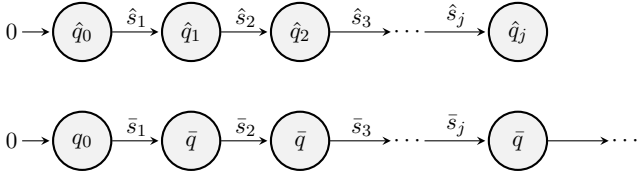


Figure 2.8: Finite paths $\hat{q}_0 \xrightarrow{\hat{s}_1} \hat{q}_1 \xrightarrow{\hat{s}_2} \dots \xrightarrow{\hat{s}_j} \hat{q}_j$ and $q_0 \xrightarrow{\bar{s}_1} \bar{q} \xrightarrow{\bar{s}_2} \dots \xrightarrow{\bar{s}_j} \bar{q}$, where $\hat{q}_j \neq \bar{q}$, $\hat{q}_0$ and q_0 may be the same.

Thus, in $\text{CC}_A(\mathcal{S})$, there is a finite path $\pi' := (\hat{q}_0, q_0) \xrightarrow{s'_1} (\hat{q}_1, \bar{q}) \xrightarrow{s'_2} \dots \xrightarrow{s'_j} (\hat{q}_j, \bar{q})$, where $\ell'(s'_1), \dots, \ell'(s'_j) \in (E')^+$. Because (i) does not hold, any state $q' \in \pi'$ with $q'(L) \neq q'(R)$ appears in path π' only once. Therefore, for all distinct $h, l \in \llbracket 1, j \rrbracket$, if $\hat{q}_h \neq \bar{q}$ or $\hat{q}_l \neq \bar{q}$, then $\hat{q}_h \neq \hat{q}_l$. Then because $\mathcal{S}$ has m states, at most $m - 1$ distinct states $(\hat{q}_l, \bar{q})$ with $\hat{q}_l \neq \bar{q}$ appear in $\{(\hat{q}_1, \bar{q}), \dots, (\hat{q}_j, \bar{q})\}$. Hence $(\bar{q}, \bar{q})$ appears in

$\{(\hat{q}_1, \bar{q}), \dots, (\hat{q}_j, \bar{q})\}$ at least twice. That is, $(\bar{q}, \bar{q})$ belongs to a reachable cycle containing an observable event and $(\hat{q}_j, \bar{q})$ is reachable from $(\bar{q}, \bar{q})$. Thus (iib) and (iic) hold. $\square$

The conditions in Theorem 2.2 can be verified in time linear in the size of the self-composition of an LFSA.

Remark 2.2. The ω -eventual strong detectability of LFSAs was verified (also in polynomial time) by using three tools — concurrent composition, observation automaton, and bifurcation automaton in the proof of Zhang and Giua (2020b, Theorem 3). However, in Theorem 2.2, only concurrent composition is adopted. Hence Theorem 2.2 largely simplifies the result in Zhang and Giua (2020b, Theorem 3). $\square$

Example 2.7. Consider LFSA $\mathcal{S}_2$ and its self-composition $CC_A(\mathcal{S}_2)$ as shown in Figure 2.5. In $CC_A(\mathcal{S}_2)$, there is no reachable, observable cycle containing a state whose two components are different, i.e., item (i) of Theorem 2.2 is not satisfied. Consider the two reachable, observable cycles $(q_0, q_0) \xrightarrow{(e_1, e_1)} (q_0, q_0)$ and $(q_1, q_1) \xrightarrow{(e_5, e_5)} (q_1, q_1)$. There is no state reachable from (q_0, q_0) such that the two components of the state are different and one of them is the same as q_0 . This also holds for (q_1, q_1) . Hence item (ii) of Theorem 2.2 is not satisfied either. Then by Theorem 2.2, $\mathcal{S}_2$ is ω -eventually strongly detectable. $\square$

Remark 2.3. Example 2.6 and Example 2.7 show that ω -eventual strong detectability is not equivalent to ω -strong detectability. Actually, the former is strictly weaker than the latter (Zhang and Giua, 2020b). In the former, the time delay depends on a specified generated event sequence, that is, it requires a universal quantifier be followed by an existential quantifier ($\forall\exists$). While in the latter, the time delay does not depend on a specified generated event sequence, that is, it requires an existential quantifier to be followed by a universal quantifier ($\exists\forall$). Hence in strong versions of detectability, $\forall\exists$ is strictly weaker than $\exists\forall$. More relations between the two definitions can be found in Zhang and Giua (2020b). $\square$

Remark 2.4. There have been a number of variants of detectability properties of LFSAs in the literature adapting to different scenarios,

e.g., strong periodic detectability⁵ and weak periodic detectability⁶ (Shu *et al.*, 2007), strong (periodic) D-detectability⁷ and weak (periodic) D-detectability⁸ (Shu and Lin, 2011b). The PSPACE-hardness of verifying these weak versions of detectability can be easily obtained by using the PSPACE-hardness result of verifying weak detectability proven by Zhang (2017). In Shu and Lin (2011b), a detector (see Definition 2.8) method was proposed to verify strong periodic detectability and strong D-detectability in polynomial time under the previously mentioned two assumptions of liveness and divergence-freeness. However, verifying strong periodic D-detectability is PSPACE-complete (Balun and Masopust, 2021b), even under the said two assumptions. Later in Zhang (2022), strong periodic detectability was verified in polynomial time without any assumption by using the detector to verify its *negation*. Strong D-detectability can also be verified in polynomial time without any assumption, but by concurrent composition via a similar procedure of verifying strong detectability as in Theorem 2.1. For other interesting variants of detectability of LFSAs and even detectability of probabilistic finite automata under the two assumptions, we refer the reader to the monograph Hadjicostis (2020) and related references therein. $\square$

2.3.2 Diagnosability

In order to define *diagnosability* for an LFSA $\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell)$, we specify a subset $E_f \subset E$ of faulty events. Diagnosability describes that one can use an observed output sequence to determine whether some faulty event has occurred. For an event sequence $s \in E^*$, $E_f \in s$ denotes that some element of E_f appears in s . In this case s is called *faulty*.

The following definition of diagnosability is chosen from Sampath *et al.* (1995) defined on live, regular languages and live, divergence-free

⁵Implies that one can periodically determine the states along *every* generated event sequence after a common time delay.

⁶Relaxing strong periodic detectability by changing *every* to *some*.

⁷Implies one can (periodically) determine that the current states do not contain any pair of previously specified states, along *every* generated event sequence after a common time delay.

⁸Relaxing strong (periodic) D-detectability by changing *every* to *some*.

LFSAs. In our monograph, because of the adoption of the concurrent composition, we can consider general LFSAs.

Definition 2.15 (Diag). Consider an LFSA $\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell)$ and a subset $E_f \subset E$ of faulty events. $\mathcal{S}$ is called E_f -diagnosable if

$$(\exists k \in \mathbb{N})(\forall s \in L(\mathcal{S}) \cap E^* E_f)(\forall s' : ss' \in L(\mathcal{S})) \\ [(|s'| > k) \implies \mathbf{D}],$$

where the diagnosis condition $\mathbf{D}$ is

$$(\forall s'' \in L(\mathcal{S}))[(\ell(s'') = \ell(ss')) \implies (E_f \in s'')], \quad (2.11)$$

recall $E^* E_f = \{se_f | s \in E^*, e_f \in E_f\}$, see **Notation** paragraph (Page 13) for the concatenation of two languages.

Intuitively, if $\mathcal{S}$ is E_f -diagnosable, then once a faulty event (e.g., the last event in s) occurs, after at least k subsequent events (e.g., s') occur, one can diagnose that some faulty event must have occurred by observing the output sequence.

Sampath *et al.* (1995) proposed a *diagnoser* method to verify diagnosability of live and divergence-free LFSAs. The diagnoser of an LFSa records state estimates along observed output sequences and also records fault propagation along transitions of states of the LFSa. And then diagnosability is verified by a relatively simple cycle condition on the diagnoser. The same as the observer mentioned above used for verifying detectability of live and divergence-free LFSAs (Shu *et al.*, 2007), the diagnoser is a variant of the powerset construction, hence also has exponential complexity. Hence diagnosability can be verified in exponential time. Also the same as the case that the observer method depends on the two assumptions of liveness/deadlock-freeness and divergence-freeness when being applied to verify detectability (Shu *et al.*, 2007), the diagnoser method also depends on these two assumptions when being applied to verify diagnosability. Later, a *twin-plant* method with polynomial-time complexity was proposed by Jiang *et al.* (2001) to verify diagnosability in polynomial time. Because in a twin plant, only observable transitions are synchronized, the method also depends on the aforementioned two assumptions. One year later, Yoo and Lafortune

(2002) proposed a *verifier* method to verify diagnosability, in polynomial time, where in a verifier, observable transitions are synchronized, unobservable transitions are also considered but their events' positions (left or right) are neglected, so that the method still depends on the two assumptions. From then on, in many papers, verification of all kinds of variants of diagnosability depends on the two assumptions. The two assumptions were removed by Cassez and Tripakis (2008) by using a generalized version of the twin-plant structure to verify the *negation* of diagnosability in polynomial time, where in the generalized version of the twin plant, observable transitions are synchronized, unobservable transitions are also considered but their events' positions (left or right) are also considered. The generalized version of the twin-plant structure and the concurrent-composition structure (Zhang and Giua, 2020b) were proposed in a similar way: they were proposed by characterizing the *negations* of inference-based properties.

In order to verify E_f -diagnosability of $\mathcal{S}$, we use the concurrent composition $CC_A(\mathcal{S}_f, \mathcal{S}_n)$ (similar to but simpler than the generalized version of the twin plant proposed by Cassez and Tripakis, 2008) of the faulty subautomaton $\mathcal{S}_f$ and the normal subautomaton $\mathcal{S}_n$, where $\mathcal{S}_f$ is obtained from $\mathcal{S}$ by only keeping faulty transitions and all their predecessors and successors, $\mathcal{S}_n$ is obtained from $\mathcal{S}$ by removing all faulty transitions. $CC_A(\mathcal{S}_f, \mathcal{S}_n)$ is computed as in Definition 2.6.

Theorem 2.3 (Zhang, 2021a). Consider an LFSA $\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell)$ and a subset $E_f \subset E$ of faulty events. $\mathcal{S}$ is not E_f -diagnosable if and only if in $CC_A(\mathcal{S}_f, \mathcal{S}_n)$, there exists a run

$$q'_0 \xrightarrow{s'_1} q'_1 \xrightarrow{e'} q'_2 \xrightarrow{s'_2} q'_3 \xrightarrow{s'_3} q'_4 \quad (2.12)$$

satisfying

$$q'_0 \in Q'_0; \quad e'(L) \in E_f; \quad |s'_3(L)| > 0. \quad (2.13)$$

Proof By definition, $\mathcal{S}$ is not E_f -diagnosable if and only if

$$\begin{aligned} & (\forall k \in \mathbb{N})(\exists s_k \in L(\mathcal{S}) \cap E^*E_f)(\exists s'_k : s_k s'_k \in L(\mathcal{S}))(\exists s''_k \in L(\mathcal{S})) \\ & [(|s'_k| > k) \wedge (\ell(s''_k) = \ell(s_k s'_k)) \wedge (E_f \notin s''_k)]. \end{aligned}$$

Choose sufficiently large k , because there are totally finitely many states in $\mathcal{S}$, by the pigeonhole principle, $\mathcal{S}$ is not E_f -diagnosable if and only if in $CC_A(\mathcal{S}_f, \mathcal{S}_n)$, there exists a run (2.12) satisfying (2.13). $\square$

Example 2.8. Consider the LFSA $\mathcal{S}_3$ in Figure 2.9. We compute concurrent composition $CC_A(\mathcal{S}_{3f}, \mathcal{S}_{3n})$ as in Figure 2.10. In $CC_A(\mathcal{S}_{3f}, \mathcal{S}_{3n})$, one sees a run $(q_0, q_0) \xrightarrow{(e_1, e_1)} (q_1, q_2) \xrightarrow{(e_2, e_2)} (q_3, q_4) \xrightarrow{(f, \epsilon)} (q_5, q_4) \xrightarrow{(u, \epsilon)} (q_5, q_4)$, which satisfies (2.13). By Theorem 2.3, $\mathcal{S}_3$ is not $\{f\}$ -diagnosable.

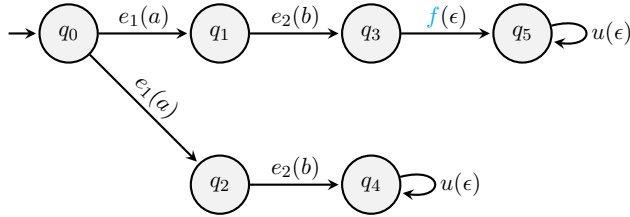


Figure 2.9: LFSA $\mathcal{S}_3$, where only event f is faulty.

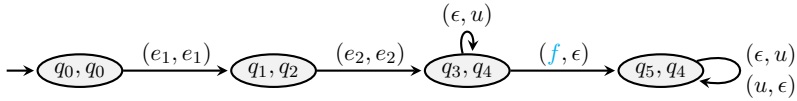


Figure 2.10: Part of $CC_A(\mathcal{S}_{3f}, \mathcal{S}_{3n})$, where $\mathcal{S}_3$ is shown in Figure 2.9.

$\square$

Remark 2.5. In the literature, how to remove the two assumptions of liveness and divergence-freeness was studied when verifying diagnosability. A widely-used method to make a non-live LFSA become live is to

add at each dead state a self-loop with an unobservable, normal event, (A)

see a recent monograph Basilio *et al.* (2021) and references therein. However, this method does not work, because this modification may change diagnosability of an LFSA as in Definition 2.15. For example, consider the following LFSA (Figure 2.11) (u and f are both unobservable):

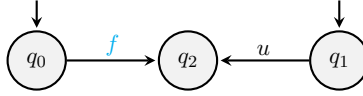


Figure 2.11: An LFSA showing that adding at each dead state a self-loop with an unobservable, normal event does not necessarily preserve diagnosability.

Choose $k = 1$, then for the unique event sequence f ended by a faulty event generated by the LFSA, there is no continuation of f . Hence the LFSA satisfies Definition 2.15 vacuously. However, after we add a self-loop on the unique dead state q_2 labeled by any non-faulty event of the LFSA or a fresh (also non-faulty) event not in the LFSA, then the modified LFSA does not satisfy Definition 2.15 any more: for all $k \in \mathbb{Z}_+$, choose event sequence $f \diamond^k$ generated by the modified LFSA, where $\diamond$ (non-faulty, could be equal to u) is the event of the newly added self-loop on q_2 , f is faulty, $|\diamond^k| \geq k$, event sequence $u \diamond^k$ generated by the modified LFSA does not contain any faulty event but $u \diamond^k$ and $f \diamond^k$ produce the same output sequence ϵ .

However, after modifying Definition 2.15 by changing “ $|s'| > k$ ” to “ $|s'| > k$ or ss' leads $\mathcal{S}$ to a dead state”, the above modification (A) preserves the modified diagnosability (Qiu and Kumar, 2006). $\square$

Remark 2.6. In Remark 2.3, we showed that in strong versions of detectability $\forall \exists$ is strictly weaker than $\exists \forall$, where $\forall \exists$ denotes ω -eventual strong detectability, $\exists \forall$ denotes strong detectability. Following this representation, the definition of diagnosability as in Definition 2.15 is in the form of $\exists \forall$. We can reformulate diagnosability in the form of $\forall \exists$ as follows:

$$(\forall s \in L(\mathcal{S}) \cap E^* E_f)(\exists k_s \in \mathbb{N})(\forall s' : ss' \in L(\mathcal{S})) \\ [(|s'| > k_s) \implies \mathbf{D}],$$

where the diagnosis condition $\mathbf{D}$ is also defined in (2.11). By the concurrent composition $CC_A(\mathcal{S}_f, \mathcal{S}_n)$, the finiteness of the number of states of LFSAs, and the pigeonhole principle, it is not difficult to see that for diagnosability, $\exists \forall$ is equivalent to $\forall \exists$ ⁹. This shows an essential

⁹Note that $\exists \forall$ is not equivalent to $\forall \exists$ in diagnosability of labeled unbounded Petri nets (Cabasino *et al.*, 2012).

difference between diagnosability and strong versions of detectability. The difference is caused by the fact that in diagnosability generated events are counted, while in detectability generated labels/outputs are counted. For example, if one output were generated, then any number of unobservable events could have been generated. This also shows that it is more difficult to study detectability than diagnosability. This difficulty becomes more significant when characterizing the complexity of verifying decentralized versions of detectability and diagnosability in LFSAs (see Section 3). $\square$

2.3.3 Predictability

Different from diagnosability, *predictability* describes whether one can use an observed output sequence to make sure some faulty event will be certain to occur.

The following definition of predictability is chosen from Genc and Lafortune (2009) defined on live regular languages and live, divergence-free LFSAs. In our monograph, because of the adoption of the concurrent composition, we can consider general LFSAs.

Definition 2.16 (Pred). Consider an LFSA $\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell)$ and a subset $E_f \subset E$ of faulty events. $\mathcal{S}$ is called *E_f -predictable* if

$$(\exists k \in \mathbb{N})(\forall s \in L(\mathcal{S}) \cap E^* E_f)(\exists s' \sqsubset s : E_f \notin s')(\forall uv \in L(\mathcal{S})) \\ [((\ell(s') = \ell(u)) \wedge (E_f \notin u) \wedge (|v| > k)) \implies (E_f \in v)].$$

Intuitively, E_f -predictability means that once a label sequence σ is generated, if some event sequence s generated by $\mathcal{S}$ ending with a faulty event has a non-faulty prefix s' whose label sequence is σ , then for every event sequence uv generated by $\mathcal{S}$ that has prefix u with $\ell(u) = \sigma$ followed by suffix v with length at least k , if u is non-faulty then v must be faulty, where k is fully determined by $\mathcal{S}$ but does not rely on any specific generated event sequence. Hence every such σ raises an alarm to definite occurrence of some faulty event.

In order to verify E_f -predictability of $\mathcal{S}$, we use the self-composition $\text{CC}_A(\mathcal{S}_n, \mathcal{S}_n)$ of the normal subautomaton $\mathcal{S}_n$. $\text{CC}_A(\mathcal{S}_n, \mathcal{S}_n)$ is also computed similarly as in Definition 2.6.

Theorem 2.4 (Zhang, 2021a). Consider an LFSA $\mathcal{S} = (Q, E, \delta, Q_0, \Sigma, \ell)$ and a subset $E_f \subset E$ of faulty events. $\mathcal{S}$ is not E_f -predictable if and only if in $\text{CC}_A(\mathcal{S}_n, \mathcal{S}_n)$, there exists a run

$$q'_0 \xrightarrow{s'_1} q'_1 \quad (2.14)$$

such that

$$q'_0 \in Q'_0; \quad (2.15a)$$

$$(q'_1(L), e_f, q) \in \delta \text{ for some } e_f \in E_f \text{ and } q \in Q; \quad (2.15b)$$

$$\text{in } \mathcal{S}_n, \text{ there is a transition cycle reachable from } q'_1(R). \quad (2.15c)$$

Proof By definition, $\mathcal{S}$ is not E_f -predictable if and only if

$$(\forall k \in \mathbb{N})(\exists s_k \in L(\mathcal{S}) \cap E^*E_f)(\forall s'_k \sqsubset s_k : E_f \notin s'_k)(\exists u_kv_k \in L(\mathcal{S})) \\ [(\ell(s'_k) = \ell(u_k)) \wedge (E_f \notin u_kv_k) \wedge (|v_k| > k)].$$

Choose sufficiently large k , because there are totally finitely many states in $\mathcal{S}$, by the pigeonhole principle, $\mathcal{S}$ is not E_f -predictable if and only if in $\text{CC}_A(\mathcal{S}_n, \mathcal{S}_n)$, there exists a run (2.14) satisfying (2.15). $\square$

One can see although the expression of E_f -predictability as in Definition 2.16 is rather complex, the expression of its negation as in Theorem 2.4 based on concurrent composition $\text{CC}_A(\mathcal{S}_n, \mathcal{S}_n)$ is rather simple.

Example 2.9. Reconsider the LFSA $\mathcal{S}_3$ in Figure 2.9. Part of the concurrent composition $\text{CC}_A(\mathcal{S}_{3n}, \mathcal{S}_{3n})$ can be obtained from Figure 2.10 by removing the transition with event (f, ϵ) . Then in $\text{CC}_A(\mathcal{S}_{3n}, \mathcal{S}_{3n})$, one sees a run $(q_0, q_0) \xrightarrow{(e_1, e_1)} (q_1, q_2) \xrightarrow{(e_2, e_2)} (q_3, q_4)$, which satisfies (2.15). Then by Theorem 2.4, $\mathcal{S}_3$ is not $\{f\}$ -predictable. $\square$

Remark 2.7. We point out that widely-used technique (e.g., in Kumar and Takai, 2010) of adding at each dead state an unobservable, normal self-loop does not necessarily preserve predictability, which is similar to the case of diagnosability (see Remark 2.5).