UNICA UNIVERSITÀ DEGLI STUDI DI CAGLIARI

DOUBLE Ph.D. DEGREE IN

Economics and Business (University of Cagliari)

and

Control Theory and Control Engineering (Xidian University)

Cycle XXXIV

TITLE OF THE Ph.D. THESIS

State Estimation of Timed Discrete Event Systems and Its Applications

Scientific Disciplinary Sector(s)

ING-INF/04 AUTOMATICA

Ph.D. Student:	Chao Gao
Supervisor (UniCa)	Prof. Alessandro GIUA
Supervisor (Xidian)	Prof. Zhiwu LI

Final exam. Academic Year 2020/2021
Thesis defence: May 2023 Session

摘要

离散事件系统是一类由事件驱动引起系统状态变化的系统。在现代工业中，离散事件系统有着广泛的应用背景。诸如制造系统、计算机网络系统、智能交通系统等均可使用离散事件系统来建模。时间结构的引入使得离散事件系统的建模及控制增加了一个维度，也为实时系统的建模和控制问题提供了研究基础。对一类系统进行建模和分析是了解该系统实际工作方式的第一步。

本文给出了两类基于单时钟的时间离散事件系统模型，即时间有限自动机(Timed Finite Automaton, TFA)与广义时间有限自动机(Generalized Timed Finite Automaton, GTFA)。在TFA与GTFA的模型中，计时函数（Timing Function）将系统的每一个变迁映射为一个允许该变迁发生的时钟区间。对TFA来说，时钟会在每次变迁发生后重置为零，并且系统不允许在一个离散状态永久性停留。而在GTFA的模型中，时钟重置函数（Clock Resetting Function）对每一个变迁指定其发生后是否重置时钟、以及重置时钟的值。本文假设TFA与GTFA所描述的时间离散事件系统是部分可观测的，其所产生的有时间信息的观测结果（Timed Observation）由一系列可观测事件与其观测时间的二元组表示。在系统逻辑结构与时间结构已知的条件下，本文分别研究了这两类模型的状态观测器设计问题。同时，本文分别展开讨论了时间离散事件系统的故障诊断、多时钟时间离散事件系统的状态观测、以及网络安全问题的攻击检测研究等时间离散事件系统状态估计的应用。本文的主要贡献如下：

1. 本文分别对TFA和GTFA提出了区域自动机(Zone Automaton) 的概念。区域自动机是一个确定有限自动机（Deterministic Finite Automaton, DFA），它的每个状态是由一个原时间离散事件系统的离散状态与一个估计时钟值范围的时间区间所构成的二元组。换句话说，这个二元组中与离散状态相关联的时间区间提供了对该离散状态下时钟值的粗略估计，区域自动机种的变迁则将时间离散事件系统表达为非时间的离散事件系统。本文提出，部分可观测的TFA/GTFA的可达性分析问题可简化为其区域自动机的可达性分析问题。

2. 基于在上文中提到的区域自动机，本文分别对TFA和GTFA提出了一种状态估计方法，该方法所得到的有限状态观测器能够描述原时间离散事件系统中所有可能的状态估计演化。给定一个有时间信息的观测结果，该方法能够给出当前可能的系统状态估计以及时钟值的估计范围。本文证明，与该观测结果一致的离散状态与其关联的时钟值能够通过计算区域自动机所产生的序列来推断。本文提出的状态估计方法可用于构建离线工作的状态观测器。与现有的在线状态

估计方法相比，可离线工作的状态观测器为进一步研究系统其他的基本属性提供了基础。本文提出，多时钟的离散事件系统状态估计能够在GTFA的模型下进行研究。多时钟GTFA的计时函数与时钟重置函数拓展到了多个时钟，每个时钟下的GTFA状态演变和可达性分析可以通过该时钟下的区域自动机所产生的序列来进行研究。

3. 作为状态估计方法的应用，本文研究了TFA的故障诊断问题。本文用有时间信息的变迁为故障行为建模，并构建了故障诊断器（Fault Recognizer）用来识别故障行为。具体来讲，本文先将故障监测器（Fault Monitor）建模成仅包含两个状态（两个状态分别为故障未发生与故障已发生）与两个事件（两个事件分别为故障事件与非故障事件）的DFA,故障诊断器则通过计算原系统的区域自动机与故障监测器与的并行组合来实现。给定一个TFA生成的观测结果，故障诊断器能够给出该观测结果下的离散状态状态估计，并判断是否可能已有故障发生。

4. 本文研究了离散事件系统的攻击检测问题。假设离散事件系统可能受到多种能够破坏系统观测结果的攻击，每种攻击都由其自己的攻击字典（Attack Dictionary）描述。此外，我们区分了连续攻击（Constant Attacks）和切换攻击（Switching Attacks）。前者仅使用一个攻击词典来连续破坏观测结果，后者则可能在每次攻击时使用不同的攻击词典。本文分别解决了非时间离散事件系统在这两类攻击下的攻击检测问题。具体来讲，针对每类攻击，本文构建了一个非确定性有限自动机（Nondeterministic Finite Automata, NFA）来描述受到攻击的系统所产生的观测结果。本文提出，这两类攻击的检测问题可分别简化为所构建的NFA的状态估计与故障诊断的经典问题。本文对时间离散事件系统的这两类攻击分别进行了形式化描述，并讨论了如何利用系统的时间结构与观测信息解决时间离散事件系统的攻击检测问题。

关键词：离散事件系统， 时间离散事件系统， 状态估计， 故障诊断， 攻击检测

ABSTRACT

Many industrial control systems can be described as discrete event systems (DES), whose state space is a discrete set where event occurrences cause transitions from one state to another. Timing introduces an additional dimension to DES modeling and control. This dissertation provides two models of timed DES endowed with a single clock, namely *timed finite automata (TFA)* and *generalized timed finite automata (GTFA)*. In addition, a *timing function* is defined to associate each transition with a time interval specifying at which clock values it may occur. While the clock of a TFA is reset to zero after each event occurs and the time semantics constrain the dwell time at each discrete state, there is an additional *clock resetting function* associated with a GTFA to denote whether the clock is reset to a value in a given closed time interval. We assume that the logical and time structure of a partially observable TFA/GTFA is known. This dissertation is dedicated to the state estimation of such timed systems based on a *timed observation*, i.e., a succession of pairs of an observable event and the time at which the event occurred. The main results are summarized as follows.

1. The notion of a *zone automaton* is introduced as a finite automaton providing a purely discrete event description of the behaviour of a TFA/GTFA of interest. Each state of a zone automaton contains a discrete state of the timed DES and a zone that is a time interval denoting a range of possible clock values. We investigate the dynamics of a zone automaton and show that one can reduce the problem of investigating the reachability of a given timed DES to the reachability analysis of a zone automaton.

2. We present a formal approach that allows one to construct offline an observer for TFA/GTFA, i.e., a finite structure that describes the state estimation for all possible evolutions. During the online phase to estimate the current discrete state according to each measurement of an observable event, one can determine which is the state of the observer reached by the current observation and check to which interval (among a finite number of time intervals) the time elapsed since the last observed event occurrence belongs. We prove that the discrete states consistent with a timed observation and the range of clock values associated with each estimated discrete state can be inferred following a certain number of runs in the zone automaton. In particular, the state estimation of timed DES under multiple clocks can be investigated in the framework of GTFA. We model such a system as a GTFA with multiple clocks, which generalizes the timing function and the clock resetting function to multiple clocks. We

show that multiple GTFA with a single clock and the associated zone automata can assist in the analysis of dynamics and reachability under each clock.

3. As an application of the state estimation approach for TFA, we assume that a given TFA may be affected by a set of faults described using timed transitions and aim at diagnosing a fault behaviour based on a timed observation. The problem of fault diagnosis is solved by constructing a zone automaton of the TFA with faults and a *fault recognizer* as the parallel composition of the zone automaton and a *fault monitor* that recognizes the occurrence of faults. We conclude that the occurrence of faults can be analyzed by exploring runs in the fault recognizer that are consistent with a given timed observation.

4. We also study the problem of attack detection in the context of DESs, assuming that a system may be subject to multiple types of attacks, each described by its own *attack dictionary*. Furthermore, we distinguish between *constant attacks*, which corrupt observations using only one of the attack dictionaries, and *switching attacks*, which may use different attack dictionaries at different steps. The problem we address is detecting whether a system has been attacked and, if so, which attack dictionaries have been used. To solve it in the framework of untimed DES, we construct a new structure that describes the observations generated by a system under attack. We show that the attack detection problem can be transformed into a classical state estimation/diagnosis problem for these new structures. We also present a formalization of the corruption of timed observations, considering attacks over a set of attack dictionaries, as a basis for investigating how knowledge of the timing structure and the time instants at which observable events occur can be exploited for the attack detection problem.

Keywords: Discrete event system, timed discrete event systems, state estimation, fault diagnosis, attack detection

List of Figures

List of Tables

List of Symbols

$\mathbb{R}_{\geq 0}$	The set of non-negative real numbers
$\mathbb{N}$	The set of non-negative integers
$\mathbb{I}$	The set of all closed time intervals
$\mathbb{I}_c$	The set of all time intervals
Q	An untimed deterministic finite automaton $Q = (X, E, \delta, x_0)$
Q_{nd}	An untimed nondeterministic finite automaton $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, X_0)$
G	A timed finite automaton $G = (X, E, \Delta, \Gamma, X_0)$ or generalized timed finite automaton $G = (X, E, \Delta, \Gamma, Reset, X_0)$
X	The set of states of an automaton
E	The set of events of an automaton
E^*	The set of all finite sequences defined over E
E_o	The set of observable events of an automaton
E_{uo}	The set of unobservable events of an automaton
E_f	The set of fault events of an automaton
x_0	The initial state of an automaton
X_0	The set of initial states of an automaton
δ	The transition function of an automaton
Δ	The transition relation of an automaton
Γ	The timing function of a timed automaton
$Reset$	The clock resetting function of a timed automaton
$L(Q)$	The generated language of an untimed automaton Q
$\mathcal{E}(G, t)$	The timed evolutions of a timed automaton G to time instant t
$\mathcal{R}(G)$	The set of timed runs generated by a timed automaton G
G_r	The region automaton of a timed automaton G
G_z	The zone automaton of a timed automaton G
$\mathcal{R}_z(G_z)$	The set of runs generated by G_z
G_f	The canonical plant for a timed automaton G with a fault f
$\mathcal{A}$	An attack dictionary
$\mathcal{F}^c_{\mathcal{A}}$	The set of constant attacks for an untimed automaton over an attack dictionary $\mathcal{A}$
$\mathcal{F}^s_{\mathcal{A}}$	The set of switching attacks for an untimed automaton over an attack dictionary $\mathcal{A}$
$Q^c_{\mathcal{A}}$	An automaton generating the corrupted observations of an

	untimed automaton under $\mathcal{F}^c_{\mathcal{A}}$
$Q^s_{\mathcal{A}}$	An automaton generating the corrupted observations of an untimed automaton under $\mathcal{F}^s_{\mathcal{A}}$

List of Abbreviations

DES	Discrete Event Systems
CPS	Cyber Physical Systems
HS	Hybrid Systems
DFA	Deterministic Finite Automaton
NFA	Nondeterministic Finite Automaton
TFA	Timed Finite Automaton
GTFA	Generalized Timed Finite Automaton
RO	Reinitialized Observations

Content

Chapter 1 Introduction

In control theory and engineering, a system refers to a collection of components or elements that work together to achieve a specific objective. While a system is a physical object, a model is a mathematical description of the behaviour of a system. The goal of systems theory is to develop a common uniform formalism for modeling, analyzing, and controlling systems of interest. Modelling and analyzing a class of systems is the first step in understanding how an existing system actually works.

Unlike a static system, which remains in a fixed state and performs a specific function until an external force is applied to it, such as storage tanks and pressure vessels, the behaviour of a dynamic system can evolve over time. Dynamic systems can be classified as *time driven* [1] and *event-driven* [2]. The former changes state over time, while the latter changes state in response to the occurrence of certain events. *Discrete-event systems* (DES) [3–7] are event-driven systems whose state space is a discrete set and transitions from one state to another are triggered by the occurrence of a *event*. The state of such a system may have logical or symbolic rather than numerical values that change in response to events that can also be described in non-numerical terms.

In a logical DES, the model does not specify the timing of event occurrences, and a common simplifying assumption is to consider only the order in which they occur. This simplification is justified when the model is to be used to study the properties of the event dynamics that are independent of specific timing assumptions, such as identifying legal sequences of operations, absence of deadlock states, etc. A number of industrial control systems can be modeled using logical DES, e.g., manufacturing system [8, 9], networked system [10–12], queueing system [13, 14], etc. Different models can be used to describe DES, including *automata* and *Petri nets*.

In the DES community, the problem of state estimation has received a lot of attention considering different observation structures, e.g., assuming that only a subset of event occurrences, or possibly also a function of the state, can be measured. Contributions to state estimation have been proposed in different formalisms, in particular automata [15–18] and Petri nets [19–24]. Most of these works are based on the notion of *observer*, namely a useful tool defined as a deterministic automaton that contains all the information to reconstruct the set of current states consistent with an observation. It allows one to move most of the burdensome parts of the computation offline and can be used efficiently for online state

estimation.

The notion of observer is fundamental and can be used as a key step to address problems of supervisory control [25], diagnosis [23, 26–28], diagnosability [29–34], in addition to characterizing a large set of dynamical properties, such as detectability [35–38], opacity [39–43] and resilience to cyber-attack [44, 45]. Some of these approaches have been extended to probabilistic settings in [46–50].

However, DESs are not inherently different from time-driven systems: a physical system that allows for a time-driven model can also be described by a DES in which the time-driven dynamics are ignored. This procedure of deriving a simpler model in a way that preserves the properties of interest while hiding the details of no interest is called *abstraction*. In turn, time-driven dynamics can be modeled by adjoining one or more clocks to a logical DES. Timing introduces a new dimension of DES modeling and control.

Over the last fifteen years, cyber-physical systems (CPS) have emerged as a key technology for developing autonomous distributed control systems [51–54]. However, one of their undesirable side effects is the fact that they are particularly exposed to cyber attacks carried out by malicious intruders. Therefore, efficient strategies for cyber security in industrial control systems are greatly needed. Attack detection, supervisory design, and investigation of how the attacks corrupt the signals are the main cyber-security issues addressed in the context of the supervisory control theory of DES. Timed DES provides a convenient framework for appropriately representing and efficiently reasoning about cyber-physical systems(CPS) subject to real-time constraints. In addition, it is also worthy of investigating how the knowledge of the timing structure and the time instants at which observable events occur can be exploited for the attack detection problem.

In this dissertation, we consider timed DESs whose timing structure is treated as a set of additional constraints that the system's evolution needs to satisfy. We provide two models of such timed DES with a single clock and different types of time semantics. One model constrains the dwell time at each discrete state, while the other model is generalized so that it can remain in a discrete state forever. We also assume that only a subset of the events is observable, i.e., a sensor is associated with such an event, while the other events are unobservable since no sensors are deployed in the system to reveal their occurrences. We assume that the logical and timed structure of a partially observable timed DES is known. Our main interest is to show how the knowledge of the timing structure and the time instants at which observable events occur can be exploited to estimate the current state of a partially observable timed DES. For both models of considered timed DES, we present a formal

approach that allows one to construct an offline observer.

As an application of the state estimation of timed DES, we assume that a timed DES may be affected by a set of faults described by timed transitions and aim at diagnosing a fault behaviour based on a timed observation. In addition, we investigate a cyber-security problem — attack detection in the framework of DESs, assuming that a plant may be subject to an attack using one or more attack dictionaries which map an observable event into a set of corrupted strings. We present a formalization of the corruption of timed observation considering attacks over a set of attack dictionaries, and a discussion on multiple attack detection of timed DES.

In the remaining part of this chapter, we first engage in an overview and motivation of the state estimation of timed DES and the attack detection of DES. After that, the organization of the dissertation and the contributions are outlined.

1.1 Overview

1.1.1 State Estimation of Timed Discrete Event Systems

In the context of DES, the notion of time has been integrated by associating a timing structure to a purely logical model. Brave and Heymann [55] introduced a time interval during which an activating event may occur, compared to the time of entry into the present state. It presents how temporal and logical behavioral features can be separated as independent treatments. Kozak [56] specifies the minimum positive time delay between occurrences of events, which requires the overall history of the event occurrences. A classic framework of the timed DES comes from Brandin and Wonham [57], where events may occur within specified time bounds, relative to the times when the events were first activated. The notion of preemptivity is proposed by introducing forcible events that can preempt clocks and other competitive events. The timed DES model allows one to characterize its performance and solve related optimization problems [3, 57–62].

From the aspect of state estimation of timed DES, the mechanism of communication delays in observations is investigated, and an algorithm to obtain state estimation under partial observation is proposed via constructing an augmented automaton that illustrates all the possible observed trajectories [63]. Observers are designed for a particular class of weighted automata, presented by Max-plus automata, which are strongly related to timed automata if a timed interpretation, i.e., the minimal time required by the process to fire the transitions, is given to weights [64]. In [65], state-based opacity of real-time automata is

investigated, and an observer (resp., a reverse observer) is proposed to concatenate current state estimates along timed output sequences generated by the real-time automata and to verify current-state opacity (resp., to verify initial-state opacity). However, the observer may not be unique (in general) due to various selections of the set of events in the observer. A class of timed and labeled finite automata called constant-time labeled automata is considered in [66], where the time information provided by time stamps can be used to refine the outcome of state estimation in various situations. In [66], events occur according to a single clock at constant times under certain time semantics. The state estimation of timed DES has also been considered in [67, 68], but no general approach concerning the construction of an observer for timed DES of models exists.

Another active area of research is that of *hybrid systems* (HS) [69–71], characterized by the interplay between discrete event and time-driven dynamics. These systems can be modeled by *hybrid automata*, a very general formalism for which, however, many properties of interest are undecidable. This has motivated several authors working in this domain to explore several subclasses of models with restricted continuous dynamics, such as the class of *timed automata* proposed by Alur and Dill [72]. The Alur-Dill timed automata are a basic event-based model endowed with a finite set of clock variables that can be updated by the occurrence of events; the occurrence of events, in turn, depends on the current values of the clocks. This model provides a convenient framework for appropriately representing and efficiently reasoning about cyber-physical systems subject to real-time constraints. While some other problems are known to be undecidable for timed automata [73–75], tractable subclasses with a restricted number of clocks [76–79] have been identified.

State estimation of timed automata has also been studied in the HS literature. In the work [72], region equivalence is defined over the set of all clock interpretations and the reachability of locations can be analysed by searching the finite quotient of a timed automaton with respect to the region equivalence. However, reachability is analyzed regardless of any observations. Tripakis studies the problem of fault diagnosis and proposes a method to check diagnosability in [80], where an online diagnoser can be implemented utilizing an online state estimator that keeps track of all the possible discrete states and the associated clock constraints after each event is measured after a delay. Thereby, the state estimation problem is theoretically solvable.

State estimation of timed automata with a single clock is explored in [81] using timed markings that represent the closure under silent transitions. It is shown that such closure can be performed as a series of subtractions between an associated interval and a regular union

of intervals that can be precomputed, and the precomputation leads the proposed approach to be more efficient compared with the online approach in [80]. Based on [81], the work [82] provides some insights to state estimation of timed automata with multiple clocks. We mention that while the state estimation of labeled timed automata has been studied in the HS literature [80–82], only online estimators have been proposed, and no general approach concerning the construction of an observer is known. This motivates us to explore the problem of state estimation of timed automata under partial observation.

1.1.2 A Cyber Security Problem: Attack Detection

CPS are intelligent interconnected systems that are particularly exposed to attacks from malicious intruders, which may pose serious threats to critical infrastructures and even lead to a system-wide standstill. In recent years, the cyber security problem of CPS has been a topic that has generated a lot of interest in various information and communication technology communities, including automatic control [83, 84].

In the domain of automatic control, the security of dynamic systems has also been approached either in time-driven systems or in DES. From the aspect of time-driven systems, security problems concerning attacks are explored considering both continuous time [85, 86] and discrete time [87, 88].

In the context of supervisory control theory of DES, the cyber security problems have been addressed in investigation of how the attacks corrupt the signals [45, 89], attack detection [44, 90–93], and supervisory control under attacks [94–103].

From the aspect of investigating the behaviour of attacks, a stealthy attack structure is proposed in [89] to select attacks that prevent the operator observing the plant from noticing when an unsafe state is reached. The concept of attackability is presented in [45], while from the perspective of the attacker, the attack under bounded sensor reading changes can be modeled as a finite state automaton.

From the aspect of attack detection, [90] investigates low rate transmission control protocol attack and presents a detection scheme using failure diagnosis of stochastic discrete event systems. [91] detects intrusions and prevents damages caused by an attacker that can hide, create or replace information. In [92] two types of deception attacks, namely replay and covert attacks, are defined and analyzed using interpreted Petri nets. In [44], a mechanism is provided to detect certain classes of attacks online so that a mitigation policy consisting in disabling all controllable events can be adopted. In [93], the notions of detectable and undetectable network attack security are proposed. The former can be recognized by the

existence of an unsafe state while the latter cannot.

Plenty of works address supervisory design for systems under attacks. A language measure method is proposed in [94] to assess the damage caused by the failure of a particular set of controllers and to determine the optimal behavior of the controlled system in the presence of an intrusion. A bipartite transition structure introduced in [95] aims to capture the game-like interaction between a supervisor and a system under attack. The notion of P-observable language is introduced in [96] and is used to characterize supervisors that are able to enforce a given specification even when multiple adversaries attack the communication channel between a plant and a supervisor. [97] models attacks on DES as inputs and outputs, and studies the supervisory control problem considering attacks on both sensors and actuators. [98] develops a method for modeling event insertion and removal attacks and presents a detection method based on the supervisory control theory framework. [99] considers the scenario where the system is subject to an actuator enablement attack and designs an attack mitigation strategy to prevent serious damage from the attack. [100] investigates the robust control problem of discrete event systems under the assumption that substitution attacks may occur. [101] investigates joint sensor-actuator cyber attacks in discrete event systems and the supervisory control problem under such attacks. [102] and [103] consider malicious attacks on sensors/actuators and present a solution methodology for synthesizing a supervisor that is robust against such attacks.

In this dissertation, inspired by the attack model of [96], we deal with the attack detection problem considering a general attack model based on the notion of *attack dictionary*, which maps an observable event into a set of corrupted strings. This attack model provides the possibility to investigate how the knowledge of the timing structure and the time instants at which observable events occur can be exploited for the attack detection problem.

1.2 Thesis Organization

This dissertation is dedicated to the state estimation of timed DES and its applications. Chapter 2 introduces the notions related to automata, state estimation of automata, and fault diagnosis of automata. The rest of the dissertation is organized as follows, accompanied by its contributions.

In Chapter 3, we provide a model of timed DES, namely *timed finite automata (TFA)*, characterized by a single clock that is reset to zero after each event occurrence. Each transition is associated with a time interval to specify when it may occur. In addition, we consider a type of time semantics that specifies the maximal dwell time at a discrete state. This chapter

considers state estimation of partially observed TFA that produces *timed observations* as a succession of pairs of an observable event and the time instant at which the event has occurred. We present a formal approach that allows one to construct offline an observer of TFA, i.e., a finite structure that describes the state estimation for all possible evolutions. The proposed solution is based on a purely discrete event description of the behaviour of the TFA, associating a finite state automaton called *a zone automaton*. We present that the problem of state reachability of the TFA can be reduced to the state reachability analysis in the associated zone automaton. An algorithm is formulated to update the state estimation of the timed automaton based on the observation of events and on the current time instant. As an application of the proposed state estimation approach, we deal with the problem of fault diagnosis in the function of measured timed observations, assuming that faulty behaviours are described by means of timed transitions.

In Chapter 4, we provide a model of timed DES generalized from TFA, called *generalized timed finite automata (GTFA)*. A GTFA is characterized by a single clock and a *clock resetting function* to denote whether the clock is reset to a value in a given closed time interval. The time semantics of this model allows a system to remain in a discrete state forever. When dealing with the state estimation problem, we then assume that observable transitions should be reset to zero to guarantee the applicability of the proposed state estimation approach, which is based on the notion of zones. We present a formal approach to construct an offline observer of this generalized model. The solution is based on determining T-reachability, which takes into account the discrete states that can be reached with an evolution producing a given observation and has a duration equal to T. The problem of T-reachability in the GTFA is reduced to the reachability analysis of the associated zone automaton. We present a discussion on the state estimation of timed DES with multiple clocks, which can be explored by extending the state estimation approach of the single-clock GTFA.

In Chapter 5, we assume that a system may be subject to multiple types of attacks, each of which is described by its own attack dictionary. Furthermore, we distinguish between *constant attacks*, which corrupt observations by using only one of the attack dictionaries, and *switching attacks*, which may use different attack dictionaries at different steps. The problem we address is that of detecting if a system has been attacked and, if so, which attack dictionaries have been used. To solve this problem, we construct a new structure that describes the observations generated by a system under attack: in particular different structures correspond to a system subject to a constant and to a switching attack. We show

that the problem of attack detection can be reduced to a classical problem of state estimation or fault diagnosis for these new structures. We note that it is worth investigating how the knowledge of the timing structure and the time instants in which observable events occur can be exploited for the attack detection problem. We present a formalization of the corruption of timed observation considering attacks over a set of attack dictionaries and discuss the perspective of attack detection of timed DES.

In Chapter 6, conclusions of this dissertation are drawn while potential future directions are suggested.

Chapter 2 Preliminaries

This chapter recalls some basic notions that will be used in this dissertation. First, we introduce the notions related to *automata*. After that, we present the classic approaches for state estimation and diagnosis of automata, respectively. Finally, we provide a sketch of timed DES by introducing the concepts of *clocks*, *timing structure*, and *time intervals*.

2.1 Automata

The state space of DES is a discrete set and transitions from one state to another are caused by *event* occurrences. Given an *alphabet* E representing a set of events, we denote by E^* the set of all finite strings on E, including the empty word ε. A string of events $s \in E^*$ is also called a *word* on E.

The concatenation of two words $s_1 \in E^*$ and $s_2 \in E^*$ is a new word $s = s_1 \cdot s_2 \in E^*$ composed by the sequence of symbols in s_1 followed by the sequence of symbols in s_2. Automata is a type of formalism of DES.

Definition 2.1 A *deterministic finite automaton* (DFA) is a four-tuple $Q = (X, E, \delta, x_0)$, where

- X is a finite set of states;
- E is an alphabet of *events*;
- $\delta : X \times E \to X$ is a transition function;
- $x_0 \in X$ is an *initial state*. □

The transition function δ is considered as a *partial function*, namely there may exist pairs $(x, e) \in X \times E$ such that $\delta(x, e)$ is not defined in Q. We write $\delta(x, e)!$ to denote that function δ is defined for the pair (x, e) in Q.

Definition 2.2 Given a DFA $Q = (X, E, \delta, x_0)$, we define the *transitive and reflexive closure* of the transition function δ as a function $\delta^* : X \times E^* \to X$ such that $\delta^*(x, s) = x_k$ if there exists a path in Q:

$$x \xrightarrow{e_1} x_1 \xrightarrow{e_2} x_2 \cdots \xrightarrow{e_k} x_k,$$

where $e_i \in E (i \in \{1, 2, \cdots, k\})$ and $s = e_1 e_2 \ldots e_k$ is a set of events. Specifically, we write $\delta^*(x, \varepsilon) = x$ for all $x \in X$. We use $\delta^*(x, s)!$ (i.e., $\delta^*(x, s)$ is defined) to denote that from state x there is a sequence of events s such that $\delta^*(x, s) = x' \in X$ holds. □

Definition 2.3 Given a DFA $Q = (X, E, \delta, x_0)$, the *language* of Q is defined as the set of all generated words $L(Q) = \{s \in E^* | \delta^*(x, s)! \in \Delta^*\} \subseteq E^*$. □

The number of events that form a word s is called its length and is denoted by $|s|$, while $|s|_e$ denotes the number of occurrence of events $e \in E$ in s.

Definition 2.4 Given a string $s \in L(Q)$, the support of s is defined as $||s|| = \{e \in E \mid |s|_e > 0\} \subseteq E$, which consists of the set of events that appear at least once in the string. □

Example 2.1 A DFA Q with $X = \{x_0, x_1, x_2\}, E = \{a, b, c, d\}$, and the initial state x_0 is graphically shown in Fig. 2.1. The transition function corresponds to Q is depicted in Table 2.1. For instance, the value x_1 at the intersection between row x_0 and column a denotes that $\delta(x_0, a) = x_1$. The symbol "-" represents that the corresponding transition is not defined. The DFA Q can model a robot that loads parts. The physical meaning corresponding to each state/event is depicted in Table 2.2. For instance, $\delta^*(x_0, ab) = x_0$ implies that after the robot grasps a part and loads it correctly, it returns to the idle state. Denoting $s = ab$, it is $|s| = 2$, $|s|_a = 1$, and $|s|_b = 1$. The support of s is $||s|| = \{a, b\}$. □

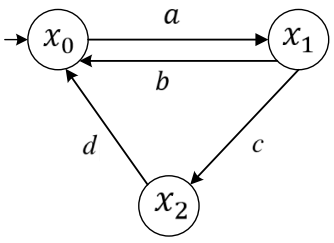


Fig. 2.1 DFA $Q = (X, E, \delta, x_0)$.

Table 2.1 Transition function corresponds to the DFA in Fig. 2.1.

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

Table 2.2 Physical meaning of states and events in Fig. 2.1.

x_0	Idle state.
x_1	Loading state.
x_2	Error state.
a	Robot grasps one part.
b	One part is correctly loaded.
c	One part is incorrectly positioned.
d	One part is repositioned.

Definition 2.5 A *nondeterministic finite automaton (NFA)* is a four-tuple $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, X_0)$, where

- X is a finite set of states;
- $E \cup \{\varepsilon\}$ is an alphabet of *events*;
- $\Delta \subseteq X \times (E \cup \{\varepsilon\}) \times X$ is the transition relation;
- $X_0 \subseteq X$ is the set of *initial states*, which may include more than one state in X. □

Definition 2.6 Given an NFA $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, X_0)$, the *transitive and reflexive closure* of the transition relation Δ is the relation $\Delta^* \subseteq X \times E^* \times X$ such that $(x, w, x_k) \in \Delta^*$ if there exists a path in Q:

$$x \xrightarrow{e_1} x_1 \xrightarrow{e_2} x_2 \cdots \xrightarrow{e_k} x_k,$$

where $e_i \in E (i \in \{1, 2, \cdots, k\})$ and $w = e_1 e_2 \ldots e_k$ is a set of events. By convention, $(x, \varepsilon, x) \in \Delta^*$ for all $x \in X$, i.e., starting from a state and generating the empty word (which may corresponds to an ε-transition or not) the automaton remains in the same state. □

Definition 2.7 Given an NFA $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, X_0)$, the *language* of Q is defined as the set of all generated words $L(Q) = \{w \in E^* | (\exists x \in X)(x_0, w, x) \in \Delta^*\} \subseteq E^*$. □

Example 2.2 Fig. 2.2 shows an NFA $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, x_0)$, where $X = \{x_0, x_1, x_2, x_3\}$, $E = \{a, b\}$, $X_0 = \{x_0\}$, and rhe transition relation is given by $\Delta = \{(x_0, \varepsilon, x_2), (x_1, a, x_1), (x_1, b, x_0), (x_1, b, x_2), (x_2, a, x_3), (x_3, a, x_1), (x_3, a, x_3)\}$. The transition relation Δ introduces two different nondeterministic primitives:

- The transition labeled with the empty word ε describes an event that occurs without being observed, namely (x_0, ε, x_2);
- Two or more transitions outgoing from the same state and having the same label describe indistinguishable events: (x_1, b, x_0) and (x_1, b, x_2), (x_3, a, x_1) and (x_3, a, x_3).

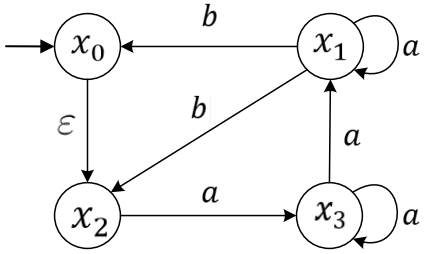


Fig. 2.2 NFA $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, x_0)$.

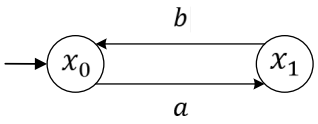
In addition, it is $(x_0, \varepsilon aab, x_2) \in \Delta^*$. □

Definition 2.8 Given two automata $Q_1 = (X_1, E_1, \delta_1, x_{01})$ and $Q_2 = (X_2, E_2, \delta_2, x_{02})$, the parallel composition of Q_1 and Q_2 is denoted by $Q_1 \| Q_2 = (X_1 \times X_2, E_1 \cup E_2, \delta, (x_{01}, x_{02}))$, where

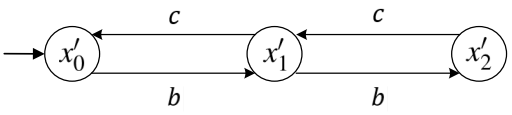
$$\delta((x_1, x_2), e) = \begin{cases} (\delta_1(x_1, e), x_2) & \text{if } e \in E_1 \setminus E_2 \\ (x_1, (\delta_2(x_2, e)) & \text{if } e \in E_2 \setminus E_1 \\ (\delta_1(x_1, e), (\delta_2(x_2, e)) & \text{if } e \in E_1 \cap E_2 \\ \text{undefined} & \text{otherwise} \end{cases}$$

□

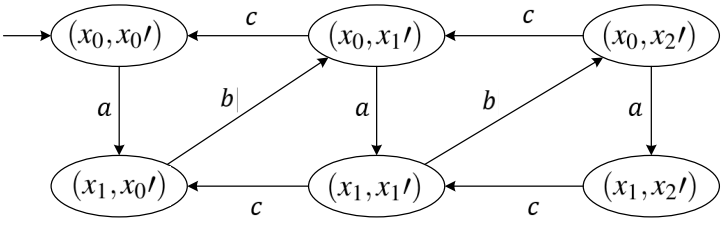
Example 2.3 Consider DFA Q_1 and Q_2 shown in Fig. 2.3(a) and Fig. 2.3(b), respectively. The parallel composition of Q_1 and Q_2 is depicted in Fig. 2.3(c). □



(a) DFA Q_1.



(b) DFA Q_2.



(c) Parallel composition $Q_1 || Q_2$.

Fig. 2.3 Parallel composition of two DFA.

An NFA Q_{nd} can always be transformed into a language-equivalent DFA, which generates the same languages as the original NFA. We denote

$$D_\varepsilon(x) = \{\bar{x} \in X \mid (x, \varepsilon, \bar{x}) \in \Delta^*\}$$

as the states reachable from x executing zero or more ε-transitions, and

$$D_e(x) = \{\bar{x} \in X \mid (x, e, \bar{x}) \in \Delta\}$$

as the states reachable from x executing exactly one e-transition. The procedure for transforming an NFA to an equivalent DFA is provided by the following algorithm.

Algorithm 1 Construction of a DFA equivalent to an NFA

Input: An NFA $Q = (X, E \cup \{\varepsilon\}, \Delta, x_0)$
Output: A DFA $Q = (X', E, \delta', x'_0)$ with $L(Q) = L(Q')$
1: let $x'_0 = D^*_\varepsilon(x_0)$, $X' = \emptyset$, and $X'_{new} = \{x'_0\}$
2: **while** $X'_{new} \neq \emptyset$ **do**
3: select a state $x' \in X'_{new}$
4: **for** each $e \in E$ **do**
5: define $\alpha(x', e) = \bigcup_{x \in x'} D_e(x)$ and $\beta(x', e) = \bigcup_{x \in \alpha(x', e)} D_\varepsilon(x)$
6: let $\bar{x}' = \beta(x', e)$ and $\delta'(x', e) = \bar{x}'$
7: **if** $\bar{x}' \notin X' \cup X'_{new}$ **then**
8: let $X'_{new} = X'_{new} \cup \{\bar{x}'\}$
9: **end if**
10: **end for**
11: let $X' = X' \cup \{x'\}$ and $X'_{new} = X'_{new} \setminus \{x'\}$
12: **end while**
13: **return** $Q = (X', E, \delta', x'_0)$.

Example 2.4 Consider again the NFA $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, x_0)$ in Fig. 2.2. The sets $D_\varepsilon(x)$, $D_a(x)$, and $D_b(x)$ for each state $x \in X$ are reported in Table 2.3. According to Algorithm 1, the DFA equivalent to Q_{nd} is depicted in Fig. 2.4. □

Table 2.3 $D_\varepsilon(x)$, $D_a(x)$, and $D_b(x)$ for each state x of Q_{nd} in Fig. 2.2.

x	$D_\varepsilon(x)$	$D_a(x)$	$D_b(x)$
x_0	$\{x_0, x_2\}$	$\emptyset$	$\emptyset$
x_1	$\{x_1\}$	$\{x_1\}$	$\{x_0, x_2\}$
x_2	$\{x_2\}$	$\{x_3\}$	$\emptyset$
x_3	$\{x_3\}$	$\{x_1, x_3\}$	$\emptyset$

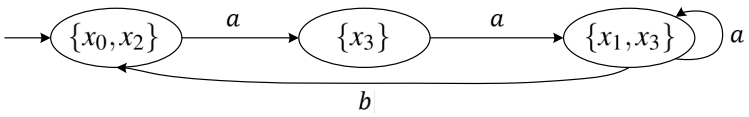


Fig. 2.4 DFA equivalent to Q_{nd} in Fig. 2.2.

2.2 State Estimation of Automata

In this section, we are interested in a different interpretation of the equivalence between DFAs and NFA that originates from systems theory. We consider partially-observed DES with an alphabet E partitioned into two disjoint subsets E_o and E_{uo}, where E_o is the set of observable events and E_{uo} is the set of unobservable events. Motivated by the notion of ε-transitions in an NFA as events that occur in the system but cannot be observed by an outside observer of the system behavior, an NFA can be seen as a model of partially-observed plant and its equivalent DFA is an observer, that allows one to estimate the state of plant.

Definition 2.9 Given a word in E^*, the *observation* is defined via the projection operator $P : E^* \longrightarrow E_o^*$ defined as $P(\varepsilon) = \varepsilon$, and for all $w \in E^*$ and $e \in E$,

$$P(we) = \begin{cases} P(w)e & \text{if } e \in E_o \\ P(w) & \text{if } e \in E_{uo}. \end{cases}$$

□

Definition 2.10 The set of states of Q consistent with a given observation w is defined as

$$\mathcal{X}(w) = \{x \in X | (\exists s \in L(Q)) P(s) = w, \delta(x_0, s) = x\}.$$

□

The sketch of the state estimation problem is depicted in Fig. 2.5. A partially-observed plant Q produces a string $s \in L(Q)$, and the mask projects s to an observation w. In plant Q, the same observation w may be generated by different runs, all starting from the initial state but possibly leading to different states. Thus there is an uncertainty on the current state of Q reached after w has been generated. State estimation aims to obtain the observer $Obs(Q)$ that computes for each word w the set of consistent states $\mathcal{X}(w)$.

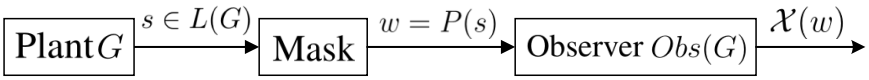


Fig. 2.5 Sketch of the state estimation for a partially-observed plant.

Remark 2.1 Given an NFA $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, x_0)$ we can use the DFA equivalent to Q_{nd} as observer, i.e., $Obs(Q_{nd}) = Q' = (X', E, \delta', x_0')$. In fact:

- the language generated by Q_{nd} and Q' are identical, i.e., $L(Q_{nd}) = L(Q')$;

- for all $w \in L(Q)$ it holds that $\mathcal{X}(w) = \delta'^*(x'_0, w)$, i.e., if the word w leads to $x' = \delta'^*(x'_0, w) \subseteq X$ in Q', then $\mathcal{X}(w) = x'$. □

Example 2.5 Consider again $Q_{nd} = (X, E \cup \{\varepsilon\}, \Delta, x_0)$ shown in Fig. 2.2. The equivalent DFA shown in Fig. 2.4 is the observer for the partially observable plant Q_{nd} by treating observable events $E_o = E$ and unobservable events $E_{uo} = \varepsilon$, denoted by $Obs(Q_{nd}) = (X_{obs}, E, \delta_{obs}, x_{0,obs})$. The state space of the $Obs(Q_{nd})$ is a subset of the power set of the state space of Q_{nd}. Considering the alphabet of Q_{nd} is $E \cup \{\varepsilon\}$, the observer $Obs(Q_{nd})$ keeps track of the estimate of the state of Q_{nd} upon transitions labeled by events in E. Given an observation $w = aaa$, it is $\mathcal{X}(w) = \{x_1, x_3\}$ according to $\delta_{obs}(x_{0,obs}, aaa) = \{x_1, x_3\}$. □

2.3 Fault Diagnosis of Automata

In many applications where the model of the system contains events that are unobservable, we may be interested in determining whether some of those unobservable events have occurred in the strings generated by the system. By modelling those unobservable events of interest as faults of a system, the problem of fault diagnosis aims to determining if one of these faults has occurred.

Consider an automaton Q that models both the normal and the faulty behavior, the alphabet can be partitioned as $E = E_o \cup E_{uo}$. The set of unobservable events can be further partitioned as $E_{uo} = E_f \cup E_{reg}$, where E_f is the set of fault events and E_{reg} is the set of regular events that do not describe a faulty behavior. Assume that Q does not contain cycles of unobservable events. The diagnosis problem consists in determining if a fault has occurred based on the observed word $w \in E_o^*$, i.e., if an evolution containing a transition labeled with a fault in E_f has produced the observation w.

Definition 2.11 Given a plant Q with alphabet $E = E_o \cup E_{uo}$ and set of fault events $E_f \subseteq E_{uo}$, a diagnosis function $\phi : E_o^* \to \{N, F, U\}$ associates to each observed word $w \in E_o^*$ a diagnosis state $\phi(w) \in \{N, F, U\}$, where

- $\phi(w) = N$ if $||s|| \cap E_f = \emptyset$ holds for all $s \in P^{-1}(w)$, namely no string s containing a fault event is consistent with w, hence no fault has occurred.
- $\phi(w) = F$ if $||s|| \cap E_f \neq \emptyset$ holds for all $s \in P^{-1}(w)$, namely all strings consistent with w contains a fault event, hence a fault has occurred certainly.
- $\phi(w) = U$ if there exists $s', s'' \in P^{-1}(w)$ such that $||s'|| \cap E_f = \emptyset$ and $||s''|| \cap E_f \neq \emptyset$. That is to say, there exists two strings s', s'' consistent with the w: s' contains a

fault event and s'' does not contain a fault event. Hence a fault may or may not have occurred. □

Example 2.6 Consider the automaton in Fig. 2.6. The set of observable events is $E_o = \{a, b, c\}$, while the set of unobservable events is $E_{uo} = \{\varepsilon_1, \varepsilon_f\}$. In particular, the set of unobservable regular events is $E_{reg} = \{\varepsilon_1\}$, and set of unobservable fault events is $E_f = \{\varepsilon_f\}$. Given three observations $w_1 = a$, $w_2 = b$, and $w_3 = aa$, it can be inferred that:

- $P^{-1}(w_1) = \{a, a\varepsilon_f\}$, $\mathcal{X}(w_1) = \{x_0, x_2\}$, and $\phi(w) = U$;
- $P^{-1}(w_2) = \{b, b\varepsilon_1\}$, $\mathcal{X}(w_2) = \{x_0, x_1\}$, and $\phi(w) = N$;
- $P^{-1}(w_3) = \{a\varepsilon_f a, a\varepsilon_f a\varepsilon_f\}$, $\mathcal{X}(w_3) = \{x_0, x_2\}$, and $\phi(w) = F$. □

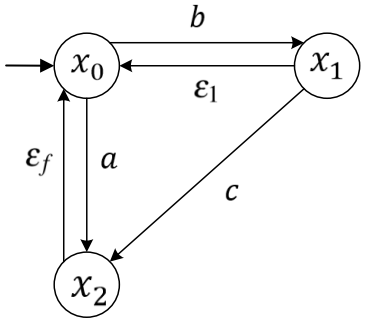


Fig. 2.6 Plant Q with a set of unobservable fault events $E_f = \{\varepsilon_f\}$.

Concerning $n(n \geq 1)$ different fault classes $E_{f,1}$, $E_{f,2}$, $\cdots$, $E_{f,n}$, one can separately determine if a fault in $E_{f,i}(1 \leq i \leq n)$ has occurred. The fault diagnosis for $E_{f,1}$, $E_{f,2}$, $\cdots$, $E_{f,n}$ can be done by solving n diagnosis problems, i.e., constructing n diagnosis functions ϕ_i for $i = 1, \cdots, n$. As shown in Fig. 2.7, given a partially observable plant Q with a set of fault events E_f, a diagnoser $Diag(Q)$ can be used to track the behavior of a plant and diagnosis prior occurrence of fault events in E_f.

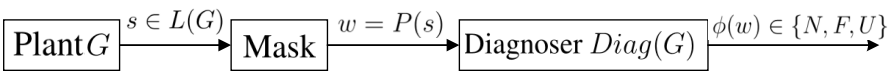


Fig. 2.7 Sketch of the fault diagnosis for a partially-observed plant.

Definition 2.12 Given an alphabet E and a set of fault events $E_f \subseteq E$, a fault monitor is defined as a DFA $M = (X_M, E, \delta_M, x_{M,0})$, where

- the set of states is $X_M = \{N, F\}$;

- the initial state is $x_{M,0} = N$;
- the transition function is $\delta_M : X_M \times E \to X_M$ such that $\delta_M(N, e) = N$ if $e \in E \setminus E_f$, $\delta_M(N, e) = F$ if $e \in E_f$, and $\delta_M(F, e) = F$ for all $e \in E$. □

Given a string $s \in E^*$, it holds that $\delta_M(N, e) = N$ if and only if $||s|| \cap E_f = \emptyset$, i.e., strings involving no a fault event lead to state N in the fault monitor, while strings involving one or more fault event lead to state F.

Definition 2.13 Given DFA $Q = (X, E, \delta, x_0)$ with alphabet E and a set of fault events $E_f \subseteq E$, let $M = (X_M, E, \delta_M, x_{M,0})$ be its fault monitor. The *fault recognizer* for Q is defined as the DFA $Rec(Q) = Q \parallel M = (X_R, E, \delta_R, x_{R,0})$, where

- the set of states is $X_R \subseteq X \times \{N, F\}$;
- the initial state is $x_{R,0} = (x_0, N)$;
- the transition function is $\delta_R : X_R \times E \to X_R$. Given a string $s \in L(Q)$ such that $\delta(x_0, s) = x$, it is $\delta_R((x_0, N), s) = (x, N)$ if $||s|| \cap E_f = \emptyset$; otherwise, $\delta_R((x_0, N), s) = (x, F)$. □

Definition 2.14 Given a DFA Q with alphabet E and a set of fault events $E_f \subseteq E$, let $Rec(Q)$ be its fault recognizer. The diagnoser of Q is the DFA $Diag(Q) = Obs(Rec(Q)) = (Y, E_o, \delta_y, y_0)$, where

- $Y \subseteq (X \times \{N\}) \cup (X \times \{F\})$, i.e., each state of the diagnoser is a set of pairs $y = \{(x_1, \gamma_1), \cdots, (x_k, \gamma_k)\}$ such that $x_i \in X$ and $\gamma_i \in \{N, F\}$ for $i = 1, 2, \cdots, k$.
- $$\delta_y^*(y_0, w) = \{(x, N) \mid (\exists s \in P^{-1}(w))\delta^*(x_0, s) = x, ||s|| \cap E_f = \emptyset\}$$
$$\cup\{(x, F) \mid (\exists s \in P^{-1}(w))\delta^*(x_0, s) = x, ||s|| \cap E_f \neq \emptyset\},$$
i.e., starting from y_0 the string w leads to state containing: (a) all pairs (x, N) where x can be reached in Q executing a string consistent with w that does not contain a fault event; and (b) all pairs (x, F) where x can be reached in Q executing a string consistent with w that contains a fault event. □

Diagnosers are similar to observers with the difference that labels N and F are attached to the states of Q in the states of $Diag(Q)$, where N implies a fault event has occurred and N implies a fault event has not occurred. We associate a diagnosis value $\phi(y)$ to each state $y = \{(x_1, \gamma_1), \cdots, (x_k, \gamma_k)\}$ of $Diag(Q)$ such that

- $\phi(y) = N$ if $\gamma_i = N$ for $i = 1, \cdots, k$;
- $\phi(y) = F$ if $\gamma_i = F$ for $i = 1, \cdots, k$;
- $\phi(y) = U$ if there exist $i, j \in \{1, \cdots, k\}$ such that $\gamma_i = N$ and $\gamma_j = F$.

Example 2.7 Consider again the automaton Q in Fig 2.6 with $E_{uo} = \{\varepsilon_1, \varepsilon_f\}$ and $E_f = \{\varepsilon_f\}$. The diagnoser of Q is shown in Fig. 2.8, denoted by $Diag(Q) = (Y, E_o, \delta_y, y_0)$. The details for each state $y \in Y$ is reported in Table 2.4. □

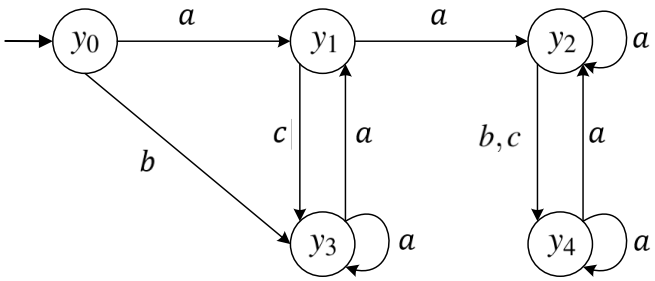


Fig. 2.8 Diagnoser $Diag(Q)$, where Q in Fig 2.6.

Table 2.4 Details for each state of $Diag(Q) = (Y, E_o, \delta_y, y_0)$ in Fig. 2.8.

$y \in Y$	$\phi(y)$
$y_0 = \{(x_0, N)\}$	N
$y_1 = \{(x_2, N), (x_0, F)\}$	U
$y_2 = \{(x_0, F), (x_2, F)\}$	F
$y_3 = \{(x_0, N), (x_1, N)\}$	N
$y_4 = \{(x_0, F), (x_1, F)\}$	F

2.4 Sketch of Timed DES

In this section, we concentrate on timed models of DES. The most straightforward instance of time-driven dynamics is the case of adjoining one or more *clocks* to the untimed DES model, resulting in a timed model. The modelling framework for timed DES can be created by associating untimed DES with a *timing structure* that determines when the clocks would have to be reset and describes the timing constraints associated with the occurrences of each event. A timing structure can be provided by introducing the notion of the *time semantics*, which specify: (a) the enabled events at each discrete state, and (b) if the system can stay at a discrete state forever.

The production of a timed DES consists of event sequences associated with a certain form of timing information rather than sequences of untimed events in the case of untimed DES. For instance, let t_k, where $k = 1, 2, \cdots$, denote the time instant when the k-th event occurs, then a production of a timed DES can be described by the sequence $(e_1,\ t_1)(e_2,\ t_2)\cdots(e_k,\ t_k)$. Note that $t_1, \cdots, t_k$ are absolute time, namely $t_1 \leq t_2 \leq \cdots \leq t_k$. The evolution of a timed DES is illustrated via the following example.

Example 2.8 Consider the DFA $Q = (X, E, \delta, x_0)$ in Example. 2.1 endowed with clocks θ_1 and θ_2, where θ_1 records the time dwelled at each discrete state and θ_2 records the time for a robot positioning a part correctly. Given a production $(a, t_1)(c, t_2)(d, t_3)(a, t_4)(b, t_5)$, Fig. 2.9 and Fig. 2.10 report the evolution of Q and the evolution of clocks with time elapses, respectively. In more detail, it evolves from the initial state x_0 to x_1 with the occurrence of event a at t_1. Meanwhile, clock θ_1 is reset to 0 and clock θ_2 continues the previous value at t_1. Next, events c and d occurs at t_2 and t_3, respectively. Both clocks θ_1 and θ_2 are reset to 0 at t_3. □

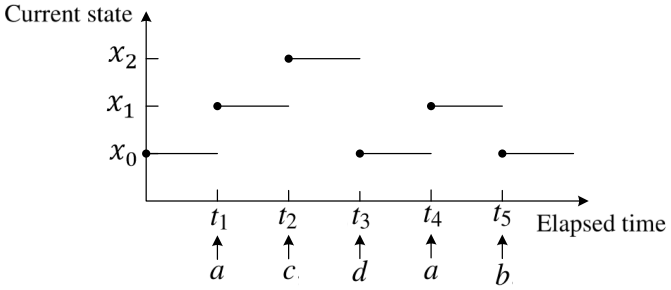
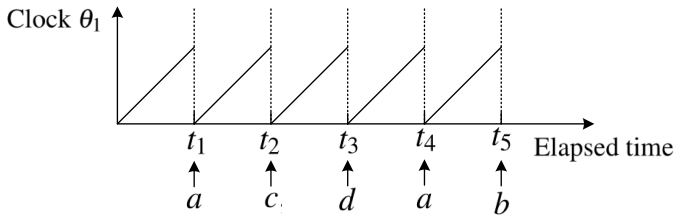


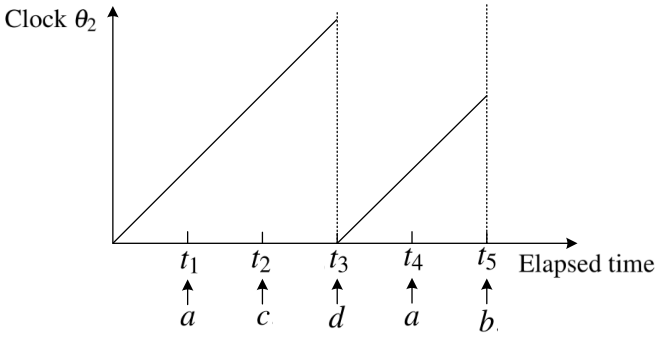
Fig. 2.9 State evolution of the DFA Q in Example. 2.1 with time elapses.

In this dissertation, we consider two types of timed DES models in Chapter 3 and Chapter 4, respectively. The time semantics in Chapter 3 constrains the dwell time at each discrete state, while the time semantics in Chapter 4 allows the system stays at a discrete state forever. Both of the models are endowed with a single clock, and their timing structures specify timing constraints of event occurrences using *time intervals*. We introduce the following notions related to time intervals and used throughout this dissertation.

By denoting the sets of non-negative real numbers and natural numbers as $\mathbb{R}_{\geq 0}$ and $\mathbb{N}$, respectively, the set of real numbers in $\mathbb{R}_{\geq 0}$ lying between a lower bound $I_l \in \mathbb{N}$ and an upper bound $I_u \in \mathbb{N} \cup \{+\infty\}$ is said to be a *time interval*. A *closed time interval* is denoted by $[I_l, I_u]$. In addition, an open segment (I_l, I_u) and a semi-open segment $[I_l, I_u)$ or $(I_l, I_u]$ can also be *time intervals*. We denote the set of all time intervals and the set of all closed time intervals as $\mathbb{I}$ and $\mathbb{I}_c$, respectively, where $\mathbb{I}_c \subseteq \mathbb{I}$.



(a) Evolution of clock θ_1.



(b) Evolution of clock θ_2.

Fig. 2.10 Evolution of clocks θ_1 and θ_2 associated with the DFA Q in Example. 2.1.

Definition 2.15 The *addition operation on two time intervals* $I_1, I_2 \in \mathbb{I}$ is defined as $I_1 \bigoplus I_2 = \{t_1 + t_2 \in \mathbb{R}_{\geq 0} \mid t_1 \in I_1, t_2 \in I_2\}$. That is to say, given $I_i = [I_{l,i}, I_{u,i}], i = 1, 2$, we have $I_1 \bigoplus I_2 = [I_{l,1} + I_{l,2}, I_{u,1} + I_{u,2}]$. The addition operation can be extended to n $(n > 1)$ time intervals in a set $\{I_1, \cdots, I_n\}$, i.e., $I_1 \bigoplus \cdots \bigoplus I_n = ((I_1 \bigoplus I_2) \bigoplus \cdots) \bigoplus I_n$, denoted as $\bigoplus_{i=1}^{n} I_i$. □

Definition 2.16 The *distance range between two time intervals* $I_1, I_2 \in \mathbb{I}$ is defined as $D(I_1, I_2) = \{|t_1 - t_2| \mid t_1 \in I_1, t_2 \in I_2\}$. □

Example 2.9 Given two time intervals $I_1 = [0, 1)$ and $I_2 = [3, 4]$, it holds that $I_1 \bigoplus I_2 = [3, 5)$ and $D(I_1, I_2) = (2, 4]$. □

Chapter 3 State Estimation of Timed Finite Automata

In this chapter, we introduce a model of timed DES, called *timed finite automata (TFA)*, characterized by a single clock that is reset to zero after each event occurrence. Each transition is associated with a time interval to specify when it may occur. In addition, we consider a type of time semantics that specifies the *maximal dwell time at a discrete state*. This chapter considers partially observed TFA, where some of the event occurrences are observable, and others are unobservable.

We consider the information coming from the observation of observable events at certain time instants, and aim at estimating and updating the set of states consistent with the whole observation. The proposed solution is based on a purely discrete event description of the behaviour of the TFA, associating a finite state automaton called a *zone automaton*. We present that the problem of state reachability of the TFA can be reduced to the state reachability analysis in the associated zone automaton. An algorithm is formulated to update the state estimation of the timed automaton based on the observation of events and on the current time instant. As an application of the proposed state estimation approach, we address the problem of fault diagnosis in the function of measured timed observations, assuming that faulty behaviours are described by means of timed transitions.

This chapter is divided into five sections. The first section introduces the TFA model and related notions. The second section introduces the notions of *region automaton* and zone automaton, which are NFA that provide a purely discrete event description of the behaviour of a TFA of interest. Compared to the region automaton, the zone automaton is more compact and thus more efficient in practical cases; consequently, the state estimation approach is based on the zone automaton. We then study the dynamics of a zone automaton and show that the problem of investigating the reachability of a TFA can be reduced to the reachability analysis of a zone automaton.

In the third section, we deal with the state estimation problem, considering two sub-problems: (a) state estimation as time passes without receiving any new observation; (b) state estimation based on the observation of events and the current time. An algorithm summarizes our proposed approach to compute the set of states consistent with a timed observation. In the fourth section, we assume that the timed system may be affected by a set of faults described by timed transitions, the occurrence of which changes the state of the plant and resets the clock. We present an approach to construct a fault recognizer that not

only provides the estimated discrete states associated with a range of clock values, but also detects the occurrence of faults. The fault diagnosis problem is solved by investigating the reachability of the fault recognizer. The fifth section concludes this chapter.

3.1 Timed Finite Automata

This section provides preliminary notions used throughout this chapter.

Definition 3.1 A *timed finite automaton (TFA)* is a 5-tuple $G = (X, E, \Delta, \Gamma, X_0)$, where

- X is a finite set of states,
- E is a finite set of events,
- $\Delta \subseteq X \times E \times X$ is a transition relation,
- $\Gamma : \Delta \to \mathbb{I}$ is a timing function,
- $X_0 \subseteq X$ is the set of initial state. □

We assume that the automaton operates under a single clock, which is reset at each event occurrence, i.e., each time the system enters in a state. The transition relation and the timing function specify the dynamics of the TFA. In more detail, given two states $x, x' \in X$ and an event $e \in E$, $(x, e, x') \in \Delta$ denotes that the occurrence of event e leads to state x' when the TFA is in state x. The timing function Γ maps the transition (x, e, x') to a time interval, which specifies a range of clock values at which the event e may occur. We further define $\Gamma_l : \Delta \to \mathbb{N}$ (resp., $\Gamma_u : \Delta \to \mathbb{N} \cup \{+\infty\}$) as the lower (resp., upper) timing function associating a transition in Δ to the left (resp., right) bound of the time interval associated with it. Therefore $\Gamma((x, e, x')) = [\Gamma_l((x, e, x')), \Gamma_u((x, e, x'))]$. In simple words, a TFA $G = (X, E, \Delta, \Gamma, X_0)$ is a NFA $G = (X, E, \Delta, X_0)$ endowed with a time structure Γ that associates with each transition in Δ a time interval in $\mathbb{I}_c$.

A TFA $G = (X, E, \Delta, \Gamma, X_0)$ can be represented by a graph, where a state $x \in X$ corresponds to a node, and each initial state in X_0 is marked by an input arrow. For each transition $(x, e, x') \in \Delta$ with $\Gamma((x, e, x')) = I$, there exists a directed edge from x to x' labeled with the symbol e and the time interval I.

Example 3.1 Consider the TFA $G = (X, E, \Delta, \Gamma, X_0)$ with $X = \{x_0, x_1, x_2\}$, $E = \{a, b, c\}$, $\Delta = \{(x_0, a, x_1), (x_0, a, x_2), (x_1, b, x_2), (x_2, c, x_0)\}$ and $X_0 = \{x_0\}$. Let the timing function Γ be defined as in Table 3.1. The graphical representation of G is visualized in Fig. 3.1. □

Table 3.1 The timing function of the TFA G in Fig. 3.1.

$\delta \in \Delta$	(x_0, a, x_1)	(x_0, a, x_2)	(x_1, b, x_2)	(x_1, b, x_2)
$\Gamma(\delta)$	$[0, 1]$	$[1, 2]$	$[1, 2]$	$[0, 1]$
$\Gamma_u(\delta)$	1	2	2	1

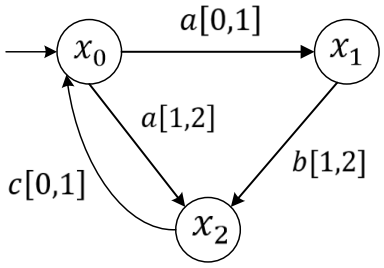


Fig. 3.1 TFA G.

Definition 3.2 Given a TFA $G = (X, E, \Delta, \Gamma, x_0)$, a *timed run* p of length k is a sequence of $k+1$ states $x_{(i)} \in X$, $i = 0, \cdots, k$, and k pairs $(e_i, t_i) \in E \times \mathbb{R}_{\geq 0}$, $i = 1, \cdots, k$, represented as

$$\rho : x_{(0)} \xrightarrow{(e_1,t_1)} x_{(1)} \xrightarrow{(e_2,t_2)} x_{(2)} \xrightarrow{(e_3,t_3)} \cdots \xrightarrow{(e_k,t_k)} x_{(k)}$$

such that the following two conditions are satisfied for all $i = 1, \cdots, k$:

$$(x_{(i-1)}, e_i, x_{(i)}) \in \Delta, \tag{3-1}$$

$$t_i - t_{i-1} \in \Gamma((x_{(i-1)}, e_i, x_{(i)})), \tag{3-2}$$

where $t_0 = 0$. The set of timed runs generated by G is denoted as $\mathcal{R}(G)$. □

In addition, the following notations are used in this chapter:

- $\sigma(\rho) = (e_1, t_1)(e_2, t_2) \cdots (e_k, t_k) \in (E \times \mathbb{R}_{\geq 0})^*$ is the *timed word generated by run* ρ;
- $S(\sigma(\rho)) = e_1 e_2 \cdots e_k$ is the *logical word generated by timed run* ρ, where $S : (E \times \mathbb{R}_{\geq 0})^* \to E^*$;
- $x_{st}(\rho) = x_{(0)}$ is the *starting state of a timed run* ρ;
- $x_{en}(\rho) = x_{(k)}$ is the *ending state of a timed run* ρ;
- $t_{en}(\rho) = t_k$ is the *ending time of run* ρ.
- $T(\rho) = t_k$ is the *duration of* ρ. According to Definition 2.15, it clearly implies $T(\rho) \in \bigoplus_{i=0}^{k-1} \Gamma(x_{(i)}, e_{i+1}, x_{(i+1)})$.

Given a timed run ρ of length 0 that only contains the starting state $x_{(0)}$ and no transition, the logical word and the timed word generated by ρ are denoted respectively as $S(\sigma(\rho)) = \varepsilon$ and $\sigma(\rho) = \lambda$, where λ denotes the *empty timed word* in $E \times \mathbb{R}_{\geq 0}$. For the timed word $\sigma(\rho)$ generated from an arbitrary timed run ρ, it is $\lambda \cdot \sigma(\rho) = \sigma(\rho) = \sigma(\rho) \cdot \lambda$.

Example 3.2 Consider a possible run of the TFA in Fig. 3.1. A timed run $\rho : x_0 \xrightarrow{(a,0.5)} x_1 \xrightarrow{(b,2)} x_2 \xrightarrow{(c,2)} x_0$ has length 3: it starts from x_0 at the initial time $t_0 = 0$ and produces the logical word $S(\sigma(\rho)) = abc$ leading to state $x_{en}(\rho) = x_0$. The timed word $\sigma(\rho) = (a, 0.5)(b, 2)(c, 2)$ corresponds to events a, b, and c occurring at time instants $t_1 = 0.5$, $t_2 = 2$, and $t_3 = 2$, respectively. The final time of the run is $t_{en}(\rho) = 2$. The production involves 3 transitions, namely (x_0, a, x_1), (x_1, b, x_2), and (x_2, c, x_0). □

Different types of semantics pose additional constraints on how long a TFA dwells at each state while generating a timed run. In this paper we consider a type of time semantics that specifies the *maximal dwell time* at a state.

Definition 3.3 Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, the *m*aximal dwell time at state $x \in X$ is defined as $d_{max}(x) = \max\{\Gamma_u((x, e, x'))|(x, e, x') \in \Delta\}$ if there exist $x' \in X$ and $e \in E$ such that $(x, e, x') \in \Delta$; otherwise $d_{max}(x) = \infty$. □

A TFA cannot stay in $x \in X$ if the clock takes a value larger than the maximal dwell time at x, i.e., $d_{max}(x)$. If there exists no enabled transition at $x \in X$, then $d_{max}(x) = \infty$, implying that G can stay at x indefinitely. Meanwhile, if there exists one or more enabled transitions at x, the maximal dwell time at x is equal to the maximum upper bound of the intervals of such transitions. It implies that all such transitions are candidates to occur. However, a transition has to be fired once the clock at x reaches the maximal dwell time at x.

Example 3.3 For the TFA in Fig. 3.1, it is $d_{max}(x_0) = 2$, which implies that the TFA G cannot remain in state x_0 while the clock value is larger than 2. Analogously, it is $d_{max}(x_1) = 2$ and $d_{max}(x_2) = 1$. □

Definition 3.4 Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, a timed run ρ of length $k \geq 0$ and a time instant $t \in \mathbb{R}_{\geq 0}$, a *timed evolution of G from 0 to t* is defined by a pair $(\sigma(\rho), t) \in (E \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$, where $0 \leq t - t_{en}(\rho) \leq d_{max}(x_{en}(\rho))$. Furthermore, we denote as

$$\mathcal{E}(G, t) = \{(\sigma(\rho), t) \mid \begin{array}{c} (\exists \rho \in \mathcal{R}(G))\ x_{st}(\rho) \in X_0, \\ 0 \leq t - t_{en}(\rho) \leq d_{max}(x_{en}(\rho))\} \end{array}$$

the *timed language of G from 0 to t*. □

In other words, a timed evolution of G from 0 to t is defined as a pair whose first entry is a timed word $\sigma(\rho)$, where ρ starts at 0 from an initial state in X_0, and whose second entry is the time instant t, where the time semantics constrains the time that the system may stay in the ending state $x_{en}(\rho)$, namely $t - t_{en}(\rho)$, to be less than or equal to the maximal dwell time of $x_{en}(\rho)$. The timed language of G from 0 to t contains all possible timed evolutions of G from 0 to t.

Definition 3.5 Given a TFA $G = (X, E, \Delta, \Gamma, x_0)$ and a timed run p, a timed word $\sigma(\rho) \in (E \times \mathbb{R}_{\geq 0})^*$ generated by run ρ can be written as

$$\sigma(\rho) = \sigma(\rho_1) \cdot \sigma(\rho_2)$$

where ρ_1 and ρ_2 are runs that can be generated by G such that $x_f(\rho_1) = x_{ini}(\rho_2)$, then $\sigma(\rho_1)$ is called a *prefix* of $\sigma(\rho)$, denoted as $\sigma(\rho_1) \preceq \sigma(\rho)$. □

3.2 Region Automaton and Zone Automaton

In this section, we introduce two models, namely *region automaton* and *zone automaton*, that are NFA providing a purely discrete event description of the behaviour of a TFA of interest. Each state of the region (resp., zone) automaton consists of a pair whose first element is a discrete state of the TFA and whose second element is a region (resp., zone), specifying a range of clock values. In other words, each state of the region (resp., zone) automaton is associated with a single discrete state but provides only a coarse estimate of the clock value.

In detail, a region automaton is constructed based on the equal partitioning of the time dwell at each state into time regions. In contrast, a zone automaton is proposed by partitioning the time spent at each state into zones, which can be determined by analysing the firability of transitions from that state; consequently, the zone automaton is more compact and thus more efficient in practical cases. After that, we investigate the dynamics of a zone automaton and show that the problem of investigating the reachability of a TFA can be reduced to the reachability analysis of a zone automaton.

This section is divided into three subsections. The first and second subsections introduce the notions of region automaton and zone automaton, respectively. The third subsection analyses the dynamics of a zone automaton.

3.2.1 Region Automaton

In this section we introduce a region automaton, which is based on partitioning the time intervals into regions according to the following definition.

Definition 3.6 The *set of regions* of a time interval $[h, k] \in \mathbb{I}$ is defined as

$$R_{[h,k]} = \{[h, h], (h, h+1), [h+1, h+1], \cdots, [k, k]\} \tag{3-3}$$

and the *successive region* of $r \in R_{[h,k]}$ is defined as

$$\phi(r) = \begin{cases} [j, j] & \text{if } r = (j-1, j) \\ (j-1, j) & \text{if } r = [j-1, j-1] \end{cases} \tag{3-4}$$

where $h < j \leq k$. □

In more detail, a generic time interval $[h, k] \in \mathbb{I}$ is divided into $2(k-h)+1$ regions consisting in the integer points belonging to the interval and the open segments between them. The successive region of a closed region is an open region and vice versa. Given a region $r \in R_{[h,k]} \setminus \{[k, k]\}$, it is obvious that $\phi(r) \in R_{[h,k]}$. Given $t \in r$ and $t' \in \phi(r)$, $\tau = t' - t \in (0, 1)$ is referred to as a *time transfer unit* which leads a time instant in a region to another time instant in its successive region.

Based on the regions, we explore the time evolution of a state in a TFA by the following definition.

Definition 3.7 Given a TFA $G = (X, E, \Delta, \Gamma, x_0)$,

$$R(x) = \begin{cases} R_{[0, d_{max}(x)]} & \text{if } d_{max}(x) \neq +\infty, \\ \{[0, 0], (0, +\infty)\} & \text{if } d_{max}(x) = +\infty. \end{cases} \tag{3-5}$$

is defined by the set of regions of a state $x \in X$. □

When entering a state $x \in X$, the clock is reset. If the maximal dwell time at x is not $+\infty$, which implies that there exists an output transition from x, then the set of regions of state x includes all regions in the interval $[0, d_{max}(x)]$, i.e., the regions to which the clock may belong while the system is in a state x. While the system is in state x with a clock value $t \in r$, an elapsed time τ leads the clock to $t' \in \phi(r)$ if it is $\phi(r) \in R(x)$. If the maximal dwell time at x equals to $+\infty$, indicating that the system may stay at x forever, then the set of regions of x is composed of two regions, namely $[0, 0]$ and $(0, +\infty)$. An elapsed time τ leads the clock from $t \in [0, 0]$ to $t' \in (0, +\infty)$. To study the time evolution and state evolution of a TFA, we introduce the notion of region automaton as follows.

Definition 3.8 Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, the *region automaton* of G is an automaton $G_r = (V, E_\tau, \Delta_r, V_0)$, where

- $V \subseteq X \times \bigcup_{x \in X} R(x)$ is a finite set of states,
- $E_\tau = E \cup \{\tau\}$ is an alphabet,
- $\Delta_r \subseteq V \times E_\tau \times V$ is a transition relation, and
- $V_0 = \{(x, [0, 0]) \mid x \in X_0\} \subseteq V$ is the initial state. □

We use a region automaton to describe the time evolution and state evolution of a TFA. Each state of a region automaton is the Cartesian product of a state $x \in X$ and a region $r \in R(x)$; the alphabet contains the symbol τ (which corresponds to time evolution) and a finite set of events E (which cause state evolution). The transition function specifies the dynamics of the automaton: if $\delta(v, e, \bar{v}) \in \Delta_r$ then the occurrence of event $e \in E$ leads to state $\bar{v}$ from v; if $\delta(v, \tau, v') \in \Delta_r$ then a transition labeled with a time transfer unit leads to state v' from v. The initial state is the pair $(x_0, [0, 0])$ denoting that the initial state of G is x_0 and the clock is initialized at 0.

Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, Algorithm 2 is proposed to construct the region automaton $G_r = (V, E_\tau, \Delta_r, v_0)$. It works as follows. We first initialize V and V_0 as $\{(x, [0, 0]) \mid x \in X_0\}$, the alphabet E_τ as the union of E and $\{\tau\}$, and a set V_{new} containing only v_0. A loop is iterated while the condition $V_{new} \neq \emptyset$ holds. A state $v = (x, r)$ in the set V_{new} is selected. If the maximal dwell time at the state x equals to $+\infty$ and $r = [0, 0]$, we define a state $\bar{v} = (x, (0, +\infty))$ and a transition $(v, \tau, \bar{v}) \in \Delta_r$. If the maximal dwell time at the state x is not equal to $+\infty$ and the successive region of r is a region in $R(x)$, we define a state $\bar{v} = (x, \phi(r))$ and a transition $(v, \tau, \bar{v}) \in \Delta_r$. We finally add the state $\bar{v}$ to the set V_{new}. For each $e \in E$, if there exists a transition (x, e, x') which can fire when the value of the timer belongs to r, a transition $(v, e, v') \in \Delta_r$ is defined with $v' = (x', [0, 0])$. State v' is added to V_{new} if it is not already in $V \cup V_{new}$. At the end of each while loop, we add the selected state v to V and delete it from V_{new}. The while loop stops once V_{new} is empty, i.e., all states in V_{new} have been explored. The algorithm returns $G_r = (V, E_\tau, \Delta_r, V_0)$, which is the region automaton of $G = (X, E, \Delta, \Gamma, X_0)$. Like any automaton, a region automaton can be represented as a graph. In particular, here we represent nodes as squares.

Example 3.4 Consider the TFA $G = (X, E, \Delta, \Gamma, X_0)$ in Fig. 3.1, the region automaton $G_r = (V, E_\tau, \Delta_r, V_0)$ constructed according to Algorithm 2 is shown in Fig. 3.2. The set of

Algorithm 2 Construction of the region automaton associated with a TFA

Input: A TFA $G = (X, E, \Delta, \Gamma, X_0)$
Output: The region automaton $G_r = (V, E_\tau, \Delta_r, V_0)$ of G
1: let $V = V_0 = \{(x, [0,0]) \mid x \in X_0\}$, $E_\tau = E \cup \{\tau\}$, $\Delta_r = \emptyset$, and $V_{new} = V_0$
2: **while** $V_{new} \neq \emptyset$ **do**
3: select a $v = (x, r) \in V_{new}$
4: **if** $(d_{max}(x) = +\infty) \wedge (r = [0,0])$ **then**
5: let $\bar{v} = (x, (0, +\infty))$ and $\Delta_r = \Delta_r \cup \{(v, \tau, \bar{v})\}$
6: **end if**
7: **if** $(d_{max}(x) \neq +\infty) \wedge (\phi(r) \in R(x))$ **then**
8: let $\bar{v} = (x, \phi(r))$ and $\Delta_r = \Delta_r \cup \{(v, \tau, \bar{v})\}$
9: let $V_{new} = V_{new} \cup \{\bar{v}\}$
10: **end if**
11: **for** each $e \in E$ **do**
12: **if** $\exists x' \in X$ s.t. $((x, e, x') \in \Delta) \wedge (r \in \Gamma(x, e, x'))$ **then**
13: let $\bar{v} = (x', [0,0])$ and $\Delta_r = \Delta_r \cup \{(v, e, \bar{v})\}$
14: **end if**
15: **if** $\bar{v} \notin V \cup V_{new}$ **then**
16: let $V_{new} = V_{new} \cup \{\bar{v}\}$
17: **end if**
18: **end for**
19: let $V = V \cup \{v\}$ and $V_{new} = V_{new} \setminus \{v\}$
20: **end while**
21: **return** $G_r = (V, E_\tau, \Delta_r, V_0)$.

states is equal to

$$V = \{(x_0, [0,0]), (x_0, (0,1)), (x_0, [1,1]), (x_0, (1,2)), (x_0, [2,2]), (x_1, [0,0]),$$

$$(x_1, (0,1)), (x_1, [1,1]), (x_1, (1,2)), (x_1, [2,2]), (x_2, [0,0]), (x_2, (0,1)), (x_2, [1,1])\}.$$

It describes all the possible states in which G can be with the corresponding possible regions to which the clock may belong. In particular, $V_0 = \{(x_0, [0,0])\}$ is the initial state. The alphabet is $E_\tau = \{a, b, c, \tau\}$. For instance, G_r starts from $v_0 = (x_0, [0,0])$, it moves to $v_1 = (x_0, (0,1))$ by transition (v_0, τ, v_1), and to $v_2 = (x_1, [0,0])$ by transition (v_0, a, v_2). □

3.2.2 Zone Automaton

Consider the case of a single transition $(x, e, x') \in \Delta$ going out from a discrete state $x \in X$ such that $\Gamma((x, e, x')) = [99, 100]$. The maximal dwell time of x is $d_{max}(x) = 100$, and the set of regions of x is $R(x) = [0,0], (0,1), [1,1], (1,2), \cdots, [99,99], (99,100), [100,100]$, which includes 101 partitioned regions and leads a large amount of states in the region automaton. This section proposes zone automaton, which is significantly compact compared

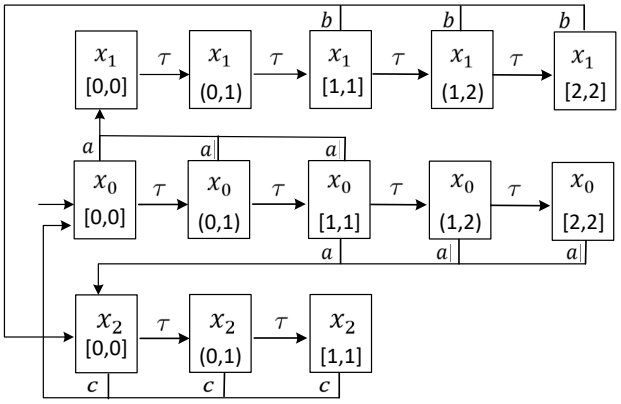


Fig. 3.2 Region automaton of G in Fig. 3.1.

with the region automaton.

In this subsection, we introduce the notion of timed states and propose a method to partition the dwell time at a discrete state into a set of zones depending on the firability of transitions. After that, we provide the definition and approach to construct the zone automaton of a given TFA.

Definition 3.9 Given a TFA G, an *timed state* is defined as a pair (x, θ), where x is a state of G and $\theta \in [0, d_{max}(x)]$ is the current value of the clock. □

In other words[1], a timed state (x, θ) keeps the track of the current clock assignment θ while G dwells at state x.

Definition 3.10 Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, *the set of active transitions at a timed state* $(x, \theta) \in X \times \mathbb{R}_{\geq 0}$ is defined as $\mathcal{A}(x, \theta) = \{(x, e, x') \in \Delta \mid (\exists e \in E)(\exists x' \in X)\ \theta \in \Gamma((x, e, x'))\}$. □

In simple words, the set of active transitions at a timed state (x, θ) includes all the transitions that may fire from x with a clock value θ. Note that clearly it is $\theta \in [0, d_{max}(x)]$. The set of active transitions at (x, θ) may vary for different values of θ in $[0, d_{max}(x)]$. This leads to the definition of clock *zones* associated with a given state $x \in X$.

Definition 3.11 Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, the *set of zones* of $x \in X$ is defined as $Z(x) = \{[0, +\infty)\}$ if $d_{max}(x) = \infty$; otherwise it is defined as a set of time intervals $Z(x) = \{z_0, \cdots z_n\} \subseteq \mathbb{I},\ n \geq 0$, where the following conditions hold:

[1] According to the usual terminology in hybrid systems community, the timed state (x, θ) is the hybrid state of the timed automaton, while x and θ are the discrete and the continuous states of the timed automaton, respectively.

- $z_0 = [0, 0]$;
- $\bigcup_{i=0}^{n} z_i = [0, d_{max}(x)]$;
- $\theta < \theta'$ holds for all $\theta \in z_{i-1}$ and for all $\theta' \in z_i$, where $i \in \{1, \cdots, n\}$;
- $\mathcal{A}(x, \theta) = \mathcal{A}(x, \theta')$ holds for all $\theta, \theta' \in z_i$, where $i \in \{0, \cdots, n\}$;
- $\mathcal{A}(x, \theta) \neq \mathcal{A}(x, \theta')$ holds for all $\theta \in z_{i-1}$ and for all $\theta' \in z_i$, where $i \in \{2, \cdots, n\}$.

In addition, $prec(z_i) = z_{i-1}$ (resp., $succ(z_i) = z_{i+1}$) is said to be the *preceding zone* (resp., *succeeding zone*) of $z_i \in Z(x)$, where $i \in \{1, \cdots, n\}$ (resp., $i \in \{0, \cdots, n-1\}$). □

If there exists no transition originating from x, G stays at x indefinitely: in such a case the set of zones of x is a singleton $\{[0, +\infty)\}$. Otherwise, the set of zones of a state x follows from the partitioning of the dwell time at x into several time intervals to which the clock may belong. The union of all zones in $Z(x)$ covers the interval $[0, d_{max}(x)]$. Any two zones of x are disjoint. If $\theta \in z_i$ and $\theta' \in z_i$, where $i \in \{1, \cdots, n\}$, the sets of active transitions at two timed states (x, θ) and (x, θ') are identical. In addition, the firability of transitions differs between (x, θ) and (x, θ') if $\theta \in z_{i-1}$ and $\theta' \in z_i$, where $i \in \{2, \cdots, n\}$. Particularly, $z_0 = [0, 0]$ is defined to be a zone associated with each state of G, apart from the case of $d_{max}(x) = +\infty$. This originates from the considered time semantics, according to whom the clock is reset whenever G arrives at a state in X. Then the clock evolves discretely from a time instant $\theta \in z_{i-1}$ to another time instant $\theta' \in z_i$, where $i \in \{1, \cdots, n\}$.

Example 3.5 Consider the TFA $G = (X, E, \Delta, \Gamma, X_0)$ in Fig. 3.3. Let us focus on the initial state x_0. There exist two transitions originating from x_0, namely $(x_0, b, x_2) \in \Delta$ with $\Gamma((x_0, b, x_2)) = [0, 1]$ and $(x_0, c, x_1) \in \Delta$ with $\Gamma((x_0, c, x_1)) = [1, 3]$. The maximal dwell time at x_0 is $d_{max}(x_0) = 3$. Consequently, the set of zones of x_0 is $Z(x_0) = \{[0, 0], (0, 1), [1, 1], (1, 3]\}$. The set of active transitions at (x_0, θ), where θ is a time instant in $z \in Z(x_0)$, are reported in Table 3.2. As for the set of zones of other states in X, we have $Z(x_1) = \{[0, 0], (0, 1), [1, 3]\}$, $Z(x_2) = \{[0, 0], (0, 1), [1, 2]\}$, $Z(x_3) = \{[0, 0], (0, 2]\}$ and $Z(x_4) = \{[0, 0], (0, 1]\}$. □

Given a discrete state $x \in X$ and two zones $z, succ(z) \in Z(x)$, we let a new event τ denotes that in a state x the clock value may evolve from any $\theta \in z$ to any $\theta' \in succ(z)$ as time elapses. Note that τ is not a specific value of time but is an abstract representation of the time-driven evolution. We now formalize the definition of zone automaton.

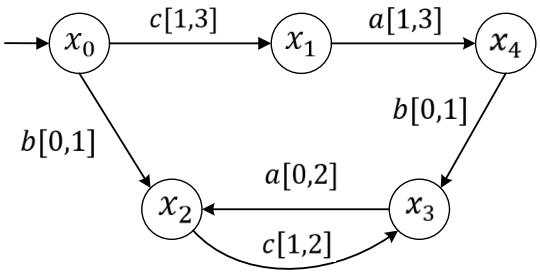


Fig. 3.3 TFA G.

Table 3.2 Sets of active transitions at (x_0, θ) for the TFA G in Fig. 3.3, where $\theta \in z_i$, $i \in \{0, 1, 2, 3\}$.

i	z_i	$\mathcal{A}(x_0, \theta), \theta \in z_i$
0	$[0, 0]$	$\{(x_0, b, x_2)\}$
1	$(0, 1)$	$\{(x_0, b, x_2)\}$
2	$[1, 1]$	$\{(x_0, b, x_2), (x_0, c, x_1)\}$
3	$(1, 3]$	$\{(x_0, c, x_1)\}$

Definition 3.12 Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, the *zone automaton* of G is an NFA $G_z = (V, E_\tau, \Delta_z, V_0)$, where

- $V \subseteq X \times \bigcup_{x \in X} Z(x)$ is the finite set of states,
- $E_\tau \subseteq E \cup \{\tau\}$ is the alphabet,
- $\Delta_z \subseteq V \times E_\tau \times V$ is the transition relation, where the transitions in Δ_z are defined by the following rules:
  - $((x, z), \tau, (x, succ(z))) \in \Delta_z$ if $z, succ(z) \in Z(x)$;
  - $((x, z), e, (x', z_0)) \in \Delta_z$ if $z \in Z(x)$, $z_0 \in Z(x')$, $(x, e, x') \in \mathcal{A}(x, \theta)$ for all $\theta \in z$,
- $V_0 = \{(x, z_0) \mid x \in X_0\} \subseteq V$ is the set of initial states. □

We use the zone automaton to describe the time-driven and event-driven evolution of a TFA $G = (X, E, \Delta, \Gamma, X_0)$. Each state in a zone automaton is a pair (x, z) with $x \in X$ and $z \in Z(x)$. The alphabet is composed of the events in E and event τ. The transition relation specifies the dynamics of the automaton: starting from a state (x, z), a transition $((x, z), \tau, (x, succ(z))) \in \Delta_z$ corresponds to a time-driven evolution of G from a clock value in z to another clock value in $succ(z)$ while G is at x; a transition $((x, z), e, (x', z_0)) \in \Delta_z$

goes from state (x, z) to state (x', z_0), indicating that the occurrence of event e yields state x' when the current state of the system is x and the current clock is in z. The set of initial states is the set of pairs of a state $x \in X_0$ and $z_0 \in Z(x)$.

Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, the zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ can be constructed by Algorithm 3. A temporary set of states V_{new} is introduced, containing all states that still need to be explored in order to compute their output transitions. A *while* loop is repeated until $V_{new} = \emptyset$. A transition $((x, z), \tau, (x, succ(z)))$ is set in Δ_z if $succ(z)$ is a zone at x. For each transition $(x, e, x') \in \Delta$ satisfying $z \subseteq \Gamma((x, e, x'))$, a transition labeled with e is set from $v = (x, z)$. Note that if the maximal dwell time of x' is $+\infty$ (resp., if it is not), the transition labeled with e would lead to state $(x', [0, +\infty))$ (resp., state (x', z_0)). To avoid redundant repetitions of the *while* loop, the state v' is included in V_{new} if v' is neither in V nor in V_{new}. The *while* loop stops once all states in V_{new} have been explored.

Algorithm 3 Construction of the zone automaton associated with a TFA

Input: A TFA $G = (X, E, \Delta, \Gamma, X_0)$
Output: The zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$
1: let $V = \emptyset$, $E_\tau = E \cup \{\tau\}$, $\Delta_z = \emptyset$, $V_0 = \{(x, z_0) \mid x \in X_0\}$, and $V_{new} = V_0$
2: **while** $V_{new} \neq \emptyset$ **do**
3: select a $v = (x, z) \in V_{new}$
4: **if** $succ(z) \in Z(x)$ **then**
5: let $\bar{v} = (x, succ(z))$, $\Delta_z = \Delta_z \cup \{(v, \tau, \bar{v})\}$, and $V_{new} = V_{new} \cup \{\bar{v}\}$
6: **end if**
7: **for** each $(x, e, x') \in \Delta$ **do**
8: **if** $z \subseteq \Gamma((x, e, x'))$ **then**
9: **if** $d_{max}(x') \neq +\infty$ **then**
10: let $v' = (x', z_0)$
11: **else**
12: let $v' = (x', [0, +\infty))$
13: **end if**
14: let $\Delta_z = \Delta_z \cup \{(v, e, v')\}$
15: **if** $v' \notin V \cup V_{new}$ **then**
16: let $V_{new} = V_{new} \cup \{v'\}$
17: **end if**
18: **end if**
19: **end for**
20: let $V = V \cup \{v\}$ and $V_{new} = V_{new} \setminus \{v\}$
21: **end while**
22: **return** $G_z = (V, E_\tau, \Delta_z, V_0)$.

Next we discuss the computational complexity of Algorithm 3. Let $Q_x = \{(x, e, x') \in \Delta \mid e \in E, x' \in X\}$ be the set of output transitions of state $x \in X$. Our approach partitions the dwell interval of a state $x \in X$ into a set of zones $Z(x)$ which depends on set Q_x.

When $Q_x \neq \emptyset$, a simple analysis shows that the maximum number of zones is bounded by $2|Q_x| + 1$. This bound is also correct for $Q_x = \emptyset$ due to $Z(x) = [0, +\infty)$. As a result, by letting $q = \max_{x \in X} |Q_x|$, the maximal number of states of G_z is $|V| \leq (2q + 1)|X|$. In Algorithm 3, the *while* loop at Step 2 is executed $|V|$ times, while the *for* loop at Step 7 is executed at most q times; consequently, the time complexity is $\mathcal{O}(q^2|X|)$.

Example 3.6 Consider the TFA $G = (X, E, \Delta, \Gamma, X_0)$ in Fig. 3.3. The zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ is shown in Fig. 3.4. The initial state is $V_0 = \{(x_0, [0, 0])\}$, implying that G starts from x_0 at clock value 0. A transition labeled with an event τ implies a time-driven evolution of G. For instance, a transition $((x_0, [0, 0]), \tau, (x_0, (0, 1)))$ represents that the clock may evolve from the value in $[0, 0]$ to any value in $(0, 1)$ if G is at x_0. Meanwhile, a transition labeled with an event in E implies an event-driven evolution of G. For instance, a transition labeled with b goes from $(x_0, (0, 1))$ to $(x_2, [0, 0])$. It represents a state evolution from x_0 to x_2 under the occurrence of an event b, upon which the clock is reset.

Compared with the set of regions at x_1, namely $R(x_1) = \{[0, 0], (0, 1), [1, 1], (1, 2), [2, 2], (2, 3), [3, 3]\}$, the zones at x_1 is $Z(x) = [0, 1), [1, 1]$, and $(1, 3]$, which leads the zone automaton to be more compact. □

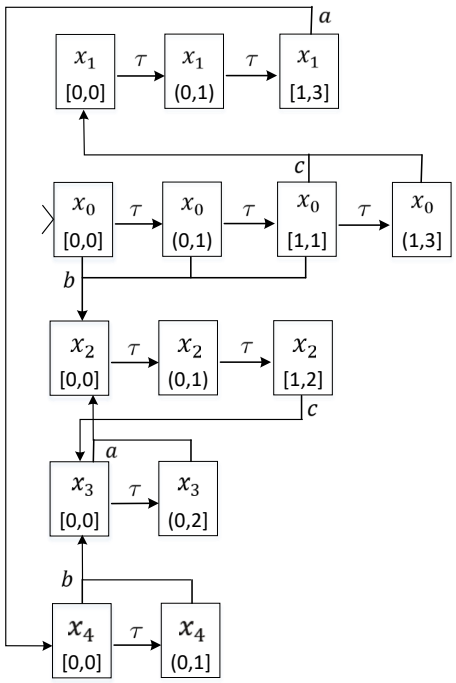


Fig. 3.4 Zone automaton G_z of the TFA G in Fig. 3.3.

3.2.3 Dynamics of Zone Automaton

In this subsection, we explore the dynamics of a zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ associated with a TFA $G = (X, E, \Delta, \Gamma, X_0)$ and discuss how the timed evolutions of G are related to the evolutions of its zone automaton G_z. We first introduce the notion of *τ-run at a state* $x \in X$ that corresponds to the elapsing of time with no event occurrence. Next, a run in G_z is considered, including also the occurrence of discrete events, to show the time-driven evolution and the event-driven evolution of G. We show that the problem of investigating the reachability of a state in G can be reduced to the reachability analysis of a state in G_z.

Definition 3.13 Consider a TFA $G = (X, E, \Delta, \Gamma, X_0)$ and a state $x \in X$ with $Z(x) = \{z_0, \cdots, z_n\}$ $(n \geq 0)$. Let the zone automaton of G be $G_z = (V, E_\tau, \Delta_z, V_0)$. A *$\tau$-run at* x of length k $(0 \leq k \leq n)$ is defined as a sequence of $k+1$ states $(x, z_i) \in V$ $(i = 0, \cdots, k)$, and event $\tau \in E_\tau$, represented as

$$\rho_\tau(x) : (x, z_0) \xrightarrow{\tau} (x, z_1) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (x, z_k)$$

such that $((x, z_{i-1}), \tau, (x, z_i)) \in \Delta_z$ holds for $i \in \{1, \cdots, k\}$. The *starting state* and the *ending state* of $\rho_\tau(x)$ are denoted as $v_{st}(\rho_\tau(x)) = (x, z_0)$ and $v_{en}(\rho_\tau(x)) = (x, z_k)$, respectively. The *duration range* of $\rho_\tau(x)$ is denoted as $d(\rho_\tau(x)) = z_k$. □

A τ-run at x represents a series of state evolutions in G_z involving the states associated with x and event τ. It essentially represents the time elapsing discretely while G is at x. The duration range $d(\rho_\tau(x)) = z_k$ is an interval describing the possible duration of the run. Next we define the notion of a *run* of a zone automaton to represent a hybrid evolution of the associated TFA.

Definition 3.14 Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$ and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, a run of length $k \geq 0$ in G_z is a sequence of $k+1$ τ-runs $\rho_\tau(x_{(i)})$ $(i = 0, \cdots, k)$ at $x_{(i)} \in X$, and k events $e_i \in E$ $(i = 1, \cdots, k)$, represented as

$$\bar{\rho} : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \rho_\tau(x_{(1)}) \cdots \xrightarrow{e_k} \rho_\tau(x_{(k)}),$$

such that $(v_{en}(\rho_\tau(x_{(i-1)})), e_i, v_{st}(\rho_\tau(x_{(i)}))) \in \Delta_z$ holds for $i \in \{1, \cdots, k\}$. In addition, the *starting state* and the *ending state* of $\bar{\rho}$ are defined as $v_{st}(\bar{\rho}) = v_{st}(\rho_\tau(x_{(0)}))$ and $v_{en}(\bar{\rho}) = v_{en}(\rho_\tau(x_{(k)}))$, respectively. The *duration range* of $\bar{\rho}$ is defined as $d(\bar{\rho}) = \bigoplus_{i=1}^{k} d(\rho_\tau(x_i))$. The logical word generated by $\bar{\rho}$ is denoted as $s(\bar{\rho}) = e_1 \cdots e_k$ via a function defined as $s : E_\tau^* \to E^*$. The set of runs generated by G_z is defined as $\mathcal{R}_z(G_z)$. □

The dynamics of a zone automaton G_z can be represented by a run in G_z; in addition, such a run corresponds to an evolution of the TFA G. In simple words, any two consecutive τ-runs $\rho_\tau(x_{(i-1)})$ and $\rho_\tau(x_{(i)})$ are connected by a state evolution caused by an event $e_i \in E$. The logical word of $\bar{\rho}$ is the sequence of events in E that have been involved in $\bar{\rho}$. The duration range of $\bar{\rho}$ is evaluated by summing up the duration ranges of all the involved τ-runs.

Definition 3.15 A state x is said to be *reachable* from state x' within $t \in \mathbb{R}_{\geq 0}$, if there exists a timed evolution $(\sigma(\rho), t) \in (E \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$ of G from 0 to t such that $x_{st}(\rho) = x'$ and $x_{en}(\rho) = x$. In addition, x is said to be *unobservable-reachable* from x' within t if there exists a timed evolution $(\sigma(\rho), t)$ from 0 to t such that $S(\sigma(\rho)) \in E_{uo}^*$. □

Theorem 3.1 *Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$ and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, a state $x \in X$ is reachable from $x' \in X$ within $t \in \mathbb{R}_{\geq 0}$, if and only if there exists a run $\bar{\rho}$ in G_z such that $t \in d(\bar{\rho})$, $v_{st}(\bar{\rho}) = (x', z_0)$, and $v_{en}(\bar{\rho}) = (x, z)$, where $z_0 \in Z(x')$ and $z \in Z(x)$.* □

Proof: (if) Let a run $\bar{\rho} : \rho_\tau(x') \xrightarrow{e'} \cdots \xrightarrow{\bar{e}} \rho_\tau(x)$ and a time instant $\bar{t} \in \mathbb{R}_{\geq 0}$ such that $e', \cdots, \bar{e} \in E$, $t \in d(\bar{\rho})$ and $t - \bar{t} \in d(\rho_\tau(x))$. Accordingly, by denoting a timed run from 0 to $\bar{t}$ as $\rho : x' \xrightarrow{(e',t')} \cdots \xrightarrow{(\bar{e},\bar{t})} x$ such that $t' \in d_{max}(x')$, $t - \bar{t} \in d_{max}(x)$ and $d(\rho) = \bar{t}$, there exists a timed evolution $(\sigma(\rho), t) \in (E \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$ of G from 0 to t, where $x_{st}(\rho) = x'$ and $x_{en}(\rho) = x$. Thus, x is reachable from x' within t.

(only if) Let $(\sigma(\rho), t) \in (E \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$ be a timed evolution of G from 0 to t such that $x_{st}(\rho) = x'$ and $x_{en}(\rho) = x$. The proof is made by induction on the length k of the timed run ρ generated by $G = (X, E, \Delta, \Gamma, X_0)$ from 0 to t. The base case is for the timed run ρ of length 0 that involves only state x' and no transition in G. We have $\sigma(\rho) = \lambda$ and $x_{st}(\rho) = x_{en}(\rho) = x' = x$. There exists a run in G_z as $\bar{\rho} : (x', [0, 0]) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (x', z)$, where $t \in d(\bar{\rho})$. Thus the base case holds.

The induction hypothesis is that the existence of a timed run $\rho : x_{(0)} \xrightarrow{(e_1,t_1)} x_{(1)} \cdots x_{(k-1)} \xrightarrow{(e_k,t_k)} x_{(k)}$ of length $k \geq 1$, where $x_{(i)} \in X$ and $e_i \in E$ for all $i \in \{1, \cdots, k\}$, implies the existence of a run $\bar{\rho} : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \rho_\tau(x_{(1)}) \cdots \rho_\tau(x_{(k-1)}) \xrightarrow{e_k} \rho_\tau(x_{(k)})$ in G_z such that $v_{st}(\bar{\rho}) = (x', z_0)$ and $v_{en}(\bar{\rho}) = (x, z)$, where $z \in Z(x)$. We now prove that the same implication holds for a timed run $\rho' : x_{(0)} \xrightarrow{(e_1,t_1)} x_{(1)} \cdots x_{(k-1)} \xrightarrow{(e_k,t_k)} x_{(k)} \xrightarrow{(e_{k+1},t_{k+1})} x_{(k+1)}$ of length $k+1$. According to ρ', we have $(x_{(k)}, e_{k+1}, x_{j_{k+1}}) \in \Delta$ and $t_{k+1} \in \Gamma((x_{(k)}, e_{k+1}, x_{j_{k+1}}))$; consequently, there exists a run $\bar{\rho}' : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \rho_\tau(x_{(1)}) \cdots \rho_\tau(x_{(k-1)}) \xrightarrow{e_k} \rho_\tau(x_{(k)}) \xrightarrow{e_{k+1}} \rho_\tau(x_{(k+1)})$ in G_z such that $t_{k+1} - t_k \in d(\rho_\tau(x_{(k+1)}))$ and $t - t_{k+1} \in f_z(v_{en}(\rho_\tau(x_{(k+1)})))$, where

the function $f_z : V \rightarrow \bigcup_{x \in X} Z(x)$ maps a state in X_z to a zone. Therefore, we have $v_{st}(\bar{\rho}') = v_{st}(\bar{\rho}) = (x', z_0)$ and $t \in d(\bar{\rho}')$. □

Theorem 3.1 provides a necessary and sufficient condition to determine the reachability of a state x from x' within t. Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$, if a state $x \in X$ is reachable from $x' \in X$ within t, then in its zone automaton there exists a run that originates from (x', z_0) and reaches (x, z), where $z \in Z(x)$. In addition, t belongs to the duration range of that run. In turn, given a run $\bar{\rho}$ in G_z such that $t \in d(\bar{\rho})$, it can be concluded that the state associated with $v_{en}(\bar{\rho})$ is reachable from the state associated with $v_{st}(\bar{\rho})$ within t. In simple words, the reachability of a state x from x' within t can be analyzed by exploring an appropriate run in G_z.

Example 3.7 Consider the TFA $G = (X, E, \Delta, \Gamma, X_0)$ in Fig. 3.3 and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ in Fig. 3.4. There exists a timed evolution $((c, 1.5)(a, 3), 4)$ of G from 0 to 4 such that x_4 is reachable from x_0 within 4. Accordingly, there exists a run $\bar{\rho} : \rho_\tau(x_0) \xrightarrow{c} \rho_\tau(x_1) \xrightarrow{a} \rho_\tau(x_4)$, where $\rho_\tau(x_0) : (x_0, [0, 0]) \xrightarrow{\tau} (x_0, (0, 1)) \xrightarrow{\tau} (x_0, [1, 1]) \xrightarrow{\tau} (x_0, (1, 3])$, $\rho_\tau(x_1) : (x_1, [0, 0]) \xrightarrow{\tau} (x_1, (0, 1)) \xrightarrow{\tau} (x_1, [1, 3])$ and $\rho_\tau(x_4) : (x_4, [0, 0]) \xrightarrow{\tau} (x_4, (0, 1])$. □

3.3 State Estimation of Timed Finite Automata

In this section, we consider partially observed TFA, where some of the event occurrences are observable and others are unobservable. We formally present the definition of a *projection function* as follows before clarifying the state estimation problem.

Definition 3.16 Given a TFA G with $E = E_o \cup E_{uo}$, a *projection function* $P : (E \times \mathbb{R}_{\geq 0})^* \longrightarrow (E_o \times \mathbb{R}_{\geq 0})^*$ is defined as $P(\lambda) = \lambda$, and

$$P(\sigma(\rho) \cdot (e, t)) = \begin{cases} P(\sigma(\rho)) & \text{if } e \in E_{uo} \\ P(\sigma(\rho)) \cdot (e, t) & \text{if } e \in E_o \end{cases}$$

for the timed word $\sigma(\rho) \in (E \times \mathbb{R}_{\geq 0})^*$ generated from any timed run $\rho \in \mathcal{R}(G)$ and for all $(e, t) \in E \times \mathbb{R}_{\geq 0}$. □

In other words, the projection operator P simply erases the pairs consisting of an unobservable event and the time of its occurrence in a timed word. As shown in Fig. 3.5, given a TFA G and a timed run $\rho \in \mathcal{R}(G)$, the projection function P always maps the timed word $\sigma(\rho)$ to an observed word $\sigma_o \in (E_o \times \mathbb{R}_{\geq 0})^*$. The pair $(\sigma_o, t) = (P(\sigma(\rho)), t)$ is the *timed observation* related to $(\sigma(\rho), t)$.

Definition 3.17 Given a TFA G with $E = E_o \cup E_{uo}$, and a *timed observation* (σ_o, t), $\mathcal{S}(\sigma_o, t) = \{(\sigma(\rho), t) \in \mathcal{E}(G, t) | P(\sigma(\rho)) = \sigma_o\}$ is said to be the *set of timed evolutions consistent with* (σ_o, t), i.e., the set of timed evolutions that can be generated by G from 0 to t producing the timed observation (σ_o, t); meanwhile $\mathcal{X}(\sigma_o, t) = \{x_{en}(\rho) \in X | (\sigma(\rho), t) \in \mathcal{S}(\sigma_o, t)\}$ is said to be *the set of states consistent with* (σ_o, t), i.e., the set of states in which G may be, after (σ_o, t) is observed. □

This section aims at calculating the set $\mathcal{X}(\sigma_o, t)$, which includes the states reached by the timed evolution $(\sigma(\rho), t)$ consistent with (σ_o, t). Looking again at Fig. 3.5, the set $\mathcal{X}(\sigma_o, t)$ is indeed the output of the state estimation process.

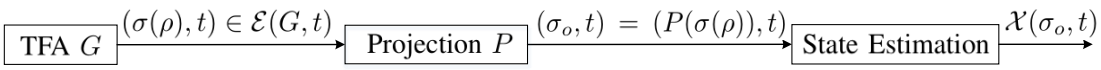


Fig. 3.5 Sketch of the state estimation problem.

Note that, in general, two different sub-problems must be solved in this setting: (a) Update the current estimate as time elapses without any new observation being received; (b) Update the current estimate when a new observation is received, based on the observed event label and the occurrence time. Accordingly, we partition this section into two subsections. In the first subsection, we consider the case where G produces no observation, and prove that we can estimate the set of possible current states as time elapses without any observation looking at appropriate runs in the associated zone automaton G_z. This is an intermediate step towards the solution of the state estimation problem under partial observation. In the second subsection, we take into account the information coming from the observation of new events at certain time instants, and prove that the set of states consistent with a timed observation (σ_o, t) can be inferred following a certain number of runs in the zone automaton G_z. In more detail, the main steps to do that in a systematic way are summarized in an algorithm.

3.3.1 State Estimation with No Observation

Definition 3.18 Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$ with the set of unobservable events E_{uo} and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, the following set of states of G_z

$$V_\lambda(x', t) = \{v_{en}(\bar{\rho}) \in V \mid \begin{array}{l} \bar{\rho} \in \mathcal{R}_\tau(G_z), v_{st}(\bar{\rho}) = (x', z_0), \\ s(\bar{\rho}) \in E_{uo}^*, t \in d(\bar{\rho})\} \end{array}$$

is said to be λ-*estimation from* $x' \in X$ *within* $t \in \mathbb{R}_{\geq 0}$. □

Given a zone automaton, the λ-estimation from x' within $t \in \mathbb{R}_{\geq 0}$ is the set of states of the zone automaton G_z that can be reached following a run of duration t, originating at (x', z_0) and producing no observation. If (x, z) belongs to the λ-estimation from x' within t, then the zone z associated with x specifies how long the TFA G could be in x. As a special case, if $V_\lambda(x', t) = \emptyset$, it means that G cannot stay in state x' from time 0 to t producing no observation.

Given a TFA G with a set of states X, a set of unobservable events E_{uo} and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, Algorithm 4 computes the λ-estimation from a state $x \in X$ within $t \in \mathbb{R}_{\geq 0}$. It first initializes a set R as the singleton $\{(x, [0, 0])\}$. The *while* loop selects an element $r = (\bar{x}, \bar{I})$ in R at each iteration. For all zones of state $\bar{x}$, we check if there exists a transition $((\bar{x}, z), e, (x', [0, 0])) \in \Delta_z$ with $e \in E_{uo}$. If so, we add to R the pair $(x', z \bigoplus \bar{I})$. Note that x' is the state reached by unobservable transitions and $z \bigoplus \bar{I}$ is the range of time it takes to reach x'. In addition, if t belongs to $z \bigoplus \bar{I}$, it can be inferred that $\bar{x}$ can be reached from x within t; consequently, the set V_λ is updated by including $(\bar{x}, z)$. At the end of each *while* loop, the explored $r \in R$ is deleted from R. Finally, the algorithm returns $V_\lambda(x, t) = V_\lambda$.

Next we analysis the computational complexity of Algorithm 4. The *while* loop at Step 2 is executed at most $|V|$ times; the *for* loop at Step 4 is executed at most $2q + 1$ times, and *for* loop at Step 5 is executed at most q times. The complexity of Algorithm 4 is thus $\mathcal{O}(q(2q + 1)|V|) = \mathcal{O}(q^3|X|)$.

Algorithm 4 Computation of the λ-estimation from a state of a TFA within a time instant

Input: A TFA G with a set of states X and a set of unobservable events E_{uo}, a zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, a state $x \in X$, and a time instant $t \in \mathbb{R}_{\geq 0}$
Output: λ-estimation $V_\lambda(x, t)$

1: let $R = \{(x, [0, 0])\}$ and $V_\lambda = \emptyset$
2: **while** $R \neq \emptyset$ **do**
3: select $r = (\bar{x}, \bar{I}) \in R$
4: **for** each $z \in Z(\bar{x})$ **do**
5: **for** each $((\bar{x}, z), e, (x', [0, 0])) \in \Delta_z$ with $e \in E_{uo}$ **do**
6: let $R = R \cup \{(x', z \bigoplus \bar{I})\}$
7: **end for**
8: **if** $t \in z \bigoplus \bar{I}$ **then**
9: let $V_\lambda = V_\lambda \cup \{(\bar{x}, z)\}$
10: **end if**
11: **end for**
12: let $R = R \setminus \{r\}$
13: **end while**
14: **return** $V_\lambda(x, t) = V_\lambda$.

Theorem 3.2 *Given a TFA $G = (X, E, \Delta, \Gamma, X_0)$ with set of unobservable events E_{uo} and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, state $x \in X$ is unobservable-reachable from $x' \in X$ within $t \in \mathbb{R}_{\geq 0}$ if and only if there exists $(x, z) \in V_\lambda(x', t)$, where $z \in Z(x)$.* □

Proof: (if) Let us suppose that there exists $(x, z) \in V_\lambda(x', t)$, where $z \in Z(x)$. Then there exists a run $\bar{\rho}$ in G_z such that $v_{st}(\bar{\rho}) = (x', z_0)$, $s(\bar{\rho}) \in E_{uo}^*$ and $t \in d(\bar{\rho})$. This implies that x is unobservable-reachable from x' within t.

(only if) Let x be unobservable-reachable from x' within t. Then, there exists a timed evolution $(\sigma(\rho), t)$ from 0 to t such that $x_{st}(\rho) = x'$, $x_{en}(\rho) = x$ and $s(\bar{\rho}) \in E_{uo}^*$. Accordingly, there exists a run $\bar{\rho}$ in G_z such that $v_{st}(\bar{\rho}) = (x', z_0)$, $f_x(v_{en}(\bar{\rho})) = x$, $s(\bar{\rho}) \in E_{uo}^*$ and $t \in d(\bar{\rho})$, where $f_x : V \to X$ maps a state in V to a state in X. Thus, there exists $(x, z) \in V_\lambda(x', t)$, where $z \in Z(x)$. □

Given a state of the TFA, Theorem 3.2 provides a criterion to estimate the set of states in which the zone automaton can be when no observation is received within a given time instant $t \in \mathbb{R}_{\geq 0}$. In other words, the λ-estimation from x' within t reveals the unobservable-reachable states from x' within t, i.e., the set of states of G when it starts its evolution from state x', and produces no observation from 0 to the time instant t.

Note that the state estimation under no observation can be seen as a full-fledged problem by itself: consider, as an example the case of a system where the state can be directly measured but only with a (possibly asynchronous) sampling period T. When T is large, between two consecutive measurements it may be necessary to estimate the current state in absence of any observation and this estimate may actually be used to optimally choose the next sampling instant. A related problem, in a decentralized setting, was studied in [104] where the asynchronous polling of distributed sub-systems was called *synchronization*.

Example 3.8 Consider the TFA $G = (X, E, \Delta, \Gamma, X_0)$ in Fig. 3.3 with $E_o = \{a\}$, $E_{uo} = \{b, c\}$, and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ in Fig. 3.4. We have $V_\lambda(x_0, 1) = \{(x_0, [1, 1]), (x_1, [0, 0]), (x_2, [0, 0]), (x_2, (0, 1)), (x_2, [1, 2]), (x_3, [0, 0])\}$. This results from the exhaustive enumeration of all the runs in G_z of duration 1 starting from $(x_0, [0, 0])$ and involving no observable event:

1. $\bar{\rho}_1 : \rho_\tau(x_0)$, where $\rho_\tau(x_0) : (x_0, [0, 0]) \xrightarrow{\tau} (x_0, (0, 1)) \xrightarrow{\tau} (x_0, [1, 1])$;

2. $\bar{\rho}_2 : \rho_\tau(x_0) \xrightarrow{c} \rho_\tau(x_1)$, where $\rho_\tau(x_0) : (x_0, [0, 0]) \xrightarrow{\tau} (x_0, (0, 1)) \xrightarrow{\tau} (x_0, [1, 1])$ and $\rho_\tau(x_1) : (x_1, [0, 0])$;

3. $\bar{\rho}_3 : \rho_\tau(x_0) \xrightarrow{b} \rho_\tau(x_2)$, where $\rho_\tau(x_0) : (x_0, [0,0]) \xrightarrow{\tau} (x_0, (0,1)) \xrightarrow{\tau} (x_0, [1,1])$ and $\rho_\tau(x_2) : (x_2, [0,0])$;

4. $\bar{\rho}_4 : \rho_\tau(x_0) \xrightarrow{b} \rho_\tau(x_2)$, where $\rho_\tau(x_0) : (x_0, [0,0])) \xrightarrow{\tau} (x_0, (0,1))$ and $\rho_\tau(x_2) : (x_2, [0,0]) \xrightarrow{\tau} (x_2, (0,1))$;

5. $\bar{\rho}_5 : \rho_\tau(x_0) \xrightarrow{b} \rho_\tau(x_2)$, where $\rho_\tau(x_0) : (x_0, [0,0])$ and $\rho_\tau(x_2) : (x_2, [0,0]) \xrightarrow{\tau} (x_2, (0,1)) \xrightarrow{\tau} (x_2, [1,2])$;

6. $\bar{\rho}_6 : \rho_\tau(x_0) \xrightarrow{b} \rho_\tau(x_2) \xrightarrow{c} \rho_\tau(x_3)$, where $\rho_\tau(x_0) : (x_0, [0,0])$, $\rho_\tau(x_2) : (x_2, [0,0]) \xrightarrow{\tau} (x_2, (0,1)) \xrightarrow{\tau} (x_2, [1,2])$ and $\rho_\tau(x_3) : (x_3, [0,0])$.

Now, let us focus on $(x_2, [1,2]) \in V_\lambda(x_0, 1)$. That is to say, if G starts its evolution from x_0 at time 0 and produces no observation during the whole time interval $[0,1]$, G can be in state x_2 at time 1 with clock value in the zone $[1,2]$. Table 3.3 summarizes the λ-estimation and the set of states consistent with (λ, t), where t belongs to a time interval in $\{[0,0], (0,1), [1,1], (1,2), [2,2], (2,3), [3,3], (3,4), [4,4], (4,5), [5,5], (5,6), [6,6], (6,7)\}$. Note that $V_\lambda(x_0, t) = \emptyset$ and $\mathcal{X}(\lambda, t) = \emptyset$ for $t \in (6,7)$. In other words, it is not possible that G enters state x_0 at time 0 and no observation occurs for more than 6 time units. □

Table 3.3 State estimation of the TFA G in Fig. 3.3 under no observation.

k	Time interval I_k	λ-estimation $V_\lambda(x_0, t)$, where $t \in I_k$	$\mathcal{X}(\lambda, t), t \in I_k$
0	[0,0]	$(x_0, [0,0]), (x_2, [0,0])$	$\{x_0, x_2\}$
1	(0,1)	$(x_0, (0,1)), (x_2, [0,0]), (x_2, (0,1))$	$\{x_0, x_2\}$
2	[1,1]	$(x_0, [1,1]), (x_1, [0,0]), (x_2, [0,0]),$ $(x_2, (0,1)), (x_2, [1,2]), (x_3, [0,0])$	$\{x_0, x_1, x_2, x_3\}$
3	(1,2)	$(x_0, (1,3]), (x_1, [0,0]), (x_1, (0,1)), (x_2, (0,1)),$ $(x_2, [1,2]), (x_3, [0,0]), (x_3, (0,2])$	$\{x_0, x_1, x_2, x_3\}$
4	[2,2]	$(x_0, (1,3]), (x_1, [0,0]), (x_1, (0,1)), (x_1, [1,3)),$ $(x_2, [1,2]), (x_3, [0,0]), (x_3, (0,2])$	$\{x_0, x_1, x_2, x_3\}$
5	(2,3)	$(x_0, (1,3]), (x_1, [0,0]), (x_1, (0,1)), (x_1, [1,3]),$ $(x_2, [1,2]), (x_3, [0,0]), (x_3, (0,2])$	$\{x_0, x_1, x_2, x_3\}$
6	[3,3]	$(x_0, (1,3]), (x_1, [0,0]), (x_1, (0,1)), (x_1, [1,3]),$ $(x_2, [1,2]), (x_3, [0,0]), (x_3, (0,2])$	$\{x_0, x_1, x_2, x_3\}$
7	(3,4)	$(x_1, (0,1)), (x_1, [1,3]), (x_3, (0,2])$	$\{x_1, x_3\}$
8	[4,4]	$(x_1, [1,3]), (x_3, (0,2])$	$\{x_1, x_3\}$
9	(4,5)	$(x_1, [1,3]), (x_3, (0,2])$	$\{x_1, x_3\}$
10	[5,5]	$(x_1, [1,3]), (x_3, (0,2])$	$\{x_1, x_3\}$
11	(5,6)	$(x_1, [1,3])$	$\{x_1\}$
12	[6,6]	$(x_1, [1,3])$	$\{x_1\}$
13	(6,7)	$\emptyset$	$\emptyset$

3.3.2 State Estimation with Partial Observation

In this subsection we focus on the state estimation problem when a timed observation is received as a pair of a non-empty timed word and a time instant. We first propose a result as follows.

Theorem 3.3 *Consider a TFA $G = (X, E, \Delta, \Gamma, X_0)$ with the set of observable events E_o and zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$. Given a timed observation $(\sigma_o, t) \in (E_o \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$, a state $x \in X$ is consistent with (σ_o, t) if and only if there exists a run $\bar{\rho}$ in G_z such that $f_x(v_{st}(\bar{\rho})) \in X_0$, $f_x(v_{en}(\bar{\rho})) = x$, $t \in d(\bar{\rho})$ and $P_l(s(\bar{\rho})) = S(\sigma_o)$, where $f_x : V \to X$ and $P_l : E^* \to E_o^*$.* □

Proof: (if) In the case that no observation is contained in σ_o, let $\bar{\rho} : \rho_\tau(x_0)$ be a run in G_z, where $x_0 \in X_0$. We have $f_x(v_{en}(\bar{\rho})) = x_0$, $t \in d(\bar{\rho})$ and $\sigma_o = \lambda$. In this case there exists only one timed evolution $(\lambda, t) \in \mathcal{E}(G, t)$ such that $x_0 \in \mathcal{X}(\lambda, t)$. In the case that there exist one or more event occurrences, let $\bar{\rho} : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \cdots \xrightarrow{e_k} \rho_\tau(x_{(k)})$ be a run in G_z, where $x_{(i)} \in X$ and $i \in \{0, \cdots, k\}$, such that $e_i \in E$, $d(\rho_\tau(x_{(i-1)})) \subseteq \Gamma((x_{(i-1)}, e_i, x_{(i)}))$ and $d(\rho_\tau(x_{(i)})) \subseteq [0, d_{max}(x_{(i)})]$ for $i \in \{1, \cdots, k\}$. Let us suppose that $x_{(0)} \in X_0$ and $x_{(k)} = x$. It can be inferred that $d(\bar{\rho}) \subseteq \bigoplus_{i=1}^{k} \Gamma((x_{(i-1)}, e_i, x_{(i)})) \bigoplus [0, d_{max}(x_{(k)})]$ and $t - t_k \in d(\rho_\tau(x_{(k)}))$ hold. For each $i \in \{1, \cdots, k\}$, let $t_i \in d(\rho_\tau(x_{(i-1)}))$ be a time instant. Then, there exists a timed run defined in G as $\rho : x_{(0)} \xrightarrow{(e_1,t_1)} \cdots \xrightarrow{(e_k,t_k)} x_{(k)}$. Consequently, there is a timed evolution $(\sigma(\rho), t) \in \mathcal{E}(G, t)$, where $\sigma(\rho) = (e_1, t_1) \cdots (e_k, t_k)$. According to $P_l(\bar{s}(\bar{\rho})) = S(\sigma_o)$, it is $(\sigma(\rho), t) \in \mathcal{S}(\sigma_o, t)$. Obviously, $x \in \mathcal{X}(\sigma_o, t)$ holds.

(only if) In the case that $\sigma_o = \lambda$, $x \in \mathcal{X}(\lambda, t)$ holds. There exists a run $\bar{\rho}$ in G_z such that x is unobservable-reachable from a state $x_0 \in X_0$ within t. Consequently, $f_x(v_{en}(\bar{\rho})) = x$, $t \in d(\bar{\rho})$ and $\bar{s}(\bar{\rho}) = \varepsilon$ hold. In the case that $\sigma_o = (e_{o1}, t_1) \cdots (e_{ok}, t_k)$, where $k \geq 1$, $0 \leq t_1 < \cdots < t_k \leq t$ and $e_{oi} \in E_o$ ($i \in \{1, \cdots, k\}$), it is $x \in \mathcal{X}(\sigma_o, t)$, implying that x is reachable from a state $x_0 \in X_0$ within t. Consequently, there exists a run $\bar{\rho}$ in G_z such that $t \in d(\bar{\rho})$, $f_x(v_{st}(\bar{\rho})) = x_0$ and $f_x(v_{en}(\bar{\rho})) = x$. Let $\bar{\rho}$ be a sequence of k runs $\bar{\rho}_i$ and k events e_{oi}, where $i \in \{1, \cdots, k\}$, as $\bar{\rho} : \bar{\rho}_0 \xrightarrow{e_{o1}} \cdots \xrightarrow{e_{ok}} \bar{\rho}_k$. Obviously, $\bar{s}(\bar{\rho}) = e_{o1} \cdots e_{on}$, $t - t_k \in d(\bar{\rho}_k)$ and $t_i - t_{i-1} \in d(\bar{\rho}_{i-1})$ hold for each $i \in \{1, \cdots, k\}$. □

Consider a timed observation (σ_o, t) produced by a TFA $G = (X, E, \Delta, \Gamma, X_0)$; a state $x \in X$ is consistent with (σ_o, t) when there exists a run $\bar{\rho}$ in G_z that produces the same logical observation as σ_o from a state in X_0 at 0 and reaches x at t. In turn, $x \in \mathcal{X}(\sigma_o, t)$ can be concluded according to the run $\bar{\rho}$ in G_z. In more detail, the run $\bar{\rho}$ contains no observable event if $\sigma_o = \lambda$. This clearly implies that $x \in \mathcal{X}(\lambda, t)$. If $\sigma_o \neq \lambda$, the run $\bar{\rho}$ is a sequence

of $k+1$ runs $\bar{\rho}_i,\ i \in \{0, \cdots, k\}$, and k observable events $e_{oi} \in E_o,\ i \in \{1, \cdots, k\}$, as $\bar{\rho} : \bar{\rho}_0 \xrightarrow{e_{o1}} \cdots \xrightarrow{e_{ok}} \bar{\rho}_k$. For each $i \in \{0, \cdots, k\}$, we have $s(\bar{\rho}_i) = \varepsilon$, implying that $\bar{\rho}_i$ produces no observation. For each $i \in \{1, \cdots, k\}$, an observable event $e_{oi} \in E_o$ leads the state evolution from $v_{en}(\bar{\rho}_{i-1})$ to $v_{st}(\bar{\rho}_i)$.

Algorithm 5 summarizes the approach that we propose, based on the previous results, to compute the set of states consistent with a timed observation (σ_o, t). It can be explained as follows. Consider a timed observation (σ_o, t) with $\sigma_o = (e_{o1}, t_1) \cdots (e_{on}, t_n)$ $(n \geq 1)$, where $e_{o1}, \cdots, e_{on} \in E_o$. The timed observation (σ_o, t) is updated whenever an observable event e_{oi} occurs at a time instant t_i, where $i \in \{1, \cdots, n\}$. The algorithm provides a set of estimated states while time elapses in $[t_{i-1}, t_i]$ with no event being observed, in addition to a set of states $\bar{X}_i \subseteq X$ consistent with each new observation (e_{oi}, t_i), where $i \in \{1, \cdots, n\}$ and $t_0 = 0$. Initially, it is imposed $\bar{X}_0 = X_0$ and $\bar{X}_i = \emptyset$ for all $i \in \{1, \cdots, n\}$. Then, for each $i \in \{1, \cdots, n\}$, the algorithm computes the λ-estimation from each state $x \in \bar{X}_{i-1}$ within $t_i - t_{i-1}$ implying the states unobservable-reachable from $\bar{X}_{i-1}$ within $t_i - t_{i-1}$, and the set $\bar{X}_i$ is updated with the states reached by transitions labeled with e_{oi} from the estimated states satisfying their timing functions. After the set $\bar{X}_n$ is determined, we compute the set of states unobservable-reachable from $\bar{X}_n$ within $t - t_n$, and return it as the set of states consistent with (σ_o, t).

In other words, during the online phase to estimate the current discrete state, one just has to determine which is the state of the observer reached by the current observation and check to which interval (among a finite number of time intervals) the time elapsed between the last observed event occurrence belongs. This approach allows one to construct offline an observer, i.e., a finite structure that describes the state estimation for all possible evolutions. We believe that this approach has a major advantage over existing online approaches for state estimations: it paves the way to address a vast range of fundamental properties (detectability, opacity, etc.) that have so far mostly been studied in the context of logical DES.

Given a timed observation $\sigma_o = (e_{o1}, t_1) \cdots (e_{on}, t_n)$ $(n \geq 1)$ generated by G, the complexity of Algorithm 5 depends on the number n of pairs (event, time instant at which the event occurs). For each pair (e_{oi}, t_i), two *for* loops are executed.

- The first *for* loop at Step 4 is executed at most $|X|$ times, calling Algorithm 4 whose complexity is $\mathcal{O}(q^3|X|)$. Thus the complexity of this loop is $\mathcal{O}(q^3|X|^2)$.

- The second *for* loop at Step 8 is executed at most $|V|$ times; hence its complexity is $\mathcal{O}(q|X|)$.

Finally, the *for* loop at Step 15, analogously to the *for* loop at Step 4, has complexity $\mathcal{O}(q^3|X|^2)$. Overall, the complexity of Algorithm 5 is $\mathcal{O}(n(q^3|X|^2 + q|X|) + q^3|X|^2) = \mathcal{O}(nq^3|X|^2)$.

Algorithm 5 State estimation of a TFA

Input: A TFA G with a set of initial states X_0, a set of observable events $E_o \subseteq E$, a zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, and a timed observation (σ_o, t) from 0 to $t \in \mathbb{R}_{\geq 0}$, where $\sigma_o = (e_{o1}, t_1) \cdots (e_{on}, t_n)$ $(n \geq 1)$ and $t_1, \cdots, t_n \in \mathbb{R}_{\geq 0}$

Output: The set of states $\mathcal{X}(\sigma_o, t)$

1: let $\bar{X}_0 = X_0$, $t_0 = 0$ and $X_\lambda = \emptyset$
2: **for** each $i \in \{1, \cdots, n\}$ **do**
3: let $e = e_{oi}$, $\bar{X}_i = \emptyset$ and $V_\lambda = \emptyset$
4: **for** each $\bar{x} \in \bar{X}_{i-1}$ **do**
5: compute $V_\lambda(\bar{x}, t_i - t_{i-1})$
6: let $V_\lambda = V_\lambda \cup V_\lambda(\bar{x}, t_i - t_{i-1})$
7: **end for**
8: **for** each $v \in V_\lambda$ **do**
9: let $x = f_x(v)$ and $z = f_z(v)$
10: **if** $\exists x' \in X$ s.t. $z \subseteq \Gamma((x, e, x'))$ **then**
11: let $\bar{X}_i = \bar{X}_i \cup \{x'\}$
12: **end if**
13: **end for**
14: **end for**
15: **for** each $\bar{x} \in \bar{X}_n$ **do**
16: compute $V_\lambda(\bar{x}, t - t_n)$ and let

$$X_\lambda = X_\lambda \cup \{x \in X \mid (\exists z \in Z(x))(x, z) \in V_\lambda(\bar{x}, t - t_n)\};$$

17: **end for**
18: **return** $\mathcal{X}(\sigma_o, t) = X_\lambda$.

Example 3.9 Consider the partially observed TFA $G = (X, E, \Delta, \Gamma, X_0)$ in Fig. 3.3 with $E_o = \{a\}$, $E_{uo} = \{b, c\}$ and a timed observation $(\sigma_o, 4)$, where $\sigma_o = (a, 1)(a, 3.5)$. It implies that the observable event a has been measured twice at $t_1 = 1$ and $t_2 = 3.5$, respectively, while the current time instant is $t = 4$.

Table 3.4 summarizes the state estimation of the TFA G in Fig. 3.3 given the timed observation $((a, 1)(a, 3.5), 4)$ according to Algorithm 5. It additionally shows how the state estimation is updated while time elapses in the time interval $[0, 4]$ taking into account the two observations of event a. In particular, the state estimation process is progressively updated as follows. We explain the process of state estimation while the observation (σ_o, t) is progressively updated over time as follows.

- Step $k = 1$: The TFA G produces observation $(\sigma_o, t) = (\lambda, t)$ for $t \in [0, 1]$. We compute the set of unobservable-reachable states from $\bar{X}_0$ within 1 as $\{x_0, x_1, x_2, x_3\}$, according to $\bigcup_{x \in \bar{X}_0} V_\lambda(x, 1) = \{(x_0, [1, 1]), (x_1, [0, 0]), (x_2, [0, 0]), (x_2, (0, 1)), (x_2, [1, 1]), (x_3, [0, 0])\}$. After the pair $(a, 1)$ is received, Algorithm 5 provides the set of states reached by a at $t_1 = 1$, denoted as $\bar{X}_1 = \{x_2\}$, according to $(x_3, [0, 0]) \in \bigcup_{x \in \bar{X}_0} V_\lambda(x, 1)$ and $[0, 0] \in \Gamma((x_3, a, x_2))$.

- Step $k = 2$: The TFA G produces observation $(\sigma_o, t) = ((a, 1), t)$ for $t \in [1, 3.5]$. We compute the set of unobservable-reachable states from $\bar{X}_1$ within 2.5 as $\{x_3\}$, according to $\bigcup_{x \in \bar{X}_1} V_\lambda(x, 2.5) = \{(x_3, (0, 2])\}$. After the pair $(a, 3.5)$ is received, Algorithm 5 provides the set of states reached by a at $t_2 = 3.5$, denoted as $\bar{X}_2 = \{x_2\}$, according to $(x_3, (0, 2]) \in \bigcup_{x \in \bar{X}_1} V_\lambda(x, 2.5)$ and $(0, 2] \in \Gamma((x_3, a, x_2))$.

- Step $k = 3$: The TFA G produces observation $(\sigma_o, t) = ((a, 1)(a, 3.5), t), t \in [3.5, 4]$. We compute the set of unobservable-reachable states from $\bar{X}_2$ within 0.5 as $\{x_2\}$, according to $\bigcup_{x \in \bar{X}_2} V_\lambda(x, 0.5) = \{(x_2, (0, 1))\}$. No more pair is received and the observation ends with $(\sigma_o, t) = ((a, 1)(a, 3.5), 4)$. Algorithm 5 provides the set of states consistent with $(\sigma_o, t) = ((a, 1)(a, 3.5), 4)$ that is equal to $\mathcal{X}(\sigma_o, t) = \{x_2\}$. □

Table 3.4 State estimation of the TFA G in Fig. 3.3 with $\bar{X}_0 = X_0$, $t_0 = 0$ and $(\sigma_o, t), t \in [0, 4]$.

k	σ_o	Time interval I ($t \in I$)	$V_\lambda = \bigcup_{x \in \bar{X}_{k-1}} V_\lambda(x, t - t_{k-1}), t \in I$	$\mathcal{X}(\sigma_o, t)$	$\bar{X}_k$
1	λ	[0,0]	$(x_0, [0, 0]), (x_2, [0, 0])$	$\{x_0, x_2\}$	$\{x_2\}$
		(0,1)	$(x_0, (0, 1)), (x_2, [0, 0]), (x_2, (0, 1))$	$\{x_0, x_2\}$	
		[1,1]	$(x_0, [1, 1]), (x_1, [0, 0]), (x_2, [0, 0]),$ $(x_2, (0, 1)), (x_2, [1, 2]), (x_3, [0, 0])$	$\{x_0, x_1, x_2, x_3\}$	
2	$(a, 1)$	[1,1]	$(x_2, [0, 0])$	$\{x_2\}$	$\{x_2\}$
		(1,2)	$(x_2, (0, 1))$	$\{x_2\}$	
		[2,2]	$(x_2, [1, 2]), (x_3, [0, 0])$	$\{x_2, x_3\}$	
		(2,3)	$(x_2, [1, 2]), (x_3, [0, 0]), (x_3, (0, 2])$	$\{x_2, x_3\}$	
		[3,3]	$(x_2, [1, 2]), (x_3, [0, 0]), (x_3, (0, 2])$	$\{x_2, x_3\}$	
		(3,4)	$(x_3, (0, 2])$	$\{x_3\}$	
3	$(a, 1)(a, 3.5)$	(3,4)	$(x_2, [0, 0])$	$\{x_2\}$	-
		[4,4]	$(x_2, (0, 1))$	$\{x_2\}$	

3.4 Fault Diagnosis of Timed Finite Automata

In this section, we assume that the timed system may be affected by a set of faults described by timed transitions whose occurrence changes the state of the plant and resets the clock. Two types of fault transitions are considered in this paper: observable fault transitions labeled with a symbol in E_o, and unobservable fault transitions labeled with a symbol in E_{uo}. The set of transitions modeling a regular behaviour is denoted as Δ_{reg}, while the set of transitions modeling a fault behaviour is denoted as Δ_{fault}. Clearly, it is $\Delta = \Delta_{reg} \cup \Delta_{fault}$.

We define a *diagnosis function* for a set of fault transitions Δ_{fault} as $\phi : (E_o \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0} \rightarrow \{F, N, U\}$ associated to each timed observation (σ_o, t) a diagnosis state $\phi((\sigma_o, t))$, where $\phi((\sigma_o, t)) = F$ (resp., $\phi((\sigma_o, t)) = N$) denotes that a fault transition in Δ_{fault} has (resp., not) been executed while producing (σ_o, t), and $\phi((\sigma_o, t)) = U$ denotes that a fault transition may or may not have been executed.

In this section, we utilize the proposed state estimation approach and deal with diagnosing a fault behaviour based on a timed observation (σ_o, t), namely computing $\phi((\sigma_o, t))$. Note that we are not distinguishing among different fault transitions. According to the notation used in most of the literature on fault diagnosis of discrete event systems ([30], [29]), this means that we assume that all faults belong to the same class.

This section is divided into two subsections: the first subsection constructs a fault recognizer and investigates the dynamics of a timed DES with faults, and the second subsection deals with the diagnosis problem of TFA in analyzing the reachability of the fault recognizer.

3.4.1 Fault Recognizer

In this subsection, we construct *a fault recognizer* that recognizes the occurrence of faults. We first transform the model G of the plant with faults into *a canonical plant* G_f with faults. For the canonical plant G_f, the zone automaton $ZA(G_f)$ is constructed. The particular structure of the canonical plant allows us to construct *a fault recognizer* by synchronizing $ZA(G_f)$ with a *fault monitor* that recognizes the occurrence of a fault denoted by a symbol f.

Definition 3.19 Consider a partially observed TFA $G = (X, E, \Delta, \Gamma, X_0)$ with $E = E_o \cup E_{uo}$. The canonical plant is modeled as a TFA $G_f = (X \cup X_f, E \cup \{f\}, \Delta_f, \Gamma_f, X_0)$, where f is an additional unobservable event used to model the occurrence of a fault transition. The set of additional states X_f, the transition relation Δ_f, and the timing function Γ_f are defined according to each $\delta = (x, e, x'') \in \Delta$ as follows:

- if $\delta \in \Delta_{fault}$ and $e \in E_{uo}$, we define $\delta_f = (x, f, x'') \in \Delta_f$ and $\Gamma_f(\delta_f) = \Gamma(\delta)$;
- if $\delta \in \Delta_{fault}$ and $e \in E_o$, we define $\{\delta_1, \delta_2\} \subseteq \Delta_f$, $\Gamma_f(\delta_1) = \Gamma(\delta)$, and $\Gamma_f(\delta_2) = [0, 0]$, where $\delta_1 = (x, f, x')$, $\delta_2 = (x', e, x'')$, and $x' \in X_f$;
- if $\delta \in \Delta_{reg}$, we define $\delta \in \Delta_f$ and $\Gamma_f(\delta) = \Gamma(\delta)$. □

In the canonical plant G_f, the new fault event f is introduced: this will allow construct the fault recognizer by synchronization with the fault monitor. Given a transition $\delta = (x, e, x'') \in \Delta$, G can generate a timed run from initial time 0 ending with $x \xrightarrow{(e,t)} x''$, where $t \in \Gamma(\delta)$. If e is associated with an observable fault transition, G_f can generate a timed run ending with $x \xrightarrow{(f,t)} x' \xrightarrow{(e,t)} x''$, implying that δ is replaced by $\delta_1 = (x, f, x')$ satisfying $\Gamma_f(\delta_1) = \Gamma(\delta)$ and following by $\delta_2 = (x', e, x'')$ that occurs immediately, i.e., $\Gamma_f(\delta_2) = [0, 0]$. In other words, considering an observable fault transition, G_f keeps track of both the occurrence of the fault f and the observation of e.

On the contrary, it is not necessary to keep track of the occurrence of the unobservable event. If e is associated with an unobservable fault transition, a new unobservable symbol f is labeled with the transition $\delta = (x, f, x'') \in \Delta_f$ of G_f. If $\delta \in \Delta_{reg}$, we let $\delta \in \Delta_f$ and $\Gamma_f(\delta) = \Gamma(\delta)$. Note that in G_f, the set of unobservable events is extended to $E_{uo} \cup \{f\}$.

Example 3.10 Consider the TFA G in Fig. 3.6 with $E_o = \{a, b, c\}$ and $\Delta_{fault} = \{(x_0, c, x_2), (x_1, d, x_3)\}$. A canonical plant $G_f = (X \cup \{x_f\}, E \cup \{f\}, \Delta_f, \Gamma_f, X_0)$ is depicted in Fig. 3.7, where fault transitions are shown in red. The transition $(x_1, d, x_3) \in \Delta_{fault}$ with an unobservable fault d is replaced by $(x_1, f, x_3) \in \Delta_f$ in G_f, and $(x_0, c, x_2) \in \Delta_{fault}$ with an observable fault c is replaced by two consecutive transitions in G_f, namely (x_0, f, x_f) and (x_f, c, x_2) satisfying that $\Gamma((x_0, f, x_f)) = [1, 2]$ and $\Gamma((x_f, c, x_2)) = [0, 0]$.

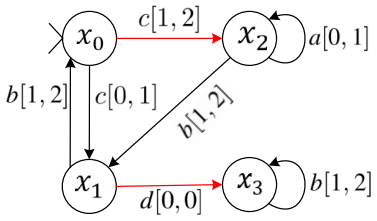


Fig. 3.6 TFA G, where fault transitions are shown in red.

Next we consider the zones of each discrete state of G_f. For x_0, from which there exist two transitions originating, namely $(x_0, c, x_2) \in \Delta$ with $\Gamma((x_0, c, x_2)) = [1, 2]$ and $(x_0, c, x_1) \in \Delta$ with $\Gamma((x_0, c, x_1)) = [1, 2]$, the maximal dwell time at x_0 is $d_{max}(x_0) = 2$.

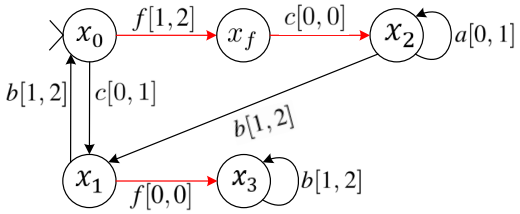


Fig. 3.7 Canonical plant G_f associated with the TFA G in Fig. 3.6 as described in Example 3.

Consequently, the set of zones of x_0 is $Z(x_0) = \{[0,0],(0,1),[1,1],(1,2]\}$. The set of active transitions at (x_0,θ), where θ is a time instant in $z \in Z(x_0)$, are reported in Table 3.5. As for the set of zones of other states of G_f, we have $Z(x_1) = \{[0,0],(0,1),[1,2]\}$, $Z(x_2) = \{[0,0],(0,1),[1,1],(1,2]\}$, $Z(x_3) = \{[0,0],(0,1),[1,2]\}$ and $Z(x_f) = \{[0,0]\}$.

Table 3.5 Sets of active transitions at (x_0,θ) for the TFA G in Fig. 3.6, where $\theta \in z_i$, $i \in \{0,1,2,3\}$.

i	z_i	$\mathcal{A}(x_0,\theta), \theta \in z_i$
0	$[0,0]$	$\{(x_0,c,x_1)\}$
1	$(0,1)$	$\{(x_0,c,x_1)\}$
2	$[1,1]$	$\{(x_0,c,x_1),(x_0,b,x_2)\}$
3	$(1,2]$	$\{(x_0,b,x_2)\}$

The zone automaton $ZA(G_f) = (V, E_\tau, \Delta_z, V_0)$ is shown in Fig. 3.8. The initial state is $(x_0,[0,0])$, implying that G starts from x_0 at clock value 0. A transition labeled with an event τ implies a time-driven evolution of G. For instance, a transition $((x_0,[0,0]),\tau,(x_0,(0,1)))$ represents that the clock may evolve from the value in $[0,0]$ to any value in $(0,1)$ if G is at x_0. Meanwhile, a transition labeled with an event in $E \cup \{f\}$ implies an event-driven evolution of G. For instance, a transition labeled with c goes from $(x_0,[1,1])$ to $(x_2,[0,0])$. It represents a state evolution from x_0 to x_2 under the occurrence of an event c, upon which the clock is reset. □

We now introduce a deterministic untimed automaton called *fault monitor* and denote it as $M = (\{N,F\}, \{f\}, \{(N,f,F), (F,f,F)\}, N)$ shown in Fig. 3.9, where state N (resp., F) denotes that no fault (resp., a fault) has occurred, and the state always evolves from N and F to F upon each occurrence of f. To deal with fault diagnosis, we construct *a fault recognizer* $Rec(G_f)$ by composing $ZA(G_f)$ with M by parallel composition such that labels N and F are attached to states of $Rec(G_f)$ to recognize the occurrence of faults.

Definition 3.20 Consider a timed DES with faults modeled by a TFA $G_f = (X \cup X_f, E \cup \{f\}, \Delta_f, \Gamma_f, X_0)$. Given zone automaton $ZA(G_f) = (V, E \cup \{f,\tau\}, \Delta_z, v_0)$ and a fault

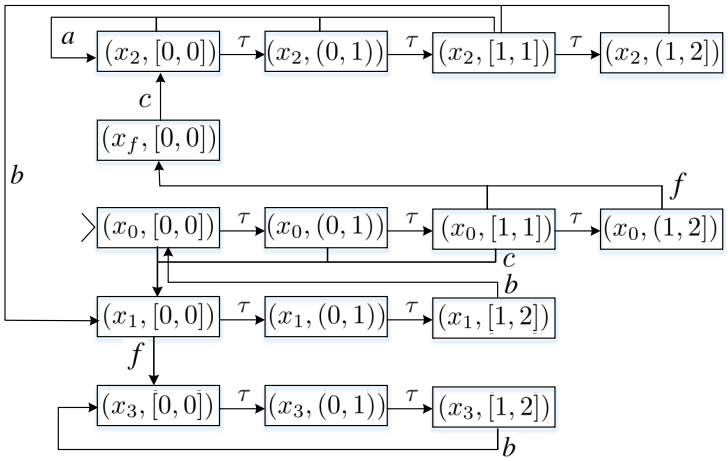


Fig. 3.8 Zone automaton $ZA(G_f)$ of G_f in Fig. 3.7.

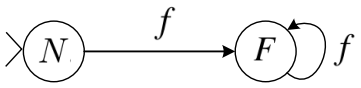


Fig. 3.9 Fault monitor M for diagnosing event f.

monitor $M = (\{N, F\}, \{f\}, \{(N, f, F), (F, f, F)\}, N)$, *the fault recognizer* is the automaton $Rec(G_f) = (X_{rec}, E_{rec}, \Delta_{rec}, X_{rec0})$, where $X_{rec} \subseteq V \times \{N, F\}$, $E_{rec} = E \cup \{f, \tau\}$, $X_{rec0} = V_0 \times \{N\}$, and a transition $\delta_{rec} \in \Delta_{rec}$ satisfies the following conditions:

- if $e = f$, $\{((v, N), f, (v', F)), ((v, F), f, (v', F))\} \subseteq \Delta_{rec}$ holds for each $(v, f, v') \in \Delta_z$;
- if $e \in E \cup \{\tau\}$, $((v, N), e, (v', N)) \in \Delta_{rec}$ holds for each $(v, e, v') \in \Delta_z$. □

Example 3.11 For instance, the fault recognizer $Rec(G_f)$ is depicted in Fig. 3.10, where G_f is shown in Fig. 3.7. □

3.4.2 Fault Diagnosis Approach

In this subsection, we deal with fault diagnosis of timed DES with faults modeled by a TFA $G_f = (X \cup X_f, E \cup \{f\}, \Delta_f, \Gamma_f, X_0)$. We first study the dynamics of its fault recognizer $Rec(G_f) = (X_{rec}, E_{rec}, \Delta_{rec}, X_{rec0})$ via the following definitions.

Definition 3.21 A *τ-run at $x \in X$* is defined as a sequence of $k+1$ states in $(x, z_i, s_{fault}) \in X_{rec}$ $(0 \leq i \leq k)$ and the event τ, represented as $\rho_\tau(x) : (x, z_0, s_{fault}) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (x, z_k, s_{fault})$, such that $((x, z_{i-1}, s_{fault}), \tau, (x, z_i, s_{fault})) \in \Delta_{rec}$ holds for $i \in \{1, \cdots, k\}$. We denote the *starting state* (resp., the *ending state*) of $\rho_\tau(x)$ as $q_{st}(\rho_\tau(x)) = (x, z_0, s_{fault})$

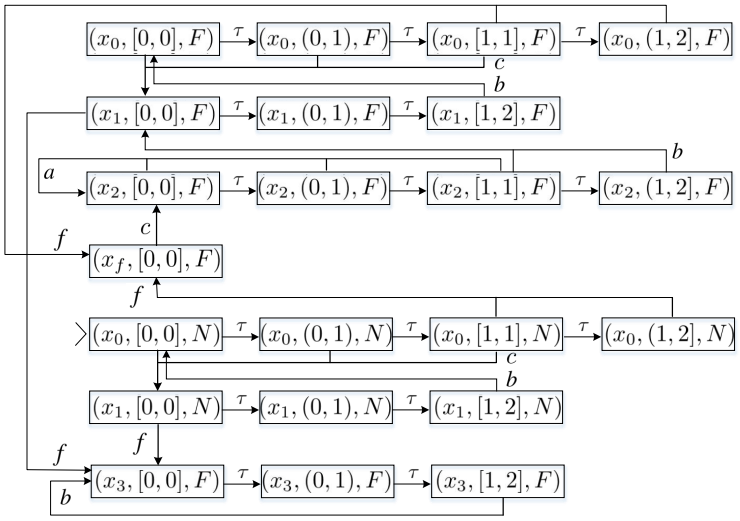


Fig. 3.10 Fault recognizer $Rec(G_f)$ of G_f in Fig. 3.7.

(resp., $q_{en}(\rho_\tau(x)) = (x, z_k, s_{fault})$). The *duration range* of $\rho_\tau(x)$ is denoted as $d(\rho_\tau(x)) = z_k$. The *fault label* of $\rho_\tau(x)$ is denoted as $f_{label}(\rho_\tau(x)) = s_{fault}$. □

In other words, a τ-run at x essentially represents the time elapsing discretely while G_f is at $x \in X$. The zone $z_i \in Z(x)$ $(0 \leq i \leq k)$ (resp., $s_{fault} \in \{N, F\}$) provides the range of the possible clock values (resp., the fault label) associated with x.

Definition 3.22 A *run in* $Rec(G_f) = (X_{rec}, E_{rec}, \Delta_{rec}, X_{rec0})$ of length k is defined as a sequence of τ-runs $\rho_\tau(x_{(i)})$ $(i \in \{0, \cdots, k\})$ at $x_{(i)} \in X$, and k events $e_i \in E$ $(i \in \{1, \cdots, k\})$, represented as

$$\bar{\rho} : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \rho_\tau(x_{(1)}) \cdots \xrightarrow{e_k} \rho_\tau(x_{(k)}),$$

such that $(q_{en}(\rho_\tau(x_{(i-1)})), e_i, q_{st}(\rho_\tau(x_{(i)}))) \in \Delta_{rec}$ holds for $i \in \{1, \cdots, k\}$. In addition, it is $f_{label}(\rho_\tau(x_{(j)})) = F$ if $e_i = f$ for $i \leq j \leq k$.

We denote the *starting state* (resp., the *ending state*) of $\bar{\rho}$ as $q_{st}(\bar{\rho}) = q_{st}(\rho_\tau(x_{(0)}))$ (resp., $q_{en}(\bar{\rho}) = q_{en}(\rho_\tau(x_{(k)}))$). The *fault label* of $\bar{\rho}$ is denoted as $f_{label}(\bar{\rho}) = f_{label}(\rho_\tau(x_k))$. The *duration range* of $\bar{\rho}$ is denoted as $d(\bar{\rho}) = \bigoplus_{i=0}^{k} d(\rho_\tau(x_{(i)}))$. The logical word generated by $\bar{\rho}$ is denoted as $s(\bar{\rho}) = e_1 \cdots e_k$ via a function defined as $s : E_\tau^* \to E^*$. The set of runs generated by G_z is defined as $\mathcal{R}(Rec(G_f))$. □

The τ-runs involving an elapsed time τ with no executed events essentially represent the time elapsing discretely. A run in $Rec(G_f)$ represents the evolutions of G_f that involve

both time elapsing and events occurrence. After an event $e_i, i \in \{1, \cdots, k\}$ is executed, the state of G_f evolves from $x_{(i-1)}$ to $x_{(i)}$. The fault label is $\bar{\rho} = F$ once a fault has occurred.

The logical word of $\bar{\rho}$ is the sequence of events in $E \cup \{f\}$ that have been involved in $\bar{\rho}$. The duration range of $\bar{\rho}$ is evaluated by summing up the duration of each τ-run at $x_{(i)}, i \in \{0, \cdots, k\}$.

Next we focus on the fault diagnosis problem when a timed observation is received as a pair of a non-empty timed word and a time instant. We propose and prove the following theorem as follows.

Theorem 3.4 Consider a TFA $G = (X, E, \Delta, \Gamma, X_0)$ with a set of fault transitions, its canonical plant $G_f = (X \cup X_f, E \cup \{f\}, \Delta_f, \Gamma_f, X_0)$, and its fault recognizer $Rec(G_f) = (X_{rec}, E_{rec}, \Delta_{rec}, X_{rec0})$. Given a timed observation $(\sigma_o, t) \in (E_o \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$, where $\sigma_o = (e_{o1}, t_1) \cdots (e_{on}, t_n), n \geq 1$, and $0 = t_0 \leq t_1 \leq \cdots \leq t_n \leq t$, then there exists a timed run $\bar{\rho} \in \mathcal{R}(Rec(G_f))$, defined as $\bar{\rho} : \bar{\rho}_{(0)} \xrightarrow{e_{o1}} \bar{\rho}_{(1)} \cdots \xrightarrow{e_{on}} \bar{\rho}_{(n)}$, such that the following conditions are satisfied:

(a) $t \in d(\bar{\rho})$, $t - t_n \in d(\bar{\rho}_{(n)})$ and $t_i - t_{i-1} \in d(\bar{\rho}_{(i-1)})$ for $i \in \{1, \cdots, n\}$;

(b) $P_l(s(\bar{\rho})) = e_{o1} \cdots e_{on}$, $P_l(s(\bar{\rho}_{(i)})) = \varepsilon$ for $i \in \{1, \cdots, n\}$, where $P_l : (E \cup \{f\})^* \to E_o^*$;

(c) $f_{label}(\bar{\rho}) = N$ if $|s(\bar{\rho})|_f = 0$; else, $f_{label}(\bar{\rho}) = F$. □

Proof: Given a timed observation $(\sigma_o, t) \in (E_o \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$, there exists a timed run of G defined as $\rho : \rho_0 \xrightarrow{(e_{o1}, t_1)} \cdots \xrightarrow{(e_{on}, t_n)} \rho_n$, such that $S(\sigma(\rho_i)) = \varepsilon$ for $i \in \{0, \cdots, n\}$, $(x_{en}(\rho_{i-1}), e_i, x_{st}(\rho_i)) \in \Delta$, $T(\rho_{i-1}) = t_i - t_{i-1}$ for $i \in \{1, \cdots, n\}$, and $T(\rho_n) = t - t_n$.

If a fault transition labeled with an observable event has been executed, there exists an associated timed run of G_f defined as $\rho_f : \rho_0 \xrightarrow{(e_{o1}, t_1)} \cdots \xrightarrow{(e_{oi-1}, t_{i-1})} \rho_{i-1} \xrightarrow{(f, t_i)} x_f \xrightarrow{(e_{oi}, t_i)} \cdots \xrightarrow{(e_{on}, t_n)} \rho_n$. Based on that, there exists a run of $Rec(G_f)$ defined as $\bar{\rho} : \bar{\rho}_0 \xrightarrow{e_{o1}} \cdots \xrightarrow{e_{oi-1}} \bar{\rho}_{i-1} \xrightarrow{f} (x_f, [0, 0], F) \xrightarrow{e_{oi}} \bar{\rho}_i \cdots \xrightarrow{e_{on}} \bar{\rho}_n$ such that $(q_{en}(\bar{\rho}_{p-1}), e_{op}, q_{st}(\bar{\rho}_p)) \in \Delta_{rec}$ and $t_p - t_{p-1} \in d(\bar{\rho}_{p-1})$ hold for $1 \leq p \leq n$. Conditions (a), (b), and (c) can be inferred accordingly.

If a fault transition labeled with an unobservable event has been executed, we have a timed run of G_f defined as $\rho_f : \rho_0 \xrightarrow{(e_{o1}, t_1)} \cdots \xrightarrow{(e_{oi}, t_i)} \rho_i \cdots \xrightarrow{(e_{on}, t_n)} \rho_n$, where $\rho_i : x_{(i0)} \xrightarrow{(e_{i1}, t_{i1})} \cdots \xrightarrow{(f, t_{ij})} x_{(ij)} \xrightarrow{(e_{ij+1}, t_{ij+1})} x_{(ij+1)} \cdots \xrightarrow{(e_{im}, t_{im})} x_{(im)} (m \geq 1, 0 \leq i \leq n)$ and $e_{ij} = f (1 \leq j \leq m)$. Accordingly, there exists an associated run of $Rec(G_f)$ defined as $\bar{\rho} : \bar{\rho}_0 \xrightarrow{e_{o1}}$

$\cdots \xrightarrow{e_{oi}} \bar{\rho}_i \cdots \xrightarrow{e_{on}} \bar{\rho}_n$, where $\bar{\rho}_i : \rho_\tau(x_{(i0)}) \xrightarrow{e_{i1}} \cdots \xrightarrow{f} \rho_\tau(x_{(ij)}) \cdots \xrightarrow{e_{im}} \rho_\tau(x_{(im)})$. Thus, conditions (a), (b), and (c) can be inferred. □

In other words, taking into account the information coming from the observation of new events at certain time instants, the occurrence of faults can be analysed by exploring all the runs in $Rec(G_f)$ consistent with the given observation. The fault label $f_{label}(\bar{\rho})$ associated with $\bar{\rho}$ denoted whether the run contains a fault ($f_{label}(\bar{\rho}) = F$) or not ($f_{label}(\bar{\rho}) = N$). By denoting the set of runs consistent with (σ_o, t) as $\mathcal{R}(Rec(G_f), (\sigma_o, t))$, an approach for fault diagnosis can be generated by the following rules:

- $\phi((\sigma_o, t)) = N$ (resp., $\phi((\sigma_o, t)) = F$) if $f_{label}(\bar{\rho}) = N$ (resp., $f_{label}(\bar{\rho}) = F$) holds for each $\bar{\rho} \in \mathcal{R}(Rec(G_f), (\sigma_o, t))$;

- otherwise, it is $\phi((\sigma_o, t)) = U$.

In simple words, the fault diagnosis of the faulty behaviour f can be done via exploring all the runs in $Rec(G_f)$. If the fault label of each run $\bar{\rho} \in \mathcal{R}(Rec(G_f), (\sigma_o, t))$ is $f_{label}(\bar{\rho}) = F$ (resp., $f_{label}(\bar{\rho}) = N$), we may conclude that f has (resp., has not) been occurred for sure; otherwise, f may or may not have been occurred.

Example 3.12 Consider the TFA $G = (X, E, \Delta, \Gamma, X_0)$ in Fig. 3.6 with $E_o = \{a, b, c\}$ and $\Delta_{fault} = \{(x_0, c, x_2),\ (x_1, d,\ x_3)\}$. Given a timed observation $(\sigma_o, 4)$, where $\sigma_o = (c, 1)(b, 2)(c, 3.5)$, the diagnosis procedures from $t = 0$ to $t = 4$ is summarized in Table 3.6. We explain the process of diagnosis while the observation (σ_o, t) is progressively updated over time as follows.

- The TFA G produces observation $(\sigma_o, t) = (\lambda, t)$ for $t \in [0, 1]$. The union of ending states of all runs in $\mathcal{R}(Rec(\lambda, 1))$ is $\{(x_0, [1, 1], N), (x_f, [0, 0], F)\}$ at $t \in [1, 1]$. Thus it is $\phi(\lambda, 1) = U$.

- The TFA G produces observation $(\sigma_o, t) = ((c, 1), t)$ for $t \in [1, 2]$. The union of ending states of all runs in $\mathcal{R}(Rec((c, 1), 2))$ is $\{(x_1, [1, 2], N), (x_2, [1, 1], F), (x_3, [1, 2], F)\}$ at $t \in (2, 3)$. Thus it is $\phi(((c, 1), 2)) = U$.

- The TFA G produces observation $(\sigma_o, t) = ((c, 1)(b, 2), t)$ for $t \in [2, 3.5]$. The union of ending state of all runs in $\mathcal{R}(Rec((c, 1)(b, 2), 3.5))$ is $\{(x_0, (1, 2], N), (x_f, [0, 0], F), (x_1, [1, 2], F), (x_3, [1, 2], F)\}$ at $t \in (3, 4)$. Thus it is $\phi(((c, 1)(b, 2), 3.5)) = U$.

- The TFA G produces observation $(\sigma_o, t) = ((c,1)(b,2)\ (c,3.5),\ t)$ for $t \in [3.5, 4]$. The union of ending state of all runs in $\mathcal{R}(Rec(G_f), (\sigma_o, t))$ is $\{(x_2, (0,1), F)\}$ at $t \in [4,4]$. Thus it is $\phi(((c,1)(b,2)(c,3.5), 4)) = F$. □

Table 3.6 Diagnosis of the TFA G in Fig. 3.6 with $E_f = \{c, d\}$ and (σ_o, t), $t \in [0, 4]$.

σ_o	$I(t \in I)$	$\bigcup_{\bar{\rho} \in \mathcal{R}(Rec(G_f),(\sigma_o,t))} q_{en}(\bar{\rho})$	$\phi((\sigma_o, t))$
λ	[0,0]	$\{(x_0, [0,0], N)\}$	N
	(0,1)	$\{(x_0, (0,1), N)\}$	N
	[1,1]	$\{(x_0, [1,1], N), (x_f, [0,0], F)\}$	U
$(c,1)$	[1,1]	$\{(x_1, [0,0], N), (x_2, [0,0], F), (x_3, [0,0], F)\}$	U
	(1,2)	$\{(x_1, (0,1), N), (x_2, (0,1), F), (x_3, (0,1), F)\}$	U
	[2,2]	$\{(x_1, [1,2], N), (x_2, [1,1], F), (x_3, [1,2], F)\}$	U
$(c,1)(b,2)$	[2,2]	$\{(x_0, [0,0], N), (x_1, [0,0], F), (x_3, [0,0], F)\}$	U
	(2,3)	$\{(x_0, (0,1), N), (x_1, (0,1), F), (x_3, (0,1), F)\}$	U
	[3,3]	$\{(x_0, [1,1], N), (x_f, [0,0], F), (x_1, [1,2], F), (x_3, [1,2], F)\}$	U
	(3,4)	$\{(x_0, (1,2], N), (x_f, [0,0], F), (x_1, [1,2], F), (x_3, [1,2], F)\}$	U
$(c,1)(b,2)(c,3.5)$	(3,4)	$\{(x_2, [0,0], F), (x_2, (0,1), F)\}$	F
	[4,4]	$\{(x_2, (0,1), F)\}$	F

3.5 Conclusions

In this section, we summarize the conclusions of this chapter as follows:

- First, we present a model of timed DES endowed with a single clock that is reset upon each event occurrence. A time interval is associated with each transition specifying at which clock values it may occur. We consider a type of time semantics that imposes constraints on the dwell time spent at each state of a TFA. A timed word generated by a TFA is defined as a sequence of pairs (event, time instant at which the event occurs). Assuming that certain events are unobservable, namely their occurrences are silent, we deal with the problem of estimating the current state of the system in function of measured timed observations.

- The proposed solution is based on partitioning the clock values at a state into zones. We construct a zone automaton that provides a purely discrete description of the considered TFA. We show that the problem of investigating the reachability of a discrete state of a TFA can be reduced to the reachability analysis of a state in its zone automaton. This result provides the basis for the observer design.

- We deal with the state estimation problem considering two sub-problems: state estimation with no observation and state estimation with partial observation. For the case where a TFA produces no observation, we introduce the notion of λ-estimation from x within t as the set of states of the zone automaton that can be reached following a run of duration t, originating at $(x, [0, 0])$ and producing no observation. An algorithm for calculating the λ-estimation is provided. We prove that the set of possible current states as time elapses without any observation can be obtained by calculating the λ-estimation, i.e., looking at appropriate runs in the associated zone automaton. For the case that a timed observation is received as a pair of a non-empty timed word and a time instant, we prove that the set of states consistent with a timed observation can be inferred following a certain number of runs in the zone automaton. An algorithm summarizes our proposed approach to compute the set of states consistent with a timed observation. According to the approach, during the online phase to estimate the current discrete state, one can determine which is the state of the observer reached by the current observation and check to which interval (among a finite number of time intervals) the time elapsed since the last observed event occurrence belongs.

- As an application of the state estimation approach, we assume that TFA may be affected by a set of faults described by timed transitions and aim at diagnosing a fault behaviour based on a timed observation. We first transform the timed plant with faults into a canonical plant and then construct the zone automaton of the canonical plant. By synchronising the zone automaton with a fault monitor that detects the occurrence of a fault, a fault recognizer that detects the occurrence of a fault can be constructed. We conclude that the occurrence of faults can be analyzed by exploring all runs in the fault recognizer that are consistent with a given timed observation.

Chapter 4 State Estimation of Generalized Timed Finite Automata

In this chapter, we introduce a model of timed DES, called *generalized timed finite automata (GTFA)*, characterized by a single clock. Recall that in a TFA, the clock is reset to zero after each event occurrence, and the time semantics constrain the dwell time at each discrete state. In a GTFA, a clock resetting function associated with each transition specifies how the clock value is updated upon its occurrence; consequently, the time semantics of a GTFA allows a system to remain in a discrete state forever. The behaviour of a GTFA can be described via its timed runs.

This chapter considers partially observed GTFA that produces timed observations as a succession of pairs of an observable event and the time instant at which the event has occurred. Our objective is to estimate the current discrete state of the automaton as a function of the current observation and the current time. When dealing with the state estimation problem, we assume that observable transitions should be reset to zero in order to guarantee the applicability of the proposed state estimation approach, which is based on the notion of zones. The solution is based on determining T*-reachability*, which takes into account the discrete states that can be reached with an evolution producing a given observation and a duration equal to T. The problem of T-reachability in the GTFA is reduced to the reachability analysis of the associated zone automaton.

This chapter is divided into five sections. The first section introduces the model of the GTFA and related notions used throughout the chapter. The second section proposes the *zone automaton* of a given GTFA and investigates the dynamics of the zone automaton. We provide a necessary and sufficient condition for the T-reachability of a discrete state in a GTFA and explain the correlation between the dynamics of a GTFA and that of its zone automaton. In the third section, we deal with the state estimation problem by considering first the case under no observation and then the case under partial observation. In the fourth section, we present a discussion on the state estimation of multi-clock timed DES, which can be explored by extending the state estimation approach of GTFA with a single clock. The fifth section concludes this chapter.

4.1 Generalized Timed Finite Automata

This section provides preliminary notions used throughout this chapter.

Definition 4.1 A *generalized timed finite automaton* (GTFA) is a six-tuple $G = (X, E, \Delta, \Gamma, Reset, X_0)$ that operates under a single clock, where

- X is a finite set of discrete states,
- E is an alphabet,
- $\Delta \subseteq X \times E \times X$ is a transition relation,
- $\Gamma : \Delta \to \mathbb{I}_c$ is a timing function,
- $Reset : \Delta \to \mathbb{I}_c \cup \{id\}$ is a clock resetting function such that for $\delta \in \Delta$, the clock is reset to be a value in a time interval $I \in \mathbb{I}_c$ ($Reset(\delta) = I$), or the clock is not reset ($Reset(\delta) = id$), and
- $X_0 \subseteq X$ is the set of initial discrete states. □

For the sake of simplicity, we assume that the clock is set to be 0 initially in this paper. In other words, a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ is an NFA $G = (X, E, \Delta, X_0)$ endowed with a time structure represented by the timing function Γ, and the clock resetting function $Reset$. The transition relation, the timing function, and the clock resetting function specify the dynamics of the GTFA. In more detail, given two discrete states $x, x' \in X$ and an event $e \in E$, $(x, e, x') \in \Delta$ denotes that the occurrence of event e leads to x' when the GTFA is at x. The timing function Γ maps the transition (x, e, x') to a closed time interval in $\mathbb{I}_c$, which specifies a range of clock values at which the event e may occur. The clock resetting function $Reset$ associates with (x, e, x') a closed time interval in $\mathbb{I}_c$ denoting the range of the clock values that are reset to be or renders that the clock is not reset upon the occurrence of the transition by associating id with (x, e, x').

A GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ can be represented by a graph, where a discrete state $x \in X$ corresponds to a node, and each initial discrete state in X_0 is marked by an input arrow. For any transition $(x, e, x') \in \Delta$, there exists a directed edge from x to x' labeled with the symbol e. The information given by the timing function Γ and the clock resetting function $Reset$ can be presented graphically. Given an edge denoting a transition $\delta \in \Delta$, the label $\theta \in \Gamma(\delta)?$ on the edge specifies if δ is enabled with respect to θ; the label $\theta :\in Reset(\delta)$ (resp., $\theta := id$) on the edge specifies to which range θ belongs (resp., specifies that the clock is not reset) after the transition is fired.

Example 4.1 Consider the GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ with $X = \{x_0, x_1, x_2, x_3, x_4\}$, $E = \{a, b, c\}$, $\Delta = \{(x_0, c, x_1), (x_0, b, x_2), (x_1, a, x_4), (x_2, c, x_3), (x_3, a, x_2),$

$(x_4, b, x_3)\}$ and $X_0 = \{x_0\}$. Let the timing function Γ and the clock resetting function $Reset$ be defined as in Table 4.1. The graphical representation of G is visualized in Fig. 4.1. □

Table 4.1 Timing function and clock resetting function of the GTFA G in Fig. 4.1.

$\delta \in \Delta$	$\Gamma(\delta)$	$Reset(\delta)$
(x_0, c, x_1)	$[1, 3]$	$[1, 1]$
(x_0, b, x_2)	$[0, 1]$	id
(x_1, a, x_4)	$[1, 3]$	$[0, 1]$
(x_2, c, x_3)	$[1, 2]$	id
(x_3, a, x_2)	$[0, 2]$	$[0, 0]$
(x_4, b, x_3)	$[0, 1]$	$[0, 0]$

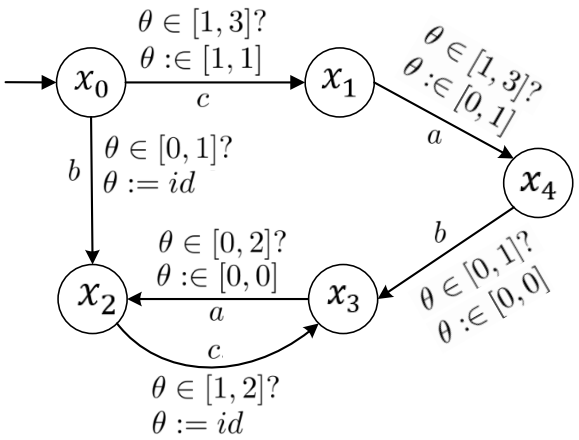


Fig. 4.1 GTFA G.

Definition 4.2 Given $G = (X, E, \Delta, \Gamma, Reset, X_0)$, a *timed (or hybrid) state* is defined as a pair $(x, \theta) \in X \times \mathbb{R}_{\geq 0}$, where θ is the current value of the clock. In other words, a timed state (x, θ) keeps the track of the current clock assignment θ while G stays at state x. □

Definition 4.3 A *timed run* ρ of length $k \geq 0$ from $t_0 \in \mathbb{R}_{\geq 0}$ to $t_k \in \mathbb{R}_{\geq 0}$ is a sequence of $k+1$ timed states $(x_{(i)}, \theta_{(i)}) \in X \times \mathbb{R}_{\geq 0}$ $(i = 0, \cdots, k)$, and k pairs $(e_i, t_i) \in E \times \mathbb{R}_{\geq 0}$ $(i = 1, \cdots, k)$, represented as

$$\rho : (x_{(0)}, \theta_{(0)}) \xrightarrow{(e_1,t_1)} \cdots (x_{(k-1)}, \theta_{(k-1)}) \xrightarrow{(e_k,t_k)} (x_{(k)}, \theta_{(k)})$$

such that the following conditions are satisfied for all $i = 1, \cdots, k$:

$$(x_{(i-1)}, e_i, x_{(i)}) \in \Delta, \tag{4-1}$$

$$\theta_{(i)} \in Reset((x_{(i-1)}, e_i, x_{(i)})), \tag{4-2}$$

$$\theta_{(i-1)} + t_i - t_{i-1} \in \Gamma((x_{(i-1)}, e_i, x_{(i)})). \tag{4-3}$$

We define the *timed word generated by* ρ as $\sigma(\rho) = (e_1, t_1)(e_2, t_2)\cdots(e_k, t_k) \in (E \times \mathbb{R}_{\geq 0})^*$. We also define the *logical word generated by* ρ as $S(\sigma(\rho)) = e_1 e_2 \cdots e_k$ via a function defined as $S : (E \times \mathbb{R}_{\geq 0})^* \to E^*$. Given a timed run ρ of length 0 that only contains the starting discrete state $x_{(0)}$ with the starting clock $\theta_{(0)}$ and no transition, the logical word and the timed word generated by ρ are denoted respectively as $S(\sigma(\rho)) = \varepsilon$ and $\sigma(\rho) = \lambda$, where λ denotes the *empty timed word* in $E \times \mathbb{R}_{\geq 0}$. For the timed word $\sigma(\rho)$ generated from an arbitrary timed run ρ, it is $\lambda \cdot \sigma(\rho) = \sigma(\rho) = \sigma(\rho) \cdot \lambda$. The *starting discrete state* and the *ending discrete state* of a timed run ρ are denoted by $x_{st}(\rho) = x_{(0)}$ and $x_{en}(\rho) = x_{(k)}$, respectively. The *starting time* and the *ending time* of ρ are denoted by $t_{st}(\rho) = t_0$ and $t_{en}(\rho) = t_k$, respectively. In addition, the *duration of* ρ is denoted as $T(\rho) = t_k - t_0$. The set of timed runs generated by G is denoted as $\mathcal{R}(G)$. □

Example 4.2 Given the GTFA in Fig. 4.1, $\rho : (x_0, 0) \xrightarrow{(c,1.5)} (x_1, 1) \xrightarrow{(a,3)} (x_4, 0.6)$ is a timed run of length 3 from time 0 to 3. The timed word $\sigma(\rho) = (c, 1.5)(a, 3)$ corresponds to events c and a occurring at time instants $t_1 = 1.5$ and $t_2 = 3$, respectively. It starts from $x_{st}(\rho) = x_0$ at the starting time 0 and terminates in $x_{en}(\rho) = x_4$ at the ending time 3. The logical word generated by ρ is $S(\sigma(\rho)) = ca$. It involves two transitions, namely (x_0, c, x_1) and (x_1, a, x_4), leading the clock reset to be $1 \in Reset((x_0, c, x_1))$ at x_1 and to be $0.6 \in Reset((x_1, a, x_4))$ at x_4, respectively. In addition, according to Eq. (4-3), we have $0 + t_1 \in \Gamma((x_0, c, x_1))$ and $1 + t_2 - t_1 \in \Gamma((x_1, a, x_4))$. Consider another run $\rho' : (x_0, 0) \xrightarrow{(b,0.5)} (x_2, 0.5) \xrightarrow{(c,2)} (x_3, 2) \xrightarrow{(a,2)} (x_2, 0)$, where three transitions (x_0, b, x_2), (x_2, c, x_3), and (x_3, a, x_2) are involved. The transitions (x_0, b, x_2) and (x_2, c, x_3) do not lead the clock to be reset, while (x_3, a, x_2) resets the clock to 0. □

Definition 4.4 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$, a *timed evolution* of G from $t_0 \in \mathbb{R}_{\geq 0}$ to $t \in \mathbb{R}_{\geq 0}$ is defined by a pair $(\sigma(\rho), t) \in (E \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$, where $t_{st}(\rho) = t_0$ and $t_{en}(\rho) \leq t$. Furthermore, we denote as

$$\mathcal{E}(G, t) = \{(\sigma(\rho), t) \mid (\exists \rho \in \mathcal{R}(G))\ x_{st}(\rho) \in X_0, t_{st}(\rho) = 0, t_{en}(\rho) \leq t\}$$

the *timed language of* G *from* 0 *to* $t \in \mathbb{R}_{\geq 0}$.

In other words, a timed evolution of G from t_0 to t is defined as a pair whose first entry is a timed word $\sigma(\rho)$, where ρ is a timed run starts at t_0 from an initial state X_0 and ends at $t_{en}(\rho)$ ($t_{en}(\rho) \leq t$), and whose second entry is the time instant t. Note that $t - t_{en}(\rho)$ is the time that the system stays at the ending discrete state $x_{en}(\rho)$. In addition, the timed language of G from 0 to t contains all possible timed evolutions of G from 0 to t.

Definition 4.5 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ and a time instant $T \in \mathbb{R}_{\geq 0}$, a discrete state $x' \in X$ is said to be *T-reachable* from $x \in X$ if there exists a timed evolution $(\sigma(\rho), t) \in (E \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$ of G such that $t - t_{st}(\rho) = T$, $x_{st}(\rho) = x$, and $x_{en}(\rho) = x'$. In addition, x' is said to be *unobservably T-reachable* from x if x' is T-reachable from x with a timed evolution $(\sigma(\rho), t)$ such that $S(\sigma(\rho)) \in E_{uo}^*$. □

In simple words, a discrete state x' is T-reachable from x if a timed evolution leads the system from x to x' with an elapsed time T. If there exists such a timed evolution that produces no observation further, x' is unobservably T-reachable from x.

Example 4.3 Consider the GTFA G in Fig. 4.1 and a timed evolution $(\sigma(\rho), 2)$, where $\rho : (x_0, 0) \xrightarrow{(b,0.5)} (x_2, 0.5) \xrightarrow{(c,2)} (x_3, 2)$ and $t_{st}(\rho) = 0$. It follows that x_3 is 2-reachable from x_0. □

4.2 Zone Automaton and its dynamics

This section proposes the notion of *zone automaton* associated with a GTFA. The zone automaton provides a purely discrete event description of the behaviour of the GTFA of interest. In detail, each state of a zone automaton is a pair (x, z) whose first element x is a discrete state of the GTFA and whose second element z is a time interval (called a zone) specifying the range of the clock values.

Given a zone automaton, its state evolves because of either the elapsed time or the occurrence of a discrete event. The first case is a time-driven evolution; the second case is an event-driven evolution. In the former, the state evolves whenever a certain amount of time has elapsed as detailed in the following, leading to another state where only the zone is changed while the discrete state associated with the GTFA remains the same. In the latter, i.e., during an event-driven evolution, the clock evolves according to the clock resetting function of the GTFA; thus, a new state of the zone automaton is reached, where the discrete state of the GTFA is typically different (it may be the same only in the case of self-loops in the given GTFA).

This section is divided into two subsections. In the first subsection, we define the zones of a discrete state, and the zone automaton associated with a GTFA, respectively. An algorithm is presented to construct the zone automaton. In the second subsection, we explore the dynamics of a zone automaton and discuss how the timed evolutions of a GTFA are related to the evolutions of its zone automaton.

4.2.1 Zone Automaton

Definition 4.6 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$, *the set of output transitions at a timed state* $(x, \theta) \in X \times \mathbb{R}_{\geq 0}$ is defined as $\mathcal{O}(x, \theta) = \{(x, e, x') \in \Delta \mid (\exists e \in E)(\exists x' \in X)\ \theta \in \Gamma((x, e, x'))\}$, and *the set of input transitions at* (x, θ) is defined as $\mathcal{I}(x, \theta) = \{(x', e, x) \in \Delta \mid (\exists e \in E)(\exists x' \in X)\ \theta \in Reset((x', e, x))\}$. □

In simple words, the set of output transitions at a timed state (x, θ) includes all the transitions that may fire from x with a clock value θ, the set of input transitions at (x, θ) includes all the transitions that may reach x with a reset clock value θ. This leads to the definition of clock *zones* associated with a given discrete state $x \in X$.

Definition 4.7 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$, the *set of zones* of $x \in X$ is defined as a set of time intervals $Z(x) = \{z_1, \cdots, z_n\} \subseteq \mathbb{I},\ n \geq 1$[1] such that $\bigcup_{i=1}^{n} z_i = [0, +\infty)$. In addition, the following conditions hold:

- $\theta < \theta'$ holds for all $\theta \in z_{i-1}$ and for all $\theta' \in z_i$, where $i \in \{2, \cdots, n\}$;
- $z_1 = [0, 0]$ if $x \in X_0$;
- $\mathcal{O}(x, \theta) = \mathcal{O}(x, \theta')$ and $\mathcal{I}(x, \theta) = \mathcal{I}(x, \theta')$ hold for all $\theta, \theta' \in z_i$, where $i \in \{1, \cdots, n\}$;
- for each $\delta \in \Delta$ originating from x or leading to x such that $Reset(\delta) = id$, we have $\{[m, m], (m, m+1), [m+1, m+1], \cdots, [n, n]\} \subseteq Z(x)$ by denoting $\Gamma(\delta) = [m, n]$;
- $\mathcal{O}(x, \theta) \neq \mathcal{O}(x, \theta')$ or $\mathcal{I}(x, \theta) \neq \mathcal{I}(x, \theta')$ holds for all $\theta \in z_{i-1}$ and $\theta' \in z_i$, where $i \in \{2, \cdots, n\}$, if none of the following cases holds: (a) $x \notin X_0$ and $\theta \neq 0$; (b) there exists no $\delta \in \Delta$ originating from x or leading to x such that $Reset(\delta) = id$.

In addition, $prec(z_i) = z_{i-1}$ (resp., $succ(z_i) = z_{i+1}$) is said to be the *preceding* (resp., *succeeding*) zone of $z_i \in Z(x)$, where $i \in \{2, \cdots, n\}$ (resp., $i \in \{1, \cdots, n-1\}$). □

[1] Note that the number of the zones may be different for different discrete states.

The set of zones of a discrete state x follows from the partitioning of the range of the clock values at x into several time intervals to which the clock may belong. The union of all zones in $Z(x)$ covers the interval $[0, +\infty)$. For $i \in \{2, \cdots, n\}$, any value in z_{i-1} is less than any value in z_i, implying that any two zones of x are disjoint. Note that the first zone of an initial discrete state $x \in X_0$ is set to be $[0, 0]$. If $\theta \in z_i$ and $\theta' \in z_i$, where $i \in \{1, \cdots, n\}$, the sets of output and input transitions at two timed states (x, θ) and (x, θ') are identical. Suppose there exists a transition δ going from x (or leading to x) such that the clock is not reset upon its occurrence. In that case, the time interval $\Gamma(\delta)$ is partitioned into a series of time intervals consisting of the integer points belonging to $\Gamma(\delta)$ and the open segments between them. If $x \notin X_0$, $\theta \neq 0$, and there exists no such transition δ, the firability of transitions differs between (x, θ) and (x, θ'), where $\theta \in z_{i-1}$ and $\theta' \in z_i$ for $i \in \{2, \cdots, n\}$.

Example 4.4 Consider again the GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ in Fig. 4.1. In addition, the sets of output and input transitions at (x_0, θ), where θ is a time instant in $z \in Z(x_0)$, are reported in Table 4.2, where the condition $*$ denotes that there exists $\delta \in \mathcal{O}(x_0, \theta) \cup \mathcal{I}(x_0, \theta)$ such that $Reset(\delta) = id$ holds for each $\theta \in z$ and $z \in Z(x_0)$. The set of zones $Z(x_0)$ can be obtained as $\{[0, 0], (0, 1), [1, 1], (1, 3], (3, +\infty)\}$. In detail, we have $[0, 0] \in Z(x_0)$ due to $x_0 \in X_0$, the time interval $\Gamma((x_0, b, x_2)) = [0, 1]$ is partitioned into three zones $[0, 0]$, $(0, 1)$, and $[1, 1]$ because the condition $*$ holds due to the transition $(x_0, b, x_2) \in \Delta$. Accordingly, we have $\{[0, 0], (0, 1), [1, 1]\} \subseteq Z(x_0)$. We can include $(1, 3] \in Z(x_0)$ (resp., $(3, +\infty) \in Z(x_0)$) because $\mathcal{O}(x_0, \theta)$ differs from θ in the zones $[1, 1]$ and $(1, 3]$ (resp., in the zones $(1, 3]$ and $(3, +\infty)$).

Analogously, we can obtain $Z(x_1) = \{[0, 1)$,$[1, 1]$, $(1, 3]$,$(3, +\infty)\}$. The sets of output and input transitions at (x_1, θ) are reported in Table 4.3 for $\theta \in z$ and $z \in Z(x_1)$. Both sets of output and input transitions differ from each two consecutive zones $z \in Z(x_1)$ and $succ(z) \in Z(x_1)$. As for other discrete states in X, we have $Z(x_2) = \{[0, 0]$, $(0, 1)$, $[1, 1]$, $(1, 2)$, $[2, 2]$, $(2, +\infty)\}$, $Z(x_3) = \{[0, 0], (0, 1), [1, 1], (1, 2), [2, 2], (2, +\infty)\}$ and $Z(x_4) = \{[0, 1], (1, +\infty)\}$. □

Given a discrete state x and two zones $z, succ(z) \in Z(x)$, let an event τ denote that at a discrete state x the clock value may evolve from any $\theta \in z$ to any $\theta' \in succ(z)$ as time elapses. In other words, the occurrence of τ implies the time-driven evolution of G that stays at a discrete state x.

Definition 4.8 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$, the *zone automaton* of G is an NFA $G_z = (V, E_\tau, \Delta_z, V_0)$, where

Table 4.2 Sets of output and input transitions at (x_0, θ) for the GTFA G in Fig. 4.1, where $\theta \in [0, +\infty)$.

z	$\mathcal{O}(x_0, \theta), \theta \in z$	$\mathcal{I}(x_0, \theta), \theta \in z$	Condition $*$
$[0, 0]$	$\{(x_0, b, x_2)\}$	$-$	Yes
$(0, 1)$	$\{(x_0, b, x_2)\}$	$-$	Yes
$[1, 1]$	$\{(x_0, b, x_2), (x_0, c, x_1)\}$	$-$	Yes
$(1, 3]$	$\{(x_0, c, x_1)\}$	$-$	No
$(3, +\infty)$	$-$	$-$	No

Table 4.3 Sets of output and input transitions at (x_1, θ) for the GTFA G in Fig. 4.1, where $\theta \in [0, +\infty)$.

z	$\mathcal{O}(x_1, \theta), \theta \in z$	$\mathcal{I}(x_1, \theta), \theta \in z$	Condition $*$
$[0, 1)$	$-$	$-$	No
$[1, 1]$	$\{(x_1, a, x_4)\}$	$\{(x_0, c, x_1)\}$	No
$(1, 3]$	$\{(x_1, a, x_4)\}$	$-$	No
$(3, +\infty)$	$-$	$-$	No

- $V \subseteq X \times \bigcup_{x \in X} Z(x)$ is the finite set of states,
- $E_\tau \subseteq E \cup \{\tau\}$ is the alphabet,
- $\Delta_z \subseteq V \times E_\tau \times V$ is the transition relation, where the transitions in Δ_z are defined by the following rules:
  - $((x, z), \tau, (x, succ(z))) \in \Delta_z$ if $z, succ(z) \in Z(x)$;
  - $((x, z), e, (x', z')) \in \Delta_z$ if $z \in Z(x)$, $z' \in Z(x')$, $(x, e, x') \in \mathcal{O}(x, \theta)$ for all $\theta \in z$, and $(x, e, x') \in \mathcal{I}(x', \theta')$ for all $\theta' \in z'$;
- $V_0 = \{(x, [0, 0]) \mid x \in X_0\}$ is the set of initial states.

We further define the function $f_x : V \to X$ (resp., $f_z : V \to \bigcup_{x \in X} Z(x)$) mapping a state in V to a discrete state in X (resp., a zone of the associated discrete state). □

We use the zone automaton to describe the time-driven and event-driven evolutions of a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$. Each state in a zone automaton is a pair (x, z) with $x \in X$ and $z \in Z(x)$. The alphabet is composed of the events in E and event τ. The transition relation specifies the dynamics of the automaton: starting from a state (x, z), a transition $((x, z), \tau, (x, succ(z))) \in \Delta_z$ corresponds to a time-driven evolution of G from a clock value in z to another clock value in $succ(z)$ while G is at x; a transition $((x, z), e, (x', z')) \in \Delta_z$

goes from (x, z) to (x', z'), indicating that the occurrence of event e yields discrete state x' with a clock value in z' when the current discrete state of the system is x and the current clock is in z. The set of initial states is the set of pairs of a discrete state $x \in X_0$ and a time interval $[0, 0]$. A zone automaton can be represented as a graph. In particular, we represent the nodes as rectangular boxes in the graph.

Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$, the zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ can be constructed by Algorithm 6. A temporary set of states V_{new} is introduced, containing all states in V that still need to be explored in order to compute their output transitions. A *while* loop is repeated until $V_{new} = \emptyset$. After selecting an element $(x, z) \in V_{new}$, a transition $((x, z), \tau, (x, succ(z)))$ is set in Δ_z if $succ(z)$ is a zone at x. For each transition $(x, e, x') \in \Delta$, if the transition satisfies $z \subseteq \Gamma((x, e, x'))$, then for each zone $z' \in Z(x')$, if $Reset((x, e, x')) \neq id$ and $z' \subseteq Reset((x, e, x'))$, or if $Reset((x, e, x')) = id$ and $z = z'$, a transition labeled with e is set from $v = (x, z)$ to $v' = (x', z')$. To avoid redundant repetitions of the *while* loop, the state v' is included in V_{new} if v' is neither in V nor in V_{new}. The *while* loop stops once all states in V_{new} have been explored.

We discuss the computational complexity of Algorithm 6 as follows. Let $Q_x = \{(x, e, x') \in \Delta \mid e \in E, x' \in X\} \cup \{(x', e, x) \in \Delta \mid e \in E, x' \in X\}$ be the set of output and input transitions of discrete state $x \in X$. Our approach partitions the range of clock values $[0, +\infty)$ at $x \in X$ into a set of zones $Z(x)$ which depends on set Q_x. A simple analysis shows that the maximum number of zones is bounded by $2|Q_x| + 1$. As a result, by letting $q = \max_{x \in X} |Q_x|$, the maximal number of states of G_z satisfies $|V| \leq (2q+1)|X|$. In Algorithm 6, the *while* loop at Step 2 is executed $|V|$ times, the *for* loop at Step 7 is executed at most q times, and the *for* loop at Step 9 is executed at most $2q + 1$ times; consequently, the time complexity is $\mathcal{O}(q^3|X|)$.

Example 4.5 Consider the GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ in Fig. 4.1. The zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ is shown in Fig. 4.2. The initial state $(x_0, [0, 0])$ implies that G starts from x_0 at clock value 0. A transition labeled with an event τ implies a time-driven evolution of G. For instance, a transition $((x_0, [0, 0], \tau, (x_0, (0, 1)))$ represents that the clock may evolve from the value in $[0, 0]$ to any value in $(0, 1)$ if G is at x_0. Meanwhile, a transition labeled with an event in E implies an event-driven evolution of G. For instance, three transitions labeled with b go from $(x_0, [0, 0])$ to $(x_2, [0, 0])$, from $(x_0, (0, 1))$ to $(x_2, (0, 1))$, and from $(x_0, [1, 1])$ to $(x_2, [1, 1])$, respectively. It represents an event-driven evolution from x_0 to x_2 under the occurrence of a transition (x_0, b, x_2). Note that the pair $(x_1, [0, 1))$ is not a state in V because it cannot be reached by any transition. □

Algorithm 6 Construction of the zone automaton associated with a GTFA

Input: A GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$
Output: The zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$

$V = \emptyset$, $E_\tau = E \cup \{\tau\}$, $\Delta_z = \emptyset$, $V_0 = \{(x, [0,0]) \mid x \in X_0\}$, and $V_{new} = V_0$
while $V_{new} \neq \emptyset$ **do**
  select a $v = (x, z) \in V_{new}$
  if $succ(z) \in Z(x)$ **then**
    let $\bar{v} = (x, succ(z))$, $\Delta_z = \Delta_z \cup \{(v, \tau, \bar{v})\}$, and $V_{new} = V_{new} \cup \{\bar{v}\}$
  end if
  for each $(x, e, x') \in \Delta$ **do**
    if $z \subseteq \Gamma((x, e, x'))$ **then**
      for each $z' \in Z(x')$ **do**
        if $Reset((x, e, x')) \neq id \wedge z' \subseteq \Gamma((x, e, x')) \vee Reset((x, e, x')) = id \wedge z = z'$ **then**
          let $v' = (x', z')$ and $\Delta_z = \Delta_z \cup \{(v, e, v')\}$
          if $v' \notin V \cup V_{new}$ **then**
            $V_{new} = V_{new} \cup \{v'\}$
          end if
        end if
      end for
    end if
  end for
  let $V = V \cup \{v\}$ and $V_{new} = V_{new} \setminus \{v\}$
end while
return $G_z = (V, E_\tau, \Delta_z, V_0)$.

4.2.2 Dynamics of Zone Automaton

In this subsection, we first introduce the notion of *τ-run at a discrete state $x \in X$* that corresponds to the elapse of time with no event occurrence. Next, a run in G_z is considered, including also the occurrence of discrete events, to show the time-driven evolution and the event-driven evolution of G. We show that the problem of investigating the reachability of a discrete state in G can be reduced to the reachability analysis of a state in the zone automaton G_z.

Definition 4.9 Consider a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ and a discrete state $x \in X$, where the number of the zones in $Z(x)$ is $n \geq 1$. Let $G_z = (V, E_\tau, \Delta_z, V_0)$ be the zone automaton of G. A *τ-run at x* of length k $(1 \leq k \leq n)$ is defined as a sequence of k states $(x, z_{(i)}) \in V$ $(i = 1, \cdots, k)$, and event $\tau \in E_\tau$, represented as

$$\rho_\tau(x) : (x, z_{(1)}) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (x, z_{(k)})$$

such that $((x, z_{(i)}), \tau, (x, z_{(i+1)})) \in \Delta_z$ holds for $i \in \{1, \cdots, k-1\}$. The *starting state* and

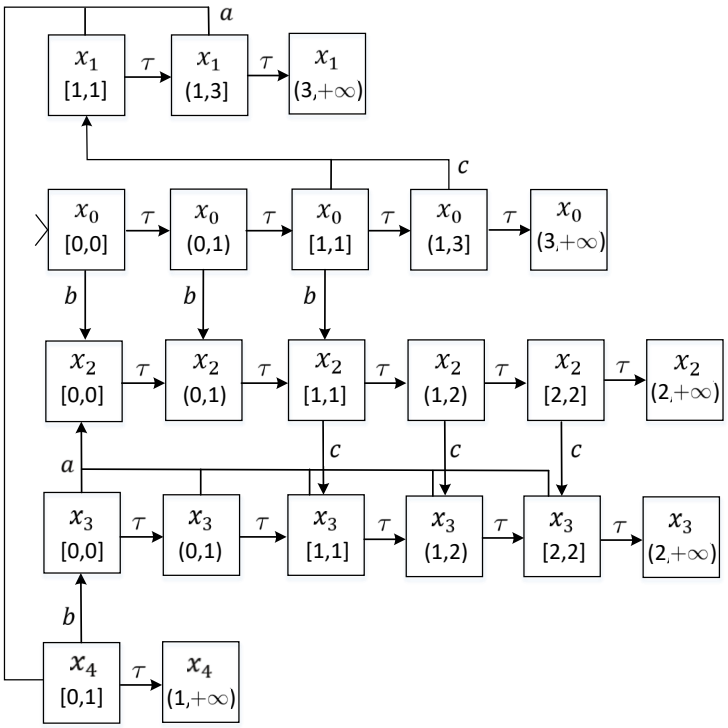


Fig. 4.2 Zone automaton G_z of the GTFA G in Fig. 4.1.

the *ending state* of $\rho_\tau(x)$ are denoted as $v_{st}(\rho_\tau(x)) = (x, z_{(1)})$ and $v_{en}(\rho_\tau(x)) = (x, z_{(k)})$, respectively. The *duration range* of $\rho_\tau(x)$ is the time distance of $z_{(1)}$ and $z_{(k)}$, denoted as $d(\rho_\tau(x)) = D(z_{(1)}, z_{(k)})$. □

A τ-run at x represents a series of evolutions of states in G_z involving the states associated with x and event τ. It essentially represents the time elapsing discretely while G is at x. The duration range $d(\rho_\tau(x))$ is an interval describing the possible duration of the run. Next we define the notion of a *run* of a zone automaton to represent a hybrid evolution of the associated GTFA.

Definition 4.10 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, a run of length $k \geq 0$ in G_z is a sequence of $k+1$ τ-runs $\rho_\tau(x_{(i)})$ $(i = 0, \cdots, k)$ at $x_{(i)} \in X$, and k events $e_i \in E$ $(i = 1, \cdots, k)$, represented as

$$\bar{\rho} : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \rho_\tau(x_{(1)}) \cdots \xrightarrow{e_k} \rho_\tau(x_{(k)}),$$

such that $(v_{en}(\rho_\tau(x_{(i-1)})), e_i, v_{st}(\rho_\tau(x_{(i)}))) \in \Delta_z$ holds for $i \in \{1, \cdots, k\}$. In addition, the *starting state* and the *ending state* of $\bar{\rho}$ are defined as $v_{st}(\bar{\rho}) = v_{st}(\rho_\tau(x_{(0)}))$ and $v_{en}(\bar{\rho}) = v_{en}(\rho_\tau(x_{(k)}))$, respectively. The *duration range* of $\bar{\rho}$ is defined as $d(\bar{\rho}) = \bigoplus_{i=1}^{k} d(\rho_\tau(x_i))$.

The logical word generated by $\bar{\rho}$ is denoted as $s(\bar{\rho}) = e_1 \cdots e_k$ via a function defined as $s : E_\tau^* \to E^*$. The set of runs generated by G_z is defined as $\mathcal{R}_z(G_z)$. □

The dynamics of a zone automaton G_z can be represented by the runs in G_z; in addition, such runs correspond to an evolution of the GTFA G. In simple words, any two consecutive τ-runs $\rho_\tau(x_{(i-1)})$ and $\rho_\tau(x_{(i)})$ are connected by an evolution of states of G_z caused by an event $e_i \in E$. The logical word of $\bar{\rho}$ is the sequence of events in E that have been involved in $\bar{\rho}$. The duration range of $\bar{\rho}$ is evaluated by summing up the duration ranges of all the involved τ-runs.

Recall that given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ and a time instant $T \in \mathbb{R}_{\geq 0}$, a discrete state $x' \in X$ is T-reachable from $x \in X$ if there exists a timed evolution $(\sigma(\rho), t)$ such that $x_{st}(\rho) = x$, $x_{en}(\rho) = x'$, and $t - t_{st}(\rho) = T$. We next formalize a result that provides a sufficient and necessary condition for the reachability of a discrete state in a GTFA and explains the correlation of the dynamics of a GTFA and that of its zone automaton.

Theorem 4.1 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$, its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ and a time instant $T \in \mathbb{R}_{\geq 0}$, $x' \in X$ is T-reachable from $x \in X$ if and only if there exists a run $\bar{\rho}$ in G_z such that $T \in d(\bar{\rho})$, $v_{st}(\bar{\rho}) = (x, z)$, and $v_{en}(\bar{\rho}) = (x', z')$, where $z \in Z(x)$ and $z' \in Z(x')$. □

Proof: (if) Let $x = x_{(0)}$, $x' = x_{(k)}$, $t_0, t_1, \cdots, t_k \in \mathbb{R}_{\geq 0}$ and $\bar{\rho} : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \cdots \xrightarrow{e_k} \rho_\tau(x_{(k)})$ be a run such that $e_i \in E$, $t_i - t_{i-1} \in d(\rho_\tau(x_{(i-1)}))$ for $i \in \{1, \cdots, k\}$, $t - t_k \in d(\rho_\tau(x_{(k)}))$, and $T = t - t_0 \in d(\bar{\rho})$. It can be inferred that $(v_{en}(\rho_\tau(x_{(i-1)})), e_i, v_{st}(\rho_\tau(x_{(i)}))) \in \Delta_z$ holds for $i \in \{1, \cdots, k\}$. Accordingly, for $i \in \{1, \cdots, k\}$, there exist $\theta_{(i-1)} + t_i - t_{i-1} \in f_z(v_{en}(\rho_\tau(x_{(i-1)})))$ and $\theta_{(i)} \in f_z(v_{st}(\rho_\tau(x_{(i)})))$ such that $(x_{(i-1)}, e_i, x_{(i)}) \in \mathcal{O}(x_{(i-1)}, \theta_{(i-1)} + t_i - t_{i-1})$ and $(x_{(i-1)}, e_i, x_{(i)}) \in \mathcal{I}(x_{(i)}, \theta_{(i)})$. It is obvious that there exists a timed evolution $(\sigma(\rho), t)$, where the timed run ρ:$(x_{(0)}, \theta_{(0)}) \xrightarrow{(e_1,t_1)} \cdots \xrightarrow{(e_k,t_k)} (x_{(k)}, \theta_{(k)})$ satisfies that $T(\rho) = t_k - t_0$. Thus, x' is T-reachable from x.

(only if) Let $(\sigma(\rho), t) \in (E \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$ be a timed evolution of G from $t_0 \in \mathbb{R}_{\geq 0}$ to $t \in \mathbb{R}_{\geq 0}$ such that $x_{st}(\rho) = x$ and $x_{en}(\rho) = x'$. The proof is made by induction on the length k of the timed run ρ generated by G from t_0 to t. The base case is for the timed run ρ of length 0 that involves only the discrete state x and no transition in G. We have $\sigma(\rho) = \lambda$ and $x_{st}(\rho) = x_{en}(\rho) = x = x'$. There exists a run $\bar{\rho} : (x, \bar{z}) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (x, z)$ in G_z, where $T = t - t_0 \in d(\bar{\rho})$. Thus the base case holds.

By denoting $x = x_{(0)}$ and $x' = x_{(k)}$, the induction hypothesis is that the existence of a timed evolution $(\sigma(\rho), t)$ generating from t_0, where $\rho : (x_{(0)}, \theta_{(0)}) \xrightarrow{(e_1,t_1)} \cdots \xrightarrow{(e_k,t_k)} (x_{(k)}, \theta_{(k)})$ of length $k \geq 1$ with $x_{(i)} \in X$ and $e_i \in E$ for all $i \in \{1, \cdots, k\}$, implies the existence of a run $\bar{\rho} : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \cdots \xrightarrow{e_k} \rho_\tau(x_{(k)})$ in G_z such that $t - t_0 \in d(\bar{\rho})$, $t - t_k \in d(\rho_\tau(x_{(k)}))$, $t_i - t_{i-1} \in d(\rho_\tau(x_{(i-1)}))$ for $i \in \{1, \cdots, k\}$, $v_{st}(\bar{\rho}) = (x_{(0)}, z)$ and $v_{en}(\bar{\rho}) = (x_{(k)}, \bar{z})$, where $z \in Z(x_{(0)})$, and $\bar{z} \in Z(x_{(k)})$.

We now prove that the same implication holds for a timed evolution $(\sigma(\rho'), t')$, where ρ': $\rho \xrightarrow{(e_{k+1},t)} (x_{(k+1)}, \theta_{(k+1)})$. According to ρ', we have $\theta_{(k+1)} \in Reset((x_{(k)}, e_{k+1}, x_{(k+1)}))$ and $\theta_{(k)} + t - t_k \in \Gamma((x_{(k)}, e_{k+1}, x_{(k+1)}))$, which implies that $(x_{(k)}, e_{k+1}, x_{(k+1)}) \in \mathcal{O}(x_{(k)}, \theta_{(k)} + t - t_k)$ and $(x_{(k)}, e_{k+1}, x_{(k+1)}) \in \mathcal{I}(x_{(k+1)}, \theta_{(k+1)})$; consequently, there exists a run $\bar{\rho}' : \bar{\rho} \xrightarrow{e_{k+1}} \rho_\tau(x_{(k+1)})$ in G_z such that $\theta_{(k+1)} \in f_z(v_{st}(\rho_\tau(x_{(k+1)})))$ and $\theta_{(k+1)} + t' - t \in f_z(v_{en}(\rho_\tau(x_{(k+1)})))$. Therefore, we have $t' - t_0 \in d(\bar{\rho}')$ according to $t' - t \in d(\rho_\tau(x_{(k+1)})))$ and $t - t_0 \in d(\bar{\rho})$. □

Given a time instant $T \in \mathbb{R}_{\geq 0}$, Theorem 4.1 provides a necessary and sufficient condition to determine the T-reachability of a discrete state x' from x. According to Theorem 4.1, given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$, if $x' \in X$ is T-reachable from $x \in X$, then in its zone automaton there exists a run that originates from (x, z) and reaches (x', z'), where $z \in Z(x)$ and $z' \in Z(x')$. In addition, T belongs to the duration range of that run. In turn, given a run $\bar{\rho}$ in G_z such that $T \in d(\bar{\rho})$, it can be concluded that the discrete state associated with $v_{en}(\bar{\rho})$ is T-reachable from the discrete state associated with $v_{st}(\bar{\rho})$. In simple words, the T-reachability of x' from x can be analyzed by exploring an appropriate run in G_z.

Example 4.6 Consider the GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ in Fig. 4.1 and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ in Fig. 4.2. There exists a timed evolution $((c, 1.5)(a, 3), 4)$ of G from 0 to 4 such that x_4 is 4-reachable from x_0. Accordingly, there exists a run $\bar{\rho} : \rho_\tau(x_0) \xrightarrow{c} \rho_\tau(x_1) \xrightarrow{a} \rho_\tau(x_4)$, where $\rho_\tau(x_0) : (x_0, [0, 0])) \xrightarrow{\tau} (x_0, (0, 1)) \xrightarrow{\tau} (x_0, [1, 1]) \xrightarrow{\tau} (x_0, (1, 3])$, $\rho_\tau(x_1) : (x_1, [1, 1]) \xrightarrow{\tau} (x_1, (1, 3])$ and $\rho_\tau(x_4) : (x_4, [0, 1])$. □

4.3 State Estimation of Generalized Timed Finite Automata

Given a partially observed GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ with a partition of alphabet $E = E_o \cup E_{uo}$, in this section we develop an approach for state estimation based on the zone automaton G_z, given a timed observation $(\sigma_o, t) \in (E_o \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$.

This section aims at calculating the set $\mathcal{X}(\sigma_o, t)$ according to Eq. 3.17, which includes the states reached by the timed evolution $(\sigma(\rho), t)$ consistent with (σ_o, t) and is the output

of the state estimation process shown in Fig. 3.5.

Assumption 4.1 (RO: Reinitialized observations) The clock is reset upon the occurrence of any observable event, i.e.,

$$(x, e, x') \in \Delta,\ e \in E_o \quad \Longrightarrow \quad Reset(x, e, x') \neq id.$$

□

This assumption is necessary to ensure that the defined zone automaton contains all relevant information to estimate the discrete state. Consider a scenario where an observable event occurs without resetting the clock. In such a case, for future estimations one may need to keep track of this exact value adding new states to the zone automaton: thus, the state space of the zone automaton could grow indefinitely as new events are observed.

We partition this section into two subsections. In the first subsection, we consider the case where G produces no observation, and prove that we can estimate the possible current discrete states with the range of clock values as time elapses without any observation by looking at appropriate runs in G_z. This is an intermediate step towards the solution of the state estimation problem under partial observation. In the second subsection, we take into account the information coming from the observation of new events at certain time instants, and prove that the discrete states consistent with a timed observation (σ_o, t) and the range of clock value associated with each estimated discrete state can be inferred following a certain number of runs in the zone automaton G_z.

4.3.1 State Estimation with No Observation

Given a state of a zone automaton, which refers to a discrete state of the associated GTFA and a range of the possible clock value, Theorem 4.2 provides a criterion to estimate the set of states in which the zone automaton can be when no observation is received for a given time instant $t \in \mathbb{R}_{\geq 0}$. To formalize this we preliminarily propose the following definition.

Definition 4.11 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ with the set of unobservable events E_{uo} and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, the following set of states of G_z

$$V_\lambda(x, z, t) = \{v_{en}(\bar{\rho}) \in V \mid (\exists \bar{\rho} \in \mathcal{R}_\tau(G_z))t \in d(\bar{\rho}), v_{st}(\bar{\rho}) = (x, z), s(\bar{\rho}) \in E_{uo}^*\}$$

is said to be *λ-estimation from $(x, z) \in V$ within $t \in \mathbb{R}_{\geq 0}$*. □

Given a zone automaton, the λ-estimation from $(x, z) \in V$ within $t \in \mathbb{R}_{\geq 0}$ is the set of states of G_z that can be reached following a run of duration t, originating at (x, z) and producing no observation. If (x', z') belongs to the λ-estimation from $(x, z) \in V$ within $t \in \mathbb{R}_{\geq 0}$, then the zone z' associated with x' specifies the range of the value of the clock.

Given a GTFA G with a set of discrete states X, a set of unobservable events E_{uo} and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, Algorithm 7 computes the λ-estimation from $(x, z) \in V$ within $t \in \mathbb{R}_{\geq 0}$. It first initializes a set R as the singleton $\{(x, z, [0, 0])\}$. The *while* loop selects an element $r = (\bar{x}, \bar{z}, \bar{I})$ in R at each iteration. For all zones of the discrete state $\bar{x}$, we check if there exists a transition $((\bar{x}, \bar{\bar{z}}), e, (x', z')) \in \Delta_z$ with $e \in E_{uo}$. If so, let $I' = \bar{I} \bigoplus D(\bar{z}, \bar{\bar{z}})$ and add to R the pair (x', z', I'). Note that (x', z') is the state of G_z reached by unobservable transitions and I' is the range of time taken to reach (x', z'). In addition, if t belongs to I', it can be inferred that $(\bar{x}, \bar{\bar{z}})$ and (x', z') can be reached from (x, z) within t; consequently, the set $V_\lambda(x, z, t)$ is updated by including $(\bar{x}, \bar{\bar{z}})$ and (x', z'). At the end of each *while* loop, the explored $r \in R$ is deleted from R. Finally, the algorithm returns $V_\lambda(x, z, t)$.

Next we discuss the computational complexity of Algorithm 7. The *while* loop at Step 2 is executed at most $|V|$ times; the *for* loop at Step 4 is executed at most $2q + 1$ times, and the *for* loop at Step 5 is executed at most q times. The complexity of Algorithm 7 is thus $\mathcal{O}(q(2q + 1)|V|) = \mathcal{O}(q^3|X|)$.

Algorithm 7 Computation of the λ-estimation from a state of a zone automaton within a time instant

Input: A TFA G with a set of discrete states X and a set of unobservable events E_{uo}, a zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, a state $(x, z) \in V$, and a time instant $t \in \mathbb{R}_{\geq 0}$
Output: λ-estimation $V_\lambda(x, z, t)$

1: let $R = \{(x, z, [0, 0])\}$ and $V_\lambda = \emptyset$
2: **while** $R \neq \emptyset$ **do**
3: select $r = (\bar{x}, \bar{z}, \bar{I}) \in R$
4: **for** each $\bar{\bar{z}} \in Z(\bar{x})$ **do**
5: **for** each $((\bar{x}, \bar{\bar{z}}), e, (x', z')) \in \Delta_z$ with $e \in E_{uo}$ **do**
6: let $I' = \bar{I} \bigoplus D(\bar{z}, \bar{\bar{z}})$ and $R = R \cup \{(x', z \bigoplus \bar{I})\}$
7: **end for**
8: **if** $t \in I'$ **then**
9: let $V_\lambda(x, z, t) = V_\lambda(x, z, t) \cup \{(\bar{x}, \bar{\bar{z}}), (x', z')\}$
10: **end if**
11: **end for**
12: let $R = R \setminus \{r\}$
13: **end while**
14: **return** $V_\lambda(x, z, t)$.

Theorem 4.2 Given a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ with set of unobservable events E_{uo} and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, $x' \in X$ is unobservably T-reachable from $x \in X$, where $T \in \mathbb{R}_{\geq 0}$, if and only if there exist $z \in Z(x)$ and $z' \in Z(x')$ such that $(x', z') \in V_\lambda(x, z, T)$. □

Proof: (if) Let us suppose that there exist $z \in Z(x)$ and $z' \in Z(x')$ such that $(x', z') \in V_\lambda(x, z, t)$. Then, there exists a run $\bar{\rho}$ in G_z such that $v_{st}(\bar{\rho}) = (x, z)$, $v_{en}(\bar{\rho}) = (x', z')$, $s(\bar{\rho}) \in E^*_{uo}$ and $T \in d(\bar{\rho})$. According to Theorem 4.1, it implies that x' is unobservably T-reachable from x.

(only if) Let x' be unobservably T-reachable from x. Then, there exists a timed evolution $(\sigma(\rho), t)$ from t_0 such that $x_{st}(\rho) = x$, $x_{en}(\rho) = x'$, $T = t - t_0$, and $s(\bar{\rho}) \in E^*_{uo}$. Accordingly, there exists a run $\bar{\rho}$ in G_z such that $v_{st}(\bar{\rho}) = (x, z)$, $v_{en}(\bar{\rho}) = (x', z')$, $s(\bar{\rho}) \in E^*_{uo}$ and $T \in d(\bar{\rho})$, where $z \in Z(x)$ and $z' \in Z(x')$. Thus, there exists $(x', z') \in V_\lambda(x, z, t)$. □

According to Theorem 4.2, the λ-estimation from (x, z) within T reveals the unobservably T-reachable discrete states from x with a clock value $\theta \in z$, i.e., the set of discrete states of G when it starts its timed evolution from (x, θ) with $\theta \in z$ and produces no observation for time T.

Example 4.7 Consider the GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ in Fig. 4.1 with $E_o = \{a\}$, $E_{uo} = \{b, c\}$, and its zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$ in Fig. 4.2. We have

$$V_\lambda(x_0, [0, 0], 1) = \{(x_0, [1, 1]), (x_1, [1, 1]), (x_2, [1, 1]), (x_3, [1, 1])\}.$$

This results from the exhaustive enumeration of all the runs in G_z of duration 1 starting from $(x_0, [0, 0])$ and involving no observable event:

1. $\bar{\rho}_1 : \rho_\tau(x_0)$,
2. $\bar{\rho}_2 : \rho_\tau(x_0) \xrightarrow{c} \rho_\tau(x_1)$,
3. $\bar{\rho}_3 : \rho_\tau(x_0) \xrightarrow{b} \rho_\tau(x_2)$,
4. $\bar{\rho}_4 : \rho_\tau(x_0) \xrightarrow{b} \rho_\tau(x_2) \xrightarrow{c} \rho_\tau(x_3)$,

where $\rho_\tau(x_0) : (x_0, [0, 0]) \xrightarrow{\tau} (x_0, (0, 1)) \xrightarrow{\tau} (x_0, [1, 1])$, $\rho_\tau(x_1) : (x_1, [1, 1])$, $\rho_\tau(x_2) : (x_2, [1, 1])$, and $\rho_\tau(x_3) : (x_3, [1, 1])$.

Now, let us focus on $(x_3, [1, 1]) \in V_\lambda(x_0, [0, 0], 1)$. That is to say, if G starts its evolution from x_0 at time 0 and produces no observation during the whole time interval $[0, 1]$, G can

Table 4.4 State estimation of the GTFA G in Fig. 4.1 under no observation for $t \in [0, 3]$.

k	Time interval I_k	λ-estimation $V_\lambda(x_0, [0,0], t)$, where $t \in I_k$	$\mathcal{X}(\lambda, t), t \in I_k$
0	[0,0]	$(x_0, [0,0]), (x_2, [0,0])$	$\{x_0, x_2\}$
1	(0,1)	$(x_0, (0,1)), (x_2, (0,1))$	$\{x_0, x_2\}$
2	[1,1]	$(x_0, [1,1]), (x_1, [1,1]), (x_2, [1,1]), (x_3, [1,1])$	$\{x_0, x_1, x_2, x_3\}$
3	(1,2)	$(x_0, (1,3]), (x_1, [1,1]), (x_1, (1,3]), (x_2, (1,2)), (x_3, (1,2))$	$\{x_0, x_1, x_2, x_3\}$
4	[2,2]	$(x_0, (1,3]), (x_1, [1,1]), (x_1, (1,3]), (x_2, [2,2]), (x_3, [2,2])$	$\{x_0, x_1, x_2, x_3\}$
5	(2,3)	$(x_0, (1,3]), (x_1, [1,1]), (x_1, (1,3]), (x_2, (2,+\infty)), (x_3, (2,+\infty))$	$\{x_0, x_1, x_2, x_3\}$
6	[3,3]	$(x_0, (1,3]), (x_1, [1,1]), (x_1, (1,3]), (x_2, (2,+\infty)), (x_3, (2,+\infty))$	$\{x_0, x_1, x_2, x_3\}$

be at x_3 at time 1 with clock value in the zone $[1, 1]$. Table 4.4 summarizes the λ-estimation and the set of discrete states consistent with (λ, t), where t belongs to a time interval in the set $\{[0, 0], (0, 1), [1, 1], (1, 2), [2, 2], (2, 3), [3, 3]\}$. □

4.3.2 State Estimation with Partial Observation

In this subsection we focus on the most general state estimation problem when a timed observation is received as a pair of a non-empty timed word and a time instant. We first propose a general result that characterizes the set of discrete states of a GTFA consistent with a given timed observations, by means of the states reachable in its zone automaton.

Theorem 4.3 Consider a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ with the set of observable events E_o and zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$. Given a timed observation $(\sigma_o, t) \in (E_o \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$, a discrete state $x \in X$ is consistent with (σ_o, t) if and only if there exists a run $\bar{\rho}$ in G_z such that $f_x(v_{st}(\bar{\rho})) \in X_0$, $f_x(v_{en}(\bar{\rho})) = x$, $t \in d(\bar{\rho})$ and $P_l(s(\bar{\rho})) = S(\sigma_o)$, where $P_l : E^* \to E_o^*$ is the natural projection. □

Proof: (if) In the case that no observation is contained in σ_o, let $\bar{\rho} : \rho_\tau(x_0)$ be a run in G_z, where $x_0 \in X_0$. We have $f_x(v_{en}(\bar{\rho})) = x_0$, $t \in d(\bar{\rho})$ and $\sigma_o = \lambda$. In this case there exists only one timed evolution $(\lambda, t) \in \mathcal{E}(G, t)$ such that $x_0 \in \mathcal{X}(\lambda, t)$. In the case that there exists one or more event occurrences, let us suppose that $x_{(0)} \in X_0$, $x_{(k)} = x$, and a run in G_z as $\bar{\rho} : \rho_\tau(x_{(0)}) \xrightarrow{e_1} \cdots \xrightarrow{e_k} \rho_\tau(x_{(k)})$ such that $t \in d(\bar{\rho})$, $x_{(i)} \in X$ and $e_i \in E$ for $i \in \{0, \cdots, k\}$. According to Algorithm 7, it can be inferred that there exists a timed evolution $(\sigma(\rho), t)$ such that $x_{(k)}$ is t-reachable from $x_{(0)}$, where the timed run $\rho : (x_{(0)}, \theta_{(0)}) \xrightarrow{(e_1,t_1)} \cdots \xrightarrow{(e_k,t_k)} (x_{(k)}, \theta_{(k)})$ satisfies $t - t_k \in d(\rho_\tau(x_{(k)}))$, $t_i - t_{i-1} \in d(\rho_\tau(x_{(i-1)}))$, $\theta_{(i)} \in Reset((x_{(i-1)}, e_i, x_{(i)}))$ and $\theta_{(i-1)} + t_i - t_{i-1} \in \Gamma((x_{(i-1)}, e_i, x_{(i)}))$ for $i \in \{1, \cdots, k\}$. According to $P_l(\bar{s}(\bar{\rho})) = S(\sigma_o)$, we have $(\sigma(\rho), t) \in \mathcal{S}(\sigma_o, t)$. Obviously, $x \in \mathcal{X}(\sigma_o, t)$ holds.

(only if) In the case that $\sigma_o = \lambda$, $x \in \mathcal{X}(\lambda, t)$ holds. There exists a run $\bar{\rho}$ in G_z such that x is unobservably t-reachable from $x_0 \in X_0$. Consequently, $f_x(v_{en}(\bar{\rho})) = x$, $t \in d(\bar{\rho})$ and $\bar{s}(\bar{\rho}) = \varepsilon$ hold. In the case that $\sigma_o = (e_{o1}, t_1) \cdots (e_{ok}, t_k)$, where $k \geq 1$, $0 \leq t_1 \leq \cdots \leq t_k \leq t$ and $e_{oi} \in E_o$ ($i \in \{1, \cdots, k\}$), it is $x \in \mathcal{X}(\sigma_o, t)$, implying that x is t-reachable from $x_0 \in X_0$. Consequently, there exists a run $\bar{\rho}$ in G_z such that $t \in d(\bar{\rho})$, $f_x(v_{st}(\bar{\rho})) = x_0$ and $f_x(v_{en}(\bar{\rho})) = x$. Let $\bar{\rho}$ be a sequence of k runs $\bar{\rho}_i$ and k observable events e_{oi}, where $i \in \{1, \cdots, k\}$, as $\bar{\rho} : \bar{\rho}_0 \xrightarrow{e_{o1}} \cdots \xrightarrow{e_{ok}} \bar{\rho}_k$. Obviously, $\bar{s}(\bar{\rho}) = e_{o1} \cdots e_{ok}$, $t - t_k \in d(\bar{\rho}_k)$ and $t_i - t_{i-1} \in d(\bar{\rho}_{i-1})$ hold for each $i \in \{1, \cdots, k\}$. □

Consider a timed observation (σ_o, t) produced by a GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$; $x \in X$ is consistent with (σ_o, t) when there exists a run $\bar{\rho}$ in G_z that produces the same logical observation as σ_o from a discrete state in X_0 at 0 and reaches x at t. In turn, $x \in \mathcal{X}(\sigma_o, t)$ can be concluded according to the run $\bar{\rho}$ in G_z. In more detail, the run $\bar{\rho}$ contains no observable event if $\sigma_o = \lambda$. This naturally implies that $x \in \mathcal{X}(\lambda, t)$. If $\sigma_o \neq \lambda$, the run $\bar{\rho}$ is a sequence of $k+1$ runs $\bar{\rho}_i$, $i \in \{0, \cdots, k\}$, and k observable events $e_{oi} \in E_o$, $i \in \{1, \cdots, k\}$, as $\bar{\rho} : \bar{\rho}_0 \xrightarrow{e_{o1}} \cdots \xrightarrow{e_{ok}} \bar{\rho}_k$. For each $i \in \{0, \cdots, k\}$, we have $s(\bar{\rho}_i) = \varepsilon$, implying that $\bar{\rho}_i$ produces no observation. For each $i \in \{1, \cdots, k\}$, an observable event $e_{oi} \in E_o$ leads the state evolution from $v_{en}(\bar{\rho}_{i-1})$ to $v_{st}(\bar{\rho}_i)$.

Proposition 4.1 Consider a GTFA G with set of observable events E_o that produces two timed observations $(\sigma_o, t_i), (\sigma_o, t) \in (E_o \times \mathbb{R}_{\geq 0})^* \times \mathbb{R}_{\geq 0}$, where $\sigma_o = (e_{o1}, t_1) \cdots (e_{oi}, t_i)$ and $t_1 \leq \cdots \leq t_i \leq t$ for $i \geq 1$. For each timed state (x, θ) reached by a timed evolution in $\mathcal{S}(\sigma_o, t_i)$, and for each $v \in V_\lambda(x, z, t - t_i)$, where $z \in Z(x)$ and $\theta \in z$, it holds $f_x(v) \in \mathcal{X}(\sigma_o, t)$ if Assumption RO holds. □

Proof: Let a timed run $\rho : (x_0, 0) \xrightarrow{(e_1, t_1)} (x_{(1)}, \theta_{(1)}) \cdots \xrightarrow{(e_k, t_k)} (x_{(k)}, \theta_{(k)}) \xrightarrow{(e_{oi}, t_i)} (x, \theta)$, where $x_{(1)}, \cdots, x_{(k)} \in X$, $e_1, \cdots, e_k \in E$, and $P(\sigma(\rho)) = \sigma_o$, the timed state (x, θ) is reached by the timed evolution $(\sigma(\rho), t_i) \in \mathcal{S}(\sigma_o, t_i)$. According to Theorem 4.3, it can be inferred that there exists a run in G_z as $\bar{\rho} : (x_0, [0, 0]) \to \cdots \to (x, z)$, where $\theta \in z$. If Assumption RO holds, it implies that there exists $z \in Z(x)$ such that $\theta \in z$ and $z \subseteq Reset((x_{(k)}, e_{oi}, x))$. Given $v \in V_\lambda(x, z, t - t_i)$, there exists a run $\bar{\rho}' : (x, z) \to \cdots \to v$ such that $P_l(s(\bar{\rho}')) = \varepsilon$ and $t - t_i \in d(\bar{\rho}')$. Given $\bar{\rho}$ and $\bar{\rho}'$, it can be inferred that $f_x(v) \in \mathcal{X}(\sigma_o, t)$ according to Theorem 4.3. □

The previous proposition shows that the state estimation can be updated by computing associated λ-estimations under Assumption RO. In more detail, given a timed state (x, θ) reached by a timed evolution consistent with a timed observation (σ_o, t_i), the λ-estimation

$V_\lambda(x, z, t - t_i)$, where $t \geq t_i$, $\theta \in z$ and $z \in Z(x)$, provides the discrete states consistent with the observation (σ_o, t).

Based on the previous results, Algorithm 8 summarizes the proposed approach to compute the set of discrete states consistent with a timed observation (σ_o, t). It can be explained as follows. Consider a timed observation (σ_o, t) with $\sigma_o = (e_{o1}, t_1) \cdots (e_{on}, t_n)$ $(n \geq 1)$, where $e_{o1}, \cdots, e_{on} \in E_o$. The timed observation (σ_o, t) is updated whenever an observable event e_{oi} occurs at a time instant t_i, where $i \in \{1, \cdots, n\}$. The algorithm provides a set of states $V_\lambda \subseteq V$ while time elapses in $[t_{i-1}, t_i]$ with no event being observed, in addition to a set of states $\bar{V}_i \subseteq V$ of G_z consistent with each new observation (e_{oi}, t_i), where $i \in \{1, \cdots, n\}$ and $t_0 = 0$. Initially, it is imposed $\bar{V}_0 = V_0$ and $\bar{V}_i = \emptyset$ for all $i \in \{1, \cdots, n\}$. Then, for any $i \in \{1, \cdots, n\}$, the algorithm computes the λ-estimation from a state $(\bar{x}, \bar{z}) \in \bar{V}_{i-1}$ within $t_i - t_{i-1}$ implying the discrete states unobservably $(t_i - t_{i-1})$-reachable from $\bar{x}$ with a clock value in $\bar{z}$, and the set $\bar{V}_i$ is updated with the states reached by transitions labeled with e_{oi} from the states in V_λ. After the set $\bar{V}_n$ is determined, we initialize V_λ to be empty and update V_λ by including the λ-estimation for each $(\bar{x}, \bar{z}) \in \bar{V}_n$ within $t - t_n$. Eventually, we return the set of discrete states of G associated with V_λ as the set of discrete states consistent with (σ_o, t).

The complexity of Algorithm 8 depends on the size n of the timed observation. For each pair (e_{oi}, t_i), two *for* loops are executed.

- The first *for* loop at Step 4 is executed at most $|V|$ times, calling Algorithm 7 whose complexity is $\mathcal{O}(q^3|X|)$. Thus the complexity of this loop is $\mathcal{O}(q^4|X|^2)$.

- The second *for* loop at Step 8 is executed at most $|V|$ times, and the *for* loop at Step 11 is executed at most $2q + 1$ times; hence its complexity is $\mathcal{O}(q^2|X|)$.

Finally, the *for* loop at Step 18, analogously to the *for* loop at Step 4, has complexity $\mathcal{O}(q^4|X|^2)$. Overall, the complexity of Algorithm 8 is $\mathcal{O}(n(q^4|X|^2 + q^2|X|) + q^4|X|^2) = \mathcal{O}(nq^4|X|^2)$.

Example 4.8 Consider the partially observed GTFA $G = (X, E, \Delta, \Gamma, Reset, X_0)$ in Fig. 4.1 with $E_o = \{a\}$, $E_{uo} = \{b, c\}$ and a timed observation $(\sigma_o, 4)$, where $\sigma_o = (a, 1)(a, 3)$. It implies that the observable event a has been measured twice at $t_1 = 1$ and $t_2 = 3$, respectively, while the current time instant is $t = 4$.

Table 4.5 summarizes the state estimation of the GTFA G in Fig. 4.1 given the timed observation $((a, 1)(a, 3), 4)$ according to Algorithm 8. It additionally shows how the state

Algorithm 8 State estimation of a GTFA

Input: A GTFA G with a set of initial discrete states X_0, a set of observable events $E_o \subseteq E$, a zone automaton $G_z = (V, E_\tau, \Delta_z, V_0)$, and a timed observation (σ_o, t) from 0 to $t \in \mathbb{R}_{\geq 0}$, where $\sigma_o = (e_{o1}, t_1) \cdots (e_{on}, t_n)$ $(n \geq 1)$ and $t_1, \cdots, t_n \in \mathbb{R}_{\geq 0}$

Output: The set of states $\mathcal{X}(\sigma_o, t)$

let $\bar{V}_0 = V_0$, $t_0 = 0$ and $X_\lambda = \emptyset$
for each $i \in \{1, \cdots, n\}$ **do**
  let $e = e_{oi}$, $\bar{V}_i = \emptyset$ and $V_\lambda = \emptyset$
  for each $(\bar{x}, \bar{z}) \in \bar{V}_{i-1}$ **do**
    compute $V_\lambda(\bar{x}, \bar{z}, t_i - t_{i-1})$ with Algorithm 7
    let $V_\lambda = V_\lambda \cup V_\lambda(\bar{x}, \bar{z}, t_i - t_{i-1})$
  end for
  for each $v \in V_\lambda$ **do**
    let $x = f_x(v)$ and $z = f_z(v)$
    if $\exists x' \in X$ s.t. $z \subseteq \Gamma((x, e, x'))$ **then**
      for each $z' \in Z(x')$ s.t. $z' \subseteq Reset((x, e, x'))$ **do**
        let $\bar{V}_i = \bar{V}_i \cup \{(x', z')\}$
      end for
    end if
  end for
end for
let $V_\lambda = \emptyset$
for each $(\bar{x}, \bar{z}) \in \bar{V}_n$ **do**
  compute $V_\lambda(\bar{x}, \bar{z}, t - t_n)$ with Algorithm 7 and let $V_\lambda = V_\lambda \cup V_\lambda(\bar{x}, \bar{z}, t - t_n)$
end for
return $\mathcal{X}(\sigma_o, t) = \{x \in X \mid (\exists z \in Z(x))(x, z) \in V_\lambda\}$.

estimation is updated while time elapses in the time interval $[0, 4]$ taking into account the two observations of event a. In particular, the state estimation process is progressively updated from $\bar{V}_0 = \{(x_0, [0, 0]), (x_2, [0, 0])\}$. We explain the process of state estimation while the observation (σ_o, t) is progressively updated over time as follows.

- Step $k = 1$: The GTFA G produces observation $(\sigma_o, t) = (\lambda, t)$ for $t \in [0, 1]$. We compute the set of unobservably 1-reachable discrete states from $\bar{X}_0$ as $\{x_0, x_1, x_2, x_3\}$, according to $\bigcup_{v \in \bar{V}_0} V_\lambda(f_x(v), f_z(v), 1) = \{(x_0, [1, 1]), (x_1, [1, 1]), (x_2, [1, 1]), (x_3, [1, 1])\}$. After the pair $(a, 1)$ is received, Algorithm 8 provides the state estimates as $\bar{V}_1 = \{(x_2, [0, 0]), (x_4, [0, 1])\}$, according to the two elements $(x_1, [1, 1])$ and $(x_3, [1, 1])$ in $\bigcup_{v \in \bar{V}_0} V_\lambda(f_x(v), f_z(v), 1)$, which imply $((x_1, [1, 1]), a, (x_4, [0, 1])) \in \Delta_z$ and $((x_3, [1, 1]), a, (x_2, [0, 0])) \in \Delta_z$.

- Step $k = 2$: The GTFA G produces observation $(\sigma_o, t) = ((a, 1), t)$ for $t \in [1, 3]$. We compute the state estimates from $\bar{V}_1$ within 2 as $\{x_2, x_3\}$, according to $\bigcup_{v \in \bar{V}_1} V_\lambda(f_x(v),$

Table 4.5 State estimation of the GTFA G in Fig. 4.1 with $\bar{X}_0 = X_0$, $t_0 = 0$ and $(\sigma_o, t), t \in [0, 4]$.

k	σ_o	Time interval I ($t \in I$)	$V_\lambda = \bigcup_{v \in \bar{V}_{k-1}} V_\lambda(f_x(v), f_z(v), t - t_{k-1})$	$\mathcal{X}(\sigma_o, t)$	$\bar{V}_k$
1	λ	[0,0]	$\{(x_0, [0,0]), (x_2, [0,0])\}$	$\{x_0, x_2\}$	$\{(x_2, [0,0]), (x_4, [0,1])\}$
		(0,1)	$\{(x_0, (0,1)), (x_2, (0,1))\}$	$\{x_0, x_2\}$	
		[1,1]	$\{(x_0, [1,1]), (x_1, [1,1]), (x_2, [1,1]), (x_3, [1,1])\}$	$\{x_0, x_1, x_2, x_3\}$	
2	$(a, 1)$	[1,1]	$\{(x_2, [0,0]), (x_3, [0,0]), (x_4, [0,1])\}$	$\{x_2, x_3, x_4\}$	$\{(x_2, [0,0])\}$
		(1,2)	$\{(x_2, (0,1)), (x_3, [0,0]), (x_3, (0,1)), (x_4, [0,1])\}$	$\{x_2, x_3, x_4\}$	
		[2,2]	$\{(x_2, [1,1]), (x_3, [0,0]), (x_3, (0,1)), (x_3, [1,1]), (x_4, [0,1])\}$	$\{x_2, x_3, x_4\}$	
		(2,3)	$\{(x_2, (1,2)), (x_3, (0,1)), (x_3, [1,1]), (x_3, (1,2)), (x_4, (1, +\infty))\}$	$\{x_2, x_3, x_4\}$	
		[3,3]	$\{(x_2, [2,2]), (x_3, [1,1]), (x_3, (1,2)), (x_3, [2,2]), (x_4, (1, +\infty))\}$	$\{x_2, x_3, x_4\}$	
3	$(a, 1)(a, 3)$	[3,3]	$\{(x_2, [0,0])\}$	$\{x_2\}$	-
		(3,4)	$\{(x_2, (0,1))\}$	$\{x_2\}$	
		[4,4]	$\{(x_2, [1,1]), (x_3, [1,1])\}$	$\{x_2, x_3\}$	

$f_z(v), 2) = \{(x_2, [2,2]), (x_3, [1,1]), (x_3, (1,2)), (x_3, [2,2]), (x_4, (1, +\infty))\}$. After the pair $(a, 3)$ is received, Algorithm 8 provides the state estimation as $\bar{V}_2 = \{(x_2, [0,0])\}$, according to $((x_3, [1,1]), a, (x_2, [0,0])) \in \Delta_z$.

- Step $k = 3$: The GTFA G produces observation $(\sigma_o, t) = ((a, 1)(a, 3), t), t \in [3, 4]$. We compute the set of unobservably 1-reachable discrete states from $\bar{X}_2$ as $\{x_2, x_3\}$, according to $\bigcup_{v \in \bar{V}_2} V_\lambda(f_x(v), f_z(v), 1) = \{(x_2, [1,1]), (x_3, [1,1])\}$. No more pair is received and the observation ends with $(\sigma_o, t) = ((a, 1)(a, 3), 4)$. Algorithm 8 provides the set of discrete states consistent with $(\sigma_o, t) = ((a, 1)(a, 3), 4)$ that is equal to $\mathcal{X}(\sigma_o, t) = \{x_2, x_3\}$. □

4.4 Application to State Estimation of Timed Discrete Event Systems under multiple clocks

This section presents a short discussion on state estimation of timed DES with multiple clocks, which can be explored by extending the state estimation approach of GTFA with a single clock. We first introduce the following definition to model timed DES with multiple clocks as GTFA with a set of clocks.

Definition 4.12 A *GTFA with $k \geq 1$ clocks* is a six-tuple $G = (X, E, \Delta, \Gamma, Reset, X_0)$ that operates under k clocks $\theta_1, \cdots, \theta_k$, where

- X, E, Δ and X_0 refer to the definition of GTFA,

- $\Gamma : \Delta \to \mathbb{I}_c^k$ is a timing function,
- $Reset : \Delta \to \{\mathbb{I}_c \cup \{id\}\}^k$ is a clock resetting function. □

A GTFA with $k \geq 1$ clocks generalizes the timing function Γ (resp., the clock resetting function $Reset$). The firability of a transition $(x, e, x') \in \Delta$ and range of the clock value at which the transition fires may involve k clocks $\theta_1, \cdots, \theta_k$. To analyse the dynamics of the GTFA G operates under each single clock $\theta_i (1 = 1, \cdots, k)$, a GTFA G_{θ_i} can be obtained by retaining the timing structure provided by θ_i and eliminating that of other $k-1$ clocks. After that, the zone automaton of G_{θ_i} can be constructed. According to Theorem 4.1 and Theorem 4.2, the reachability of G_{θ_i} can be analysed by exploring runs in its zone automaton.

Example 4.9 Consider a system that triggers an alarm after a problem occurred at a certain time and repairs the problem in a while. Its behavior is described by a partially observed GTFA $G' = (X, E, \Delta, \Gamma, Reset, X_0)$ with two clocks θ_1 and θ_2, where $X = \{x_0, x_1, x_2\}$, $X_0 = \{x_0\}$, $E = \{a, b, c, d\}$, and the set of observable events is $E_o = \{b, d\}$. The clock θ_1 is to record and control the elapsed time taken to repair the occurred problem, and θ_2 is to record and control the time dwelling at each discrete state. The discrete states x_0, x_1, x_2, and x_3 accordingly denote a safe state, an alarm state, a repairing state, and a failsafe state. In addition, each event in E has a specific physical meaning: a means that a problem occurs, b means that repair is processing, c means a delay, d means that the problem is repaired. Let Δ, Γ and $Reset$ be defined as in Table 4.6. The graphical representation of G' is visualized in Fig. 4.3.

According to Table 4.6, we can obtain the GTFA G'_{θ_1} and G'_{θ_2}, depicted in Fig. 4.4(a) and Fig. 4.4(b), that operate under a single clock θ_1 and a single clock θ_2, respectively. The zone automaton of G'_{θ_1} (resp., G'_{θ_2}) is reported in Fig. 4.4(c) (resp., Fig. 4.4(d)). □

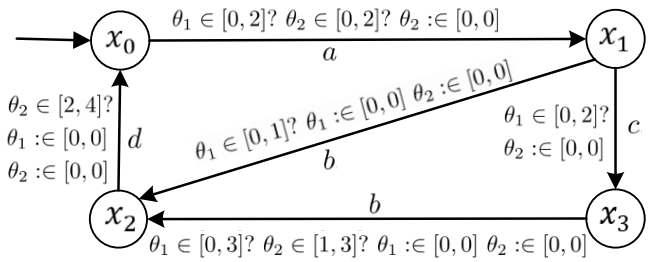
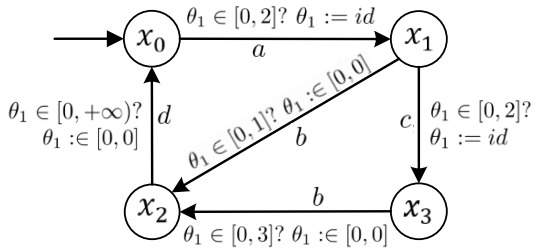
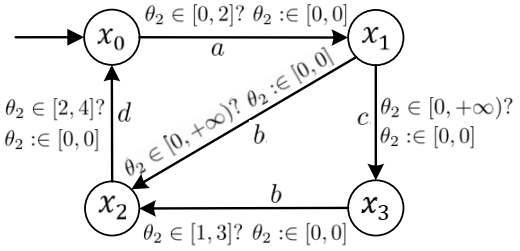


Fig. 4.3 GTFA G' with two clocks θ_1 and θ_2.

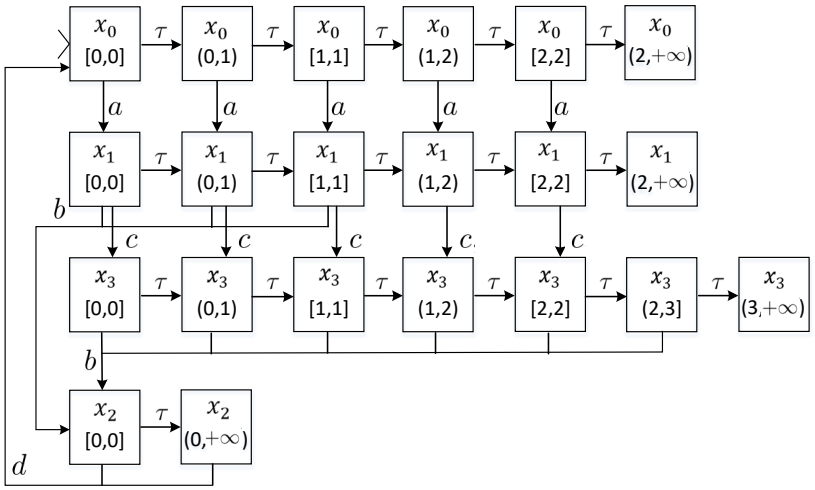
As a future work, the state estimation of GTFA with multiple clocks $\theta_1, \cdots, \theta_k$ can be explored by constructing the concurrent composition of $G_{\theta_1}, \cdots, G_{\theta_k}$ such that



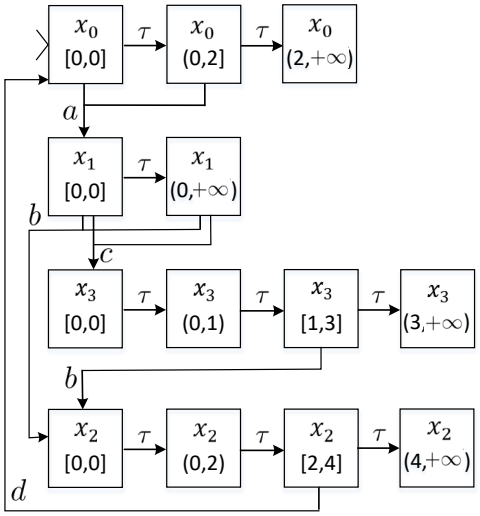
(a) GTFA G'_{θ_1}.



(b) GTFA G'_{θ_2}.



(c) Zone automaton of G'_{θ_1}.



(d) Zone automaton of G'_{θ_2}.

Fig. 4.4 GTFA G' in Fig. 4.3 and its zone automaton with each single clock.

Table 4.6 Transition relation Δ, timing function Γ and the clock resetting function $Reset$ of the GTFA G' in Fig. 4.3.

$\delta \in \Delta$	$\Gamma(\delta)$	$Reset(\delta)$
(x_0, a, x_1)	$\theta_1 : [0, 2]$ $\theta_2 : [0, 2]$	$\theta_1 : id$ $\theta_2 : [0, 0]$
(x_1, b, x_2)	$\theta_1 : [0, 1]$ $\theta_2 : [0, +\infty)$	$\theta_1 : [0, 0]$ $\theta_2 : [0, 0]$
(x_1, c, x_3)	$\theta_1 : [0, 2]$ $\theta_2 : [0, +\infty)$	$\theta_1 : id$ $\theta_2 : [0, 0]$
(x_3, b, x_2)	$\theta_1 : [0, 3]$ $\theta_2 : [1, 3]$	$\theta_1 : [0, 0]$ $\theta_2 : [0, 0]$
(x_2, d, x_0)	$\theta_1 : [0, +\infty)$ $\theta_2 : [2, 4]$	$\theta_1 : [0, 0]$ $\theta_2 : [0, 0]$

- events associated to more than one clock must be synchronized, and
- each state is characterized by k time intervals (one for each clock).

Given a timed observation (σ_o, t), the state estimation of G with multiple clocks $\theta_1, \cdots, \theta_k$ can be realized by analysing the reachability of the composed automaton.

4.5 Conclusions

In this section, we summarize the conclusions of this chapter as follows:

- We present a model of timed DES, called *generalized timed finite automata (GTFA)*, characterized by a single clock. In contrast to the TFA, the clock of the GTFA does not have to be reset to 0 each time an event occurs. Particularly, a clock resetting function is associated with a GTFA to denote whether the clock is reset to a value in a given closed time interval. Consequently, the time semantics now permits a system to remain in a discrete state forever. The timed structure of a GTFA specifies the set of clock values that allow an event to occur and how the clock is reset upon the event occurrence.
- We consider partially observed GTFA that produces timed observations as a succession of pairs of an observable event and the time instant at which the event has occurred. Our objective is to estimate the current discrete state of the automaton as a function of the current observation and the current time. When dealing with the state estimation problem, we assume that observable transitions should be reset to zero to

guarantee the applicability of the proposed state estimation approach, which is based on the notion of zones.

- The state estimation problem is solved by considering two sub-problems: state estimation with no observation and state estimation with partial observation. The solution is based on determining the T-reachability of a GTFA, which takes into account the discrete states that can be reached with an evolution producing a given observation and a duration equal to T. The problem of T-reachability in the GTFA is reduced to the reachability analysis of the associated zone automaton.

- We present a discussion on the state estimation of timed DES with multiple clocks, which can be explored by extending the state estimation approach of GTFA with a single clock. We model a multi-clock timed DES as a GTFA with multiple clocks, which generalizes the timing function and the clock resetting function to multiple clocks. Given a GTFA G with $k \geq 1$ multiple clocks, the dynamics under each clock can be analyzed by k GTFA $G_{\theta_i}(i = 1, \cdots, k)$ with the single i-th clock. The zone automaton associated with G_{θ_i} can be constructed to explore the reachability of G under the i-th clock. In future work, the state estimation of GTFA G with multiple clocks can be explored by exploring the concurrent composition of the GTFA $G_{\theta_1}, \cdots, G_{\theta_k}$.

Chapter 5 A Cyber Security Problem: Attack Detection

Cyber-physical systems have emerged as a key technology in developing distributed and autonomous large-scale systems. However, one of their undesirable side effects is that they are exposed to cyber attacks carried out by malicious intruders. Therefore, efficient strategies for cyber security are in high demand.

In this chapter, we consider a plant modelled as a DES that is partially observed through a communication channel by an operator which monitors its evolution, as shown in Fig. 5.1. After the plant generates a production, the mask projects the production to an observation. The occurrence of unobservable events produces no observable symbols such that only observable events can be measured. In the absence of an attack, the operator receives the observation through a communication channel. However, the communication channel may be subject to cyber attacks which corrupt the observation by replacing, erasing and inserting symbols and lead to false state estimation of the operator.

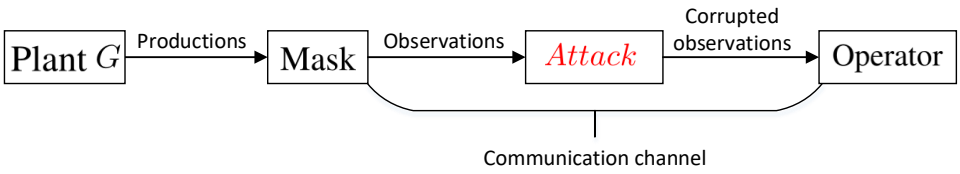


Fig. 5.1 Plant under attack.

Inspired by the attack model of [96], we assume that the observation produced by a plant can be corrupted by an intruder, which can change an observable event into different strings of observable events through one or more *attack dictionaries* defined as follows.

Definition 5.1 An *attack dictionary* on a plant Q is a set-valued function $A : E_o \longrightarrow 2^{E_o^*} \setminus \{\emptyset\}$, which maps an observable event into a non-empty set of strings. □

An attack dictionary on a plant Q describes all possible ways in which an attacker can corrupt an observable event. In this chapter, we assume that the observation produced by a plant can be corrupted by an intruder which, through one or more *attack dictionaries*, can change events into different strings. In detail, a system may be subject to multiple types of attacks, each described by its own attack dictionary. Furthermore, we distinguish between *constant attacks*, which corrupt observations by using only one of the attack dictionaries, and *switching attacks*, which may use different attack dictionaries at different steps. Given a

system subject to attacks, suppose that one knows the possible dictionaries that the intruder may use to corrupt the observation. Our goal is to provide a method for detecting whether an attack has occurred and also the nature of the attack, i.e. which dictionaries were used by the attacker.

We divide this chapter into five sections. The previous three sections deal with the attack detection problem in the framework of untimed DES, modeled by DFA. In more detail, the first section models how constant and switching attacks corrupt the observations produced by DFA using an attack dictionary. The second and third sections deal with the detection of constant and switching attacks, respectively. In both cases, we compute an NFA that describes the observations produced by the original system under attack: in particular, different structures correspond to a system under constant and switching attacks. We show that the detection of a constant (or switching) attack can be reduced to a classical state estimation (or fault diagnosis) problem for the obtained NFA.

Consider the scenario where a system is embedded with a timing structure that is hidden from an intruder. The cyber security of the system can benefit from the timing information. This motivates us to discuss the aforementioned attacks on timed DES modelled as TFA. The fourth section presents a discussion of this future work. Finally, the fifth section concludes this chapter.

5.1 Models of Attacks on Deterministic Finite Automata

This section first provides the model of attacks and then introduces the notions of constant and switching attacks, respectively.

Definition 5.2 Given an attack dictionary $A : E_o \longrightarrow 2^{E_o^*}$, the set of *attacks* over A is defined by

$$\mathcal{F}_A = \{f_A : E_o^* \longrightarrow E_o^* | (\forall w \in E_o^*)(\forall e \in E_o) f_A(\varepsilon) = \varepsilon, f_A(we) \in f_A(w)A(e)\}.$$

□

In other words, an attack is a function such that every event $e \in E_o$ in observation w is mapped into a string in $A(e)$. We notice that, as a special case, $A(e)$ may contain e and this attack may not corrupt the observation. Note that if two attack dictionaries A' and A are such that for all $e \in E_o$, it holds that $A(e) \subseteq A'(e)$, then $\mathcal{F}_A \subseteq \mathcal{F}_{A'}$.

Example 5.1 Given attack dictionaries A and A' with $E = \{a, b, c\}$ and $E_o = \{a, c\}$, where $A(a) = \{a, \varepsilon\}, A(c) = \{a\}$, $A'(a) = \{a, \varepsilon\}$ and $A'(c) = \{a, c\}$, consider attacks $f_1 \in \mathcal{F}_A$

and $f_2 \in \mathcal{F}_{A'}$ such that for all $w \in E_o^*$,

$$f_1(\varepsilon) = \varepsilon;\ f_1(wc) = f_1(w)a;$$

$$f_1(wa) = \begin{cases} f_1(w)a & \text{if } |w|_a \text{ is even} \\ f_1(w) & \text{if } |w|_a \text{ is odd;} \end{cases}$$

$$f_2(\varepsilon) = \varepsilon;\ f_2(wa) = f_2(w);$$

$$f_2(wc) = \begin{cases} f_2(w)a & \text{if } |w|_c \text{ is even} \\ f_2(w)c & \text{if } |w|_c \text{ is odd.} \end{cases}$$

The attacks f_1 and f_2 satisfy Definition 5.2 according to A and A' separately. The observable events a and c are always mapped to events in $A(a)$ and $A(c)$ by f_1, respectively. In any observation corrupted by f_2, a and c are replaced by the elements in $A'(a)$ and $A'(c)$ respectively. □

The notion of cyber attacks can be generalized by assuming that a system may be subject to multiple types of attacks, each of which is described by its own attack dictionary. To keep into account the case of no attack, a new attack dictionary A_0 is introduced in $\mathcal{A}$. For all $e \in E_0$, it holds that $A_0(e) = \{e\}$. As a result, the set of attack dictionaries is updated to be $\mathcal{A} = \{A_0, A_1, \cdots, A_n\}$.

Definition 5.3 Given a set of attack dictionaries $\mathcal{A} = \{A_0, \cdots, A_n\}$ with $A_i : E_o \longrightarrow 2^{E_o^*} \setminus \{\emptyset\}$, where $i \in \{0, \cdots, n\}$, the set of constant attacks over $\mathcal{A}$ is defined by

$$\mathcal{F}_{\mathcal{A}}^c = \{f_{\mathcal{A}} : E_o^* \longrightarrow E_o^* | (\exists i \in \{0, \cdots, n\})\ f_{\mathcal{A}} \in \mathcal{F}_{A_i}\}.$$

□

Being subject to a constant attack over $\mathcal{A} = \{A_0, \cdots, A_n\}$, the system could only be corrupted by one of the attack dictionaries in $\mathcal{A}$, which implies that $f_{\mathcal{A}} \in \bigcup_{A_i \in \mathcal{A}} \mathcal{F}_{A_i}$ always holds for an attack function in the set of constant attacks.

Definition 5.4 Given a set of attack dictionaries $\mathcal{A} = \{A_1, \cdots, A_n\}$ with $A_i : E_o \longrightarrow 2^{E_o^*}$, $i \in \{0, \cdots, n\}$, the set of switching attacks over $\mathcal{A}$ is defined by

$$\mathcal{F}_{\mathcal{A}}^s = \{f_{\mathcal{A}} : E_o^* \longrightarrow E_o^* | (\forall w \in E_o^*)(\forall e \in E_o)(\exists i \in \{0, \cdots, n\})$$
$$f_{\mathcal{A}}(\varepsilon) = \varepsilon, f_{\mathcal{A}}(we) \in f_{\mathcal{A}}(w)A_i(e)\}.$$

□

In other words, if a system is under a switching attack over $\mathcal{A} = \{A_0, \cdots, A_n\}$, the same observable event at different steps could be attacked using different attack dictionaries

among $A_0, \cdots, A_n$. Obviously, a constant attack can be seen as a special case of a switching attack where always the same attack dictionary is selected. As a consequence, for any set $\mathcal{A}$, it holds that $\mathcal{F}^c_{\mathcal{A}} \subseteq \mathcal{F}^s_{\mathcal{A}}$.

Example 5.2 Given a set of attack dictionaries $\mathcal{A} = \{A_0, A_1, A_2\}$, where $A_0(a) = \{a\}$, $A_0(c) = \{c\}$, $A_1(a) = \{\varepsilon\}$, $A_1(c) = \{a\}$, $A_2(a) = \{c\}$ and $A_2(c) = \{ac, c\}$, consider the attacks $f_1 \in \mathcal{F}^c_{\mathcal{A}}$ and $f_2 \in \mathcal{F}^s_{\mathcal{A}}$ such that for all $w \in E^*_o$,

$$f_1(\varepsilon) = \varepsilon;\ f_1(wa) \in f_1(w)A_1(a);\ f_1(wc) \in f_1(w)A_1(c);$$

$$f_2(\varepsilon) = \varepsilon;\ f_2(wa) \in f_2(w)A_1(a);$$
$$f_2(wc) \in \begin{cases} f_2(w)A_1(c) & \text{if } |w|_c \text{ is even} \\ f_2(w)A_2(c) & \text{if } |w|_c \text{ is odd.} \end{cases}$$

Now we consider the plant Q in Fig. 5.2 with $E = \{a, b, c\}$ and $E_o = \{a, c\}$ which may be subject to f_1 or f_2. The corruption of f_1 and f_2 to the observations of Q is shown in Table 5.1. Attack f_1 always corrupts the events in the observation with the attack dictionary A_1. On the contrary, f_2 either uses the attack dictionary A_1 or the attack dictionary A_2. □

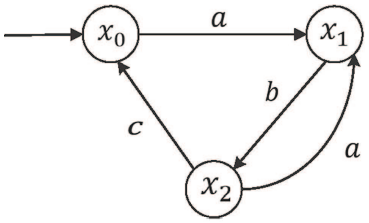


Fig. 5.2 Automaton Q.

Table 5.1 Effect of the constant and switching attack in Example 5.2.

$s \in L(Q)$	$w = P(s)$	$f_1(w)$	$f_2(w)$
ε	ε	ε	ε
a, ab	a	ε	ε
$aba, abab$	aa	ε	ε
abc	ac	a	a
$ababa, ababab$	aaa	ε	ε
$abca, abcab$	aca	a	a
$abcabc$	$acac$	aa	aac
$abababa, abababab$	$aaaa$	ε	ε
$\cdots$	$\cdots$	$\cdots$	$\cdots$

5.2 Constant Attacks Detection on Deterministic Finite Automata

This section deals with detection of constant attacks. The solution is based on constructing an NFA that describes the observations generated by the DFA under constant attacks. We show that the detection of a constant attack can be reduced to the state estimation problem for the obtained NFA.

In the case of a constant attack, Algorithm 9 shows how to construct the NFA generating the corrupted observations. A new set of events is introduced, denoted as $E_a = \{a_0, ..., a_n\}$. The occurrence of event a_i, $i \in \{1, ..., n\}$, corresponds to the fact that the attack A_i is active. The occurrence of a_0 corresponds to no attack on the system, or equivalently to attack A_0.

Algorithm 9 works as follows. We first initialize the set of states with the initial state $\hat{x}_0$, and define the set of attack labels and the observation partition of $Q^c_{\mathcal{A}}$. Then for each attack dictionary A_i, we define a new set of states $\{x^i_0, ..., x^i_m\}$ and the transition function $\hat{\delta}$ is updated as $\hat{\delta}(\hat{x}_0, a_i) = x^i_0$. Then, the transition function $\hat{\delta}$ is updated. In particular, for each $k \in \{0, ..., m\}$ and for each observable event that is enabled at x_k, if $x'_k = \delta(x_k, e)$, then for all $w \in A_i(e)$, it is $\hat{\delta}(x^i_k, w) = x^i_{k'}$.

Example 5.3 Given the plant in Fig. 5.2, suppose that Q may be attacked using the set of attack dictionaries $\mathcal{A} = \{A_0, A_1, A_2\}$ of the previous example. The model of Q under a constant attack generated by Algorithm 9 is shown in Fig. 5.3. □

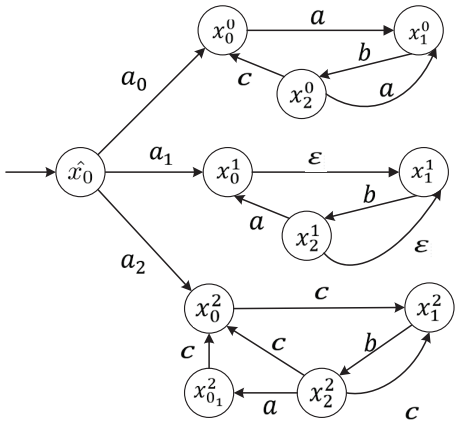


Fig. 5.3 NFA $Q^c_{\mathcal{A}}$ resulting from Algorithm 9 when Q is the DFA in Fig. 5.2.

Theorem 5.1 Given a DFA $Q = (X, E, \delta, x_o)$ with a mask $P : E \longrightarrow E_o$ and a set of attack dictionaries $\mathcal{A} = \{A_0, \cdots, A_n\}$, let $Q^c_{\mathcal{A}} = (\hat{X}, \hat{E}, \hat{\delta}, \hat{x}_0)$ be the NFA constructed

Algorithm 9 Construction of the NFA generating the corrupted observations under a constant attack

Input: A DFA $Q = (X, E, \delta, x_0)$ with set of states $X = \{x_0, \cdots, x_m\}$ and observation partition $E = E_o \cup E_{uo}$; a set of attack dictionaries $\mathcal{A} = \{A_0, A_1, \cdots, A_n\}$

Output: An NFA $Q^c_{\mathcal{A}} = (\hat{X}, \hat{E}, \hat{\delta}, \hat{x}_0)$ with observation partition $\hat{E} = \hat{E}_o \cup \hat{E}_{uo}$

let $\hat{X} = \{\hat{x}_0\}$
let $E_a = \{a_0, \cdots, a_n\}$
let $\hat{E}_o = E_o$, $\hat{E}_{uo} = E_{uo} \cup E_a$, and $\hat{E} = E \cup E_a$
for all $i \in \{0, \cdots, n\}$ **do**
  let $\hat{X} = \hat{X} \cup \{x^i_0, \cdots, x^i_m\}$
  let $\hat{\delta}(\hat{x}_0, a_i) = x^i_0$
  for all $k \in \{0, \cdots, m\}$ **do**
    for all $e \in E_o \cap enab(x_k)$ **do**
      let $x_{k'} = \delta(x_k, e)$
      for all $w = e_1 e_2 \cdots e_p \in A_i(e), p \geq 1$ **do**
        if $p = 1$ **then**
          let $\hat{\delta}(x^i_k, w) = x^i_{k'}$
        end if
        if $p > 1$ **then**
          let $\hat{\delta}(x^i_k, e_1) := x^i_{k'_1}$, $\hat{\delta}(x^i_{k'_1}, e_2) := x^i_{k'_2}$, $\cdots$, and $\hat{\delta}(x^i_{k'_{p-1}}, e_p) := x^i_{k'}$
        end if
      end for
    end for
  end for
end for
return $Q^c_{\mathcal{A}} = (\hat{X}, \hat{E}, \hat{\delta}, \hat{x}_0)$.

using Algorithm 9. It holds that $\hat{P}(L(Q^c_{\mathcal{A}})) = \bigcup_{f \in F^c_{\mathcal{A}}} f(P(L(Q)))$, where $\hat{P} : \hat{E}^* \longrightarrow \hat{E}^*_o$ is the projection over alphabet $\hat{E}$. □

Proof: The inclusion relationship $\hat{P}(L(Q^c_{\mathcal{A}})) \subseteq \bigcup_{f \in F^c_{\mathcal{A}}} f(P(L(Q)))$ trivially follows from the way the NFA Q^c_A is constructed according to Algorithm 9. To prove $\bigcup_{f \in F^c_{\mathcal{A}}} f(P(L(Q))) \subseteq \hat{P}(L(Q^c_{\mathcal{A}}))$, we consider a string $s \in L(Q)$ and the observation $w = P(s) = e_1 e_2 \cdots e_p, p \geq 1$. Let $\hat{w} = w_1 w_2 \cdots w_p$ be a corrupted observation of w resulting from a certain dictionary in $\mathcal{A}$, namely let $\hat{w} \in \bigcup_{f \in F^c_{\mathcal{A}}} f(P(L(Q)))$. By $\mathcal{F}^c_{\mathcal{A}} = \bigcup_{i=0}^{n} \mathcal{F}_{A_i}$, there exists $i \in \{0, \cdots, n\}$ such that $w_1 = A_i(e_1), \cdots, w_p = A_i(e_p)$. According to Algorithm 9, for all e such that $\delta(x_k, e) = x'_k$ where $x_k, x'_k \in X$, there exists $i \in \{0, \cdots, n\}$ such that $\hat{\delta}(x^i_k, w) = x^i_{k'}$ has been defined in $Q^c_{\mathcal{A}}$, where $x^i_k, x^i_{k'} \in \hat{X}$ and $w \in A_i(e)$. By $\hat{E}_o = E_o$ and $\hat{E}_{uo} = E_{uo} \cup E_a$, it follows that $\hat{w} \in \hat{P}(L(Q^c_{\mathcal{A}}))$. This implies that $\bigcup_{f \in F^c_{\mathcal{A}}} f(P(L(Q))) \subseteq \hat{P}(L(Q^c_{\mathcal{A}}))$. □

Theorem 5.1 provides a characterization of the language of the NFA Q^c_A. In simple

words, the observable projection of the language generated by Q_A^c is equal to the union of all the possible corrupted words that can be generated with a constant attack in $\mathcal{A}$ acting on Q.

Once an NFA $Q_{\mathcal{A}}^c = (\hat{X}, \hat{E}, \hat{\delta}, \hat{x}_0)$ is built according to Algorithm 9, one can use classical notions of state estimation to detect the attack. Suppose that a DFA Q with set of states $X = \{x_0, \cdots, x_m\}$ is attacked using a set of attack dictionaries $\mathcal{A} = \{A_0, A_1, \cdots, A_n\}$. The state reached by the observer of $Q_{\mathcal{A}}^c$ executing an observation w represents the set $C(w)$ of states of Q consistent with observation w (see Definition 2.10).

Thus let $X_i = \{x_k^i \in \hat{X} | k \in \{1, \cdots, m\}\}$ for all $i \in \{0, \cdots, n\}$. Given an observation w we can conclude that:

- the system is not under attack if $C(w) \subseteq X_0$;
- the system is under attack if $X_0 \cap C(w) = \emptyset$;
- the system is under the attack with dictionary $A_i (i > 0)$ if $C(w) \subseteq X_i$;
- the system is not under the attack with dictionary $A_i (i > 0)$ if $X_i \cap C(w) = \emptyset$.

Example 5.4 Consider again the plant Q in Fig. 5.2 with $E = \{a, b, c\}$ and $E_o = \{a, c\}$ under attack as described in Example 5.2. Consider the NFA $Q_{\mathcal{A}}^c$ in Fig. 5.3. The observer of $Q_{\mathcal{A}}^c$ can be obtained as shown in Fig. 5.4. For instance, if the received observation is cc, we can conclude that a constant attack of A_2 happens; if the observation is aca, no attack happens; if the observation is $aaaa$, a constant attack of A_1 could happen. □

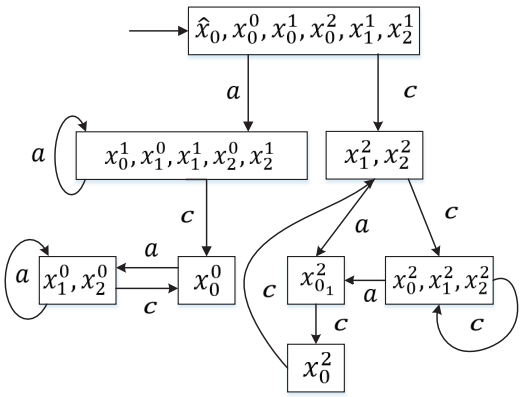


Fig. 5.4 Observer of $Q_{\mathcal{A}}^c$.

5.3 Switching Attacks Detection on Deterministic Finite Automata

Let us now focus on the case of switching attacks in this section. We first provide Algorithm 10 to construct an NFA generating a language whose observable projection is equal to the union of all the possible corrupted words that can be generated with a switching attack in $\mathcal{A}$ acting on a certain DFA Q. After that, we show that the detection of a switching attack can be reduced to the fault diagnosis problem of the obtained NFA.

Algorithm 10 Construction of the NFA generating the corrupted observations under a switching attack

Input: A DFA $Q = (X, E, \delta, x_0)$ with set of states $X = \{x_0, \cdots, x_m\}$ and observation partition $E = E_o \cup E_{uo}$; a set of attack dictionaries $\mathcal{A} = \{A_1, \cdots, A_n\}$

Output: An NFA $Q^s_{\mathcal{A}} = (\hat{X}, \hat{E}, \hat{\delta}, x_0)$ with observation partition $\hat{E} = \hat{E}_o \cup \hat{E}_{uo}$; a set of attack labels $E_a = \{a_1, \cdots, a_n\}$

1: let $\hat{X} = X$, and $\hat{\delta} = \delta$
2: let $E_a = \{a_1, \cdots, a_n\}$
3: let $\hat{E}_o = E_o$, $\hat{E}_{uo} = E_{uo} \cup E_a$, and $\hat{E} = E \cup E_a$
4: **for all** $k \in \{0, \cdots, m\}$ **do**
5: **for all** $e \in E_o \cap enab(x_k)$ **do**
6: **for all** $i = \{1, \cdots, n\}$ **do**
7: **if** $A_i(e) \neq \{\varepsilon\}$ **then**
8: $\hat{X} = \hat{X} \cup \{x^i_k\}$ and $\hat{\delta}(x_k, a_i) = x^i_k$
9: **for all** $w = e_1 e_2 \cdots e_p \in A_i(e), p \geq 1$ **do**
10: **if** $p = 1$ **then**
11: let $\hat{\delta}(x^i_k, w) = \delta(x_k, e)$
12: **end if**
13: **if** $p > 1$ **then**
14: $\hat{\delta}(x^i_k, e_1) := x^i_{k_1}$, $\hat{\delta}(x^i_{k_1}, e_2) := x^i_{k_2}$, $\cdots$, and $\hat{\delta}(x^i_{k_{p-1}}, e_p) := \delta(x_k, e)$
15: **end if**
16: **end for**
17: **end if**
18: **if** $A_i(e) = \{\varepsilon\}$ **then**
19: let $\hat{\delta}(x_k, a_i) = \delta(x_k, e)$
20: **end if**
21: **end for**
22: **end for**
23: **end for**

Algorithm 10 works as follows. We first initialize the set of states $\hat{X}$ as X and define the set of attack labels $E_a = \{a_0, ..., a_n\}$ with the observation partition $\hat{E} = \hat{E}_o \cup \hat{E}_{uo}$. Then we explore all states $x_k \in X$ and all the events $e \in E_o$ that are enabled at x_k. We consider all possible attack dictionaries A_i that may occur on event e and distinguish two cases: (1) $A_i(e) \neq \varepsilon$ and (2) $A_i(e) = \varepsilon$. In Case (1), we add a new state to $\hat{X}$, denoted as x^i_k and let

$\hat{\delta}(x_k, a_i) = x_k^i$. Then for all $w \in A_i(e)$, we define $\hat{\delta}(x_k^i, w) = \delta(x_k, e)$. In Case (2), no new state is added to $\hat{X}$, but $\hat{\delta}$ is updated to $\hat{\delta}(x_k, a_i) = \delta(x_k, e)$.

Example 5.5 Continuing with plant Q and $\mathcal{A} = \{A_1, A_2\}$ in Example 5.2, the model of Q under a switching attack over $\mathcal{A}$ is generated as shown in Fig. 5.5 according to Algorithm 10. □

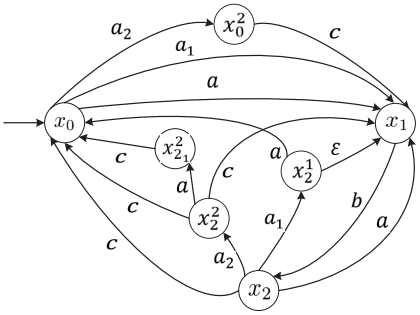


Fig. 5.5 NFA $Q_{\mathcal{A}}^s$ resulting from Algorithm 10 when Q is the DFA in Fig. 5.2.

The following theorem provides a characterization of the language of the NFA $Q_{\mathcal{A}}^s$. In simple words, the observable projection of the language generated by $Q_{\mathcal{A}}^s$ is equal to the union of all the possible corrupted words that can be generated with a switching attack in $\mathcal{A}$ acting on Q.

Theorem 5.2 Given a DFA $Q = (X, E, \delta, x_o)$ with a mask $P : E \longrightarrow E_o$ and a set of attack dictionaries $\mathcal{A} = \{A_0, \cdots, A_n\}$, let $Q_{\mathcal{A}}^s = (\hat{X}, \hat{E}, \hat{\delta}, x_0)$ be the NFA constructed using Algorithm 10. It holds that $\hat{P}(L(Q_{\mathcal{A}}^s)) = \bigcup_{f \in F_{\mathcal{A}}^s} f(P(L(Q)))$, where $\hat{P} : \hat{E}^* \longrightarrow \hat{E}_o^*$ is the projection over alphabet $\hat{E}$. □

Proof: We first prove that $\bigcup_{f \in F_{\mathcal{A}}^s} f(P(L(Q))) \subseteq \hat{P}(L(Q_{\mathcal{A}}^s))$. We prove this by induction on the length of the corrupted observation $\hat{w} \in \bigcup_{f \in F_{\mathcal{A}}^s} f(P(L(Q)))$. To prove the basis step, we consider a word $\hat{w}$ of null length, namely $\hat{w} = \varepsilon$. Clearly it is $\varepsilon \in \bigcup_{f \in F_{\mathcal{A}}^s} f(P(L(Q)))$ according to Definition 5.7 and $\varepsilon \in \hat{P}(L(Q_{\mathcal{A}}^s))$. To prove the induction step, assume that, if a word $\hat{w} \in \bigcup_{f \in F_{\mathcal{A}}^s} f(P(L(Q)))$, $|\hat{w}| \leq n$, then it also holds $\hat{w} \in \hat{P}(L(Q_{\mathcal{A}}^s))$. We show that this implies that, if for a certain $e \in E_o$, and a certain $i \in \{0, \cdots, n\}$, it is $\hat{w}A_i(e) \in \bigcup_{f \in F_{\mathcal{A}}^s} f(P(L(Q)))$, then $\hat{w}A_i(e) = f(w)A_i(e)$ for a certain $w \in P(L(Q))$ and $f(w)A_i(e) = f(w)\hat{P}(a_i A_i(e)) \in f(w) \cdot \hat{P}(L(Q_{\mathcal{A}}^s))$, which in turn implies that $\hat{w}A_i(e) \in \hat{P}(L(Q_{\mathcal{A}}^s))$. Thus $\bigcup_{f \in F_{\mathcal{A}}^s} f(P(L(Q))) \subseteq \hat{P}(L(Q_{\mathcal{A}}^s))$ holds.

To prove the reverse inclusion, namely $\hat{P}(L(Q^s_{\mathcal{A}})) \subseteq \bigcup_{f \in F^s_{\mathcal{A}}} f(P(L(Q)))$, we notice that, by Algorithm 10, it is $L(Q^s_{\mathcal{A}}) = \{\hat{s} \in E^* : \hat{\delta}(x_0, \hat{s}) \text{ is defined}\}$, let us consider an observation $\hat{w}w' \in \hat{P}(L(Q^s_{\mathcal{A}}))$. It is obvious that for all $i \in \{0, \cdots, n\}$, there exists $A_i(e) = w'$ such that $\hat{w}w' = f(w)A_i(e) \in \bigcup_{f \in F^s_{\mathcal{A}}} f(P(L(Q)))$. Thus $\hat{P}(L(Q^s_{\mathcal{A}})) \subseteq \bigcup_{f \in F^s_{\mathcal{A}}} f(P(L(Q)))$. As a result, $P(L(Q_{\mathcal{A}})) = \bigcup_{f \in F^s_{\mathcal{A}}} f(P(L(Q)))$ holds. □

In the case of switching attacks, we can perform attack detection constructing suitable diagnosers for $Q^s_{\mathcal{A}} = (\hat{X}, \hat{E}, \hat{\delta}, x_0)$. The attack labels $E_a = \{a_1, \ldots, a_n\}$ can be seen as the set of fault events whose occurrence should be reconstructed based on the corrupted observations. Assuming that each attack label belongs to a different fault class, one can construct n diagnosers $Diag_i$ (for $i = 1, \ldots, n$) each one devoted to detecting the occurrence of a_i. By considering all attack labels in the same fault class, one can construct a global diagnoser, whose purpose it that of detecting if the system is under attack.

Algorithm 11 shows how these diagnosers can be constructed following the approach described in [3]. We first build a label automaton for each attack label and then construct a diagnoser for each attack label by building an observer for the parallel composition between $Q^s_{\mathcal{A}}$ and each $Q_{label(i)}$. A global diagnoser can be similarly constructed.

Algorithm 11 Construction of diagnoser for switching attacks

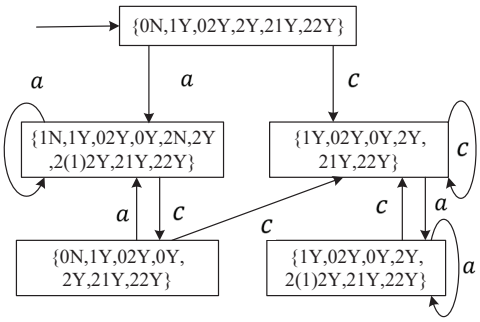
Input: An NFA $Q^s_{\mathcal{A}} = (\hat{X}, \hat{E}, \hat{\delta}, x_0)$ with $\mathcal{A} = \{A_1, \cdots, A_n\}$, set of fault events $E_a = \{a_1, \ldots, a_n\}$

Output: Diagnosers $Diag_i(Q^s_{\mathcal{A}}) = (Y_i, E_o, \delta_{y,i}, y_0)$ for $i = 1, \ldots, n$ and global diagnoser $Diag_g(Q^s_{\mathcal{A}}) = (Y, E_o, \delta_y, y_0)$.

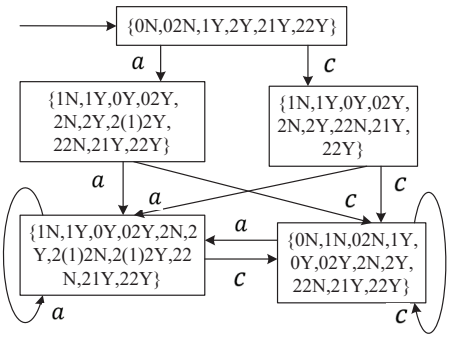
1: **for all** $i \in \{1, \cdots, n\}$ **do**
2: $\quad \delta_{l_i}(N, a_i) = Y$, $\delta_{l_i}(Y, a_i) = Y$
3: $\quad Q_{label(i)} = (\{N, Y\}, \{a_i\}, \delta_{l_i}, N)$
4: $\quad Diag_i(Q^s_{\mathcal{A}}) = Obs(Q^s_{\mathcal{A}} \parallel Q_{label(i)})$
5: **end for**
6: **for all** $i \in \{1, \cdots, n\}$ **do**
7: $\quad \delta_l(N, a_i) = Y, \delta_l(Y, a_i) = Y$
8: **end for**
9: $Q_{label} = (\{N, Y\}, E_a, \delta_l, N)$
10: **return** $Diag_g(Q^s_{\mathcal{A}}) = Obs(Q^s_{\mathcal{A}} \parallel Q_{label})$.

Example 5.6 Consider again the plant Q in Fig. 5.2 with $E = \{a, b, c\}$ and $E_o = \{a, c\}$ under attack as described in Example 5.2. The NFA $Q^s_{\mathcal{A}}$ is depicted in Fig. 5.5. The diagnosers for each attack dictionary and global diagnoser can be constructed by Algorithm 11.

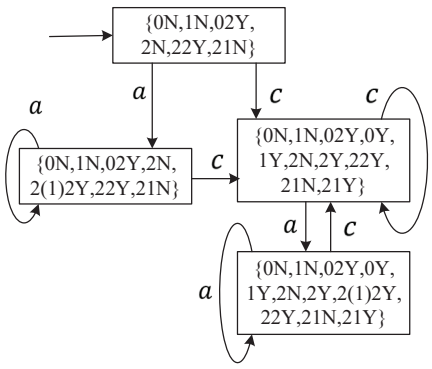
The global diagnoser are given in Fig. 5.6(a) , and the diagonser for A_1 and A_2 are given in Fig. 5.6(b) and Fig. 5.6(c). Note that in the diagnoser we use a simplified notation for describing the states of $Q^s_{\mathcal{A}}$: x_k is denoted by k, x^i_k is denoted by ki, and $x^i_{k_p}$ is denoted by $k(p)i$, where $k \in \{0,1,2\}$, $i \in \{0,1,2\}$, $p \geq 1$. For instance, x^1_2 with attack label Y is denoted by $21Y$ in Fig. 5.6. □



(a) The global diagnoser $Diag_g(Q^s_{\mathcal{A}})$.



(b) The diagnoser $Diag_1(Q^s_{\mathcal{A}})$.



(c) The diagnoser $Diag_2(Q^s_{\mathcal{A}})$.

Fig. 5.6 Diagnosers of $Q^s_{\mathcal{A}}$ in Fig. 5.5.

5.4 Application to Multiple Attacks Detection on Timed Finite Automata

This section presents a discussion on the detection of constant and switching attacks for timed DES modeled as TFA. We first formalize the corruption of timed observation considering attacks over a set of attack dictionaries as follows.

Definition 5.5 Given an attack dictionary $A : E_o \longrightarrow 2^{E_o^*}$, the set of *attacks* over A for a

timed DES is defined by

$$\mathcal{F}_A = \{f_A : (E_o \times \mathbb{R}_{\geq 0})^* \longrightarrow (E_o \times \mathbb{R}_{\geq 0})^* | (\forall \sigma_o \in (E_o \times \mathbb{R}_{\geq 0})^*)(\forall e \in E_o)(\exists n \geq 1) \\ e_1 \cdots e_n \in A(e), f_A(\lambda) = \lambda, \\ f_A(\sigma_o \cdot (e,t)) = f_A(\sigma_o) \cdot (e_1,t) \cdots (e_n,t)\}.\square$$

In other words, an attack for a TFA is a function such that every pair $(e,t) \in E_o \times \mathbb{R}_{\geq 0}$ following with a timed observation σ_o is mapped into a pair $(e_1,t) \cdots (e_n,t)(n \geq 1)$, where $e_1 \cdots e_n$ is a string in $A(e)$. Note that the attack may not corrupt the observation if $e \in A(e)$.

Assume that a TFA may be subject to constant attacks that can corrupt observations by using only one of the attack dictionaries, and switching attacks that may use different attack dictionaries at different steps. The attack dictionary A_0 is introduced in $\mathcal{A}$ to denote the case of no attack. Next we propose the formalization for TFA of constant and switching attacks over a set of attack dictionaries $\mathcal{A} = \{A_0, A_1, \cdots, A_n\}$, respectively.

Definition 5.6 Given a set of attack dictionaries $\mathcal{A} = \{A_0, \cdots, A_n\}$ with $A_i : E_o \longrightarrow 2^{E_o^*} \setminus \{\emptyset\}$, where $i \in \{0, \cdots, n\}$, the set of constant attacks over $\mathcal{A}$ is defined by

$$\mathcal{F}_{\mathcal{A}}^c = \{f_{\mathcal{A}} : (E_o \times \mathbb{R}_{\geq 0})^* \longrightarrow (E_o \times \mathbb{R}_{\geq 0})^* | (\exists i \in \{0, \cdots, n\})\ f_{\mathcal{A}} \in \mathcal{F}_{A_i}\}. \qquad \square$$

Definition 5.7 Given a set of attack dictionaries $\mathcal{A} = \{A_0, \cdots, A_n\}$ with $A_i : E_o \longrightarrow 2^{E_o^*}$, $i \in \{0, \cdots, n\}$, the set of switching attacks over $\mathcal{A}$ is defined by

$$\mathcal{F}_{\mathcal{A}}^s = \{f_{\mathcal{A}} : (E_o \times \mathbb{R}_{\geq 0})^* \longrightarrow (E_o \times \mathbb{R}_{\geq 0})^* | (\forall \sigma_o \in (E_o \times \mathbb{R}_{\geq 0})^*)(\exists i \in \{0, \cdots, n\}) \\ (\forall e \in E_o)(\exists e_1 \cdots e_m \in A_i(e)) f_{\mathcal{A}}(\lambda) = \lambda, \\ f_{\mathcal{A}}(\sigma_o \cdot (e,t)) = f_{\mathcal{A}}(\sigma_o)(e_1,t) \cdots (e_m,t)\}.\square$$

For a TFA, a constant attack over $\mathcal{A} = \{A_0, \cdots, A_n\}$ corrupt the timed observation by one of the attack dictionaries in $\mathcal{A}$. In contrast, the same observable event at different steps could be attacked by a switching attack over $\mathcal{A} = \{A_0, \cdots, A_n\}$ using different attack dictionaries among $A_0, \cdots, A_n$. For timed DES, it still holds that $\mathcal{F}_{\mathcal{A}}^c \subseteq \mathcal{F}_{\mathcal{A}}^s$.

A perspective of future works is constructing a TFA to describe the corruption of constant/switching attacks on the timed observations. The attack detection problem for the original TFA can be further explored by dealing with the issue of state estimation or fault diagnosis for the constructed TFA. It is worth investigating how the knowledge of the timing structure and the knowledge of the time instants in which observable events occur can be exploited for the cyber security of a system.

5.5 Conclusions

In this section, we summarize the conclusions of this chapter as follows:

- We consider a plant modelled as untimed DES that is partially observed through a communication channel by an operator which monitors its evolution. We assume that the communication channel may be subject to multiple attacks, each described by its own attack dictionary. An attack is modelled by a function that maps events to different strings according to one or more attack dictionaries. We distinguish between constant attacks, which corrupt observations using only one of the attack dictionaries, and switching attacks, which may use different attack dictionaries at different steps.

- Suppose that one knows the possible dictionaries that the intruder may use to corrupt the observation. Our goal is to provide a procedure to detect whether an attack has occurred and also to detect the nature of the attack, i.e., which dictionaries were used by the attacker. The solution for a constant (resp., switching) attack is based on constructing an NFA that describes the observations generated by the original system under a constant (resp., switching) attack. We show that the detection of a constant attack (resp., a switching attack) can be reduced to a classical problem of state estimation (resp., fault diagnosis) for the obtained NFA.

- It is worth investigating how the knowledge of the timing structure and the time instants at which observable events occur can be exploited for the cyber security of a system. We briefly discuss the detection of constant and switching attacks for timed DES modelled as TFA. The corruption of timed observation considering attacks over a set of attack dictionaries is formalized. A perspective of future work is to construct a TFA to describe the corruption of constant/switching attacks on the timed observations. The problem of attack detection for the original TFA can be further explored by addressing the problem of state estimation or fault diagnosis for the constructed TFA.

Chapter 6 Conclusions and Future Works

In this dissertation, we consider partially observable timed DES endowed with a single clock. Assuming that the logical and timed structure of a partially observable timed DES is known, this dissertation investigates the state estimation of such timed DES. The conclusions and perspectives are presented as follows.

(1) Models of Timed Discrete Event Systems

We present two models of timed DES, namely TFA and GTFA. Both TFA and GTFA are endowed with a single clock. In addition, a timing function is defined to associate a time interval to each transition specifying at which clock values it may occur. The difference between TFA and GTFA is that: for a TFA, the clock is reset to zero after each event occurs, and the time semantics constrain the dwell time at each discrete state; for a GTFA, there is a clock resetting function associated with each transition specifying how the clock value is updated upon its occurrence; consequently, the time semantics of a GTFA allows a system to remain in a discrete state forever. Both models with partial observation produce a timed observation as a succession of pairs of an observable event and the time instant at which the event has occurred.

(2) Discrete Description of Timed Discrete Event Systems

We provide a purely discrete event description of the behaviour of a TFA/GTFA by associating it with a finite state automaton called a zone automaton. Each state of the zone automaton consists of a pair whose first element is a discrete state of the TFA/GTFA and whose second element is a time interval, called a zone, specifying a range of clock values. Each state of a zone automaton is associated with a single discrete state and provides a coarse estimate of the clock value. The problem of state reachability in the TFA/GTFA is reduced to the reachability analysis of the associated zone automaton.

(3) State Estimation of Timed Discrete Event Systems

State estimation is solved by dealing with two different sub-problems: (a) updating the current estimate as time elapses without receiving any new observation; (b) updating the current estimate when a new observation, namely an observed event and the occurrence time of it, is received. We present a formal approach that allows one to construct offline an observer of TFA/GTFA, i.e., a finite structure that describes the state estimation for all possible evolutions. During the online phase to estimate the current discrete state, one can determine which is the state of the observer reached by the current observation and check to

which interval (among a finite number of time intervals) the time elapsed between the last observed event occurrence belongs. We believe that our approach in this work has a major advantage over existing online approaches for state estimations: it paves the way to address a wide range of fundamental properties (detectability, opacity, etc.) that have mostly been studied in the context of logical DES.

In particular, the state estimation of timed DES under multiple clocks can be investigated in the framework of GTFA. We model such a system as a multiple-clock GTFA, which generalizes the timing function and the clock resetting function to multiple clocks. Multiple GTFA with a single clock and the associated zone automata can assist in analyzing the dynamics and reachability under each clock. In future work, it is of great interest to investigate a structure, for instance, a product of zone automata of multiple clocks, that can provide state estimation of multiple-clock GTFA by checking its reachability.

(4) Applications of Timed Discrete Event Systems

As an application of the state estimation approach for TFA, we assume that TFA may be affected by a set of faults described by timed transitions and aim at diagnosing a fault behaviour based on a timed observation. The proposed solution is based on constructing a fault recognizer that recognizes the occurrence of faults. We conclude that the occurrence of faults can be analyzed by exploring runs in the fault recognizer that are consistent with a given timed observation. Future work includes exploring the diagnosability of TFA, i.e., investigating whether one can certainly detect a fault in a given time interval, which could be of great interest for real-time systems.

We study the problem of attack detection in the context of DESs, assuming that a system can be attacked using one or more attack dictionaries, which map an observable event to a set of corrupted strings. We distinguish between constant attacks, which corrupt observations using only one of the attack dictionaries, and switching attacks, which may use different attack dictionaries at different steps. The problem we address is to determine whether a system has been attacked by a constant (resp., switching) attack and, if so, which attack dictionaries have been used. To solve it in the context of untimed DES, we construct an NFA that produces all the observations generated by a system under a constant (resp., switching) attack. We show that the attack detection problem can be transformed into a classical state estimation (resp., fault diagnosis) problem for the constructed NFA. We note that it is worth investigating how the knowledge of the timing structure can be exploited for the attack detection problem. We present a formalization of the corruption of timed observation considering attacks over a set of attack dictionaries. A perspective of future

work is to construct a TFA that describes the corruption of constant (resp., switching) attacks on the timed observations. The attack detection problem for the original TFA can be further explored by addressing the state estimation (resp., fault diagnosis) problem for the constructed TFA.

References

[1] JENSEN E D, LOCKE C D, TOKUDA H. A time-driven scheduling model for real-time operating systems[C] // In Proceedings of the Real-Time Systems Symposium Conference. 1985 : 112 – 122.

[2] CASSANDRAS C G. The event-driven paradigm for control, communication and optimization[J]. Journal of Control and Decision, 2014, 1(1) : 3 – 17.

[3] CASSANDRAS C G, LAFORTUNE S. Introduction to Discrete Event Systems[M]. Cham : Springer, 2021.

[4] LAFORTUNE S. Discrete event systems: Modeling, observation, and control[J]. Annual Review of Control, Robotics, and Autonomous Systems, 2019, 2 : 141 – 159.

[5] RAMADGE P J, WONHAM W M. Supervisory control of a class of discrete event processes[J]. SIAM Journal on Control and Optimization, 1987, 25(1) : 206 – 230.

[6] RAMADGE P J, WONHAM W M. The control of discrete event systems[J]. Proceedings of the IEEE, 1989, 77(1) : 81 – 98.

[7] WONHAM W M, CAI K. Supervisory Control of Discrete-Event Systems[M]. Cham : Springer, 2019.

[8] ZHOU M, DICESARE F. Petri Net Synthesis for Discrete Event Control of Manufacturing Systems[M]. Boston, MA : Springer, 1993.

[9] OMOGBAI O, SALONITIS K. Manufacturing system lean improvement design using discrete event simulation[J]. Procedia CIRP, 2016, 57 : 195 – 200.

[10] SHU S, LIN F. Decentralized control of networked discrete event systems with communication delays[J]. Automatica, 2014, 50(8) : 2108 – 2112.

[11] LIN F. Control of networked discrete event systems: dealing with communication delays and losses[J]. SIAM Journal on Control and Optimization, 2014, 52(2) : 1276 – 1298.

[12] ZHAO B, LIN F, WANG C, et al. Supervisory control of networked timed discrete event systems and its applications to power distribution networks[J]. IEEE Transactions on Control of Network Systems, 2015, 4(2) : 146 – 158.

[13] CAO X-R, HO Y-C. Models of discrete event dynamic systems[J]. IEEE Control Systems Magazine, 1990, 10(4) : 69 – 76.

[14] VAN TENDELOO Y, VANGHELUWE H. Discrete event system specification modeling and simulation[C] // In Proceedings of the 2018 Winter Simulation Conference (WSC). 2018 : 162 – 176.

[15] RAMADGE P J. Observability of discrete event systems[C] // 1986 25th IEEE conference on

decision and control. 1986 : 1108 – 1112.

[16] OZVEREN C, WILLSKY A. Observability of discrete event dynamic systems[J/OL]. IEEE Transactions on Automatic Control, 1990, 35(7) : 797 – 806. http://dx.doi.org/10.1109/9.57018.

[17] KUMAR R, GARG V, MARCUS S I. Predicates and predicate transformers for supervisory control of discrete event dynamical systems[J]. IEEE Transactions on Automatic Control, 1993, 38(2) : 232 – 247.

[18] HADJICOSTIS C. Estimation and Inference in Discrete Event Systems[M]. [S.l.] : Springer, 2020.

[19] GIUA A, SEATZU C. Observability of place/transition nets[J]. IEEE Transactions on Automatic Control, 2002, 47(9) : 1424 – 1437.

[20] GIUA A, SEATZU C, CORONA D. Marking estimation of Petri nets with silent transitions[J]. IEEE Transactions on Automatic Control, 2007, 52(9) : 1695 – 1699.

[21] JIROVEANU G, BOEL R K, BORDBAR B. Online monitoring of large Petri net models under partial observation[J]. Discrete Event Dynamic Systems, 2008, 18(3) : 323 – 354.

[22] RU Y, HADJICOSTIS C N. Bounds on the number of markings consistent with label observations in Petri nets[J]. IEEE Transactions on Automation Science and Engineering, 2009, 6(2) : 334 – 344.

[23] CABASINO M P, GIUA A, SEATZU C. Fault detection for discrete event systems using Petri nets with unobservable transitions[J]. Automatica, 2010, 46(9) : 1531 – 1539.

[24] TONG Y, LI Z, GIUA A. On the equivalence of observation structures for Petri net generators[J]. IEEE Transactions on Automatic Control, 2015, 61(9) : 2448 – 2462.

[25] YIN X, LAFORTUNE S. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems[J]. IEEE Transactions on Automatic Control, 2015, 61(8) : 2140 – 2154.

[26] ZAYTOON J, LAFORTUNE S. Overview of fault diagnosis methods for discrete event systems[J]. Annual Reviews in Control, 2013, 37(2) : 308 – 320.

[27] ZAD S H, KWONG R H, WONHAM W M. Fault diagnosis in discrete-event systems: Framework and model reduction[J]. IEEE Transactions on Automatic Control, 2003, 48(7) : 1199 – 1212.

[28] LEFEBVRE D, DELHERM C. Diagnosis of DES with Petri net models[J]. IEEE Transactions on Automation Science and Engineering, 2007, 4(1) : 114 – 118.

[29] SAMPATH M, SENGUPTA R, LAFORTUNE S, et al. Diagnosability of discrete-event systems[J]. IEEE Transactions on automatic control, 1995, 40(9) : 1555 – 1575.

[30] SAMPATH M, SENGUPTA R, LAFORTUNE S, et al. Failure diagnosis using discrete-event models[J]. IEEE Transactions on Control Systems Technology, 1996, 4(2) : 105 – 124.

[31] BASILE F, CHIACCHIO P, DE TOMMASI G. On K-diagnosability of Petri nets via integer linear

programming[J]. Automatica, 2012, 48(9) : 2047 – 2058.

[32] CABASINO M P, GIUA A, LAFORTUNE S, et al. A new approach for diagnosability analysis of Petri nets using verifier nets[J]. IEEE Transactions on Automatic Control, 2012, 57(12) : 3104 – 3117.

[33] CABASINO M P, GIUA A, SEATZU C. Diagnosability of discrete-event systems using labeled Petri nets[J]. IEEE Transactions on Automation Science and Engineering, 2013, 11(1) : 144 – 153.

[34] RAMIREZ-TREVINO A, RUIZ-BELTRAN E, ARAMBURO-LIZARRAGA J, et al. Structural diagnosability of DES and design of reduced Petri net diagnosers[J]. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2011, 42(2) : 416 – 429.

[35] SHU S, LIN F, YING H. Detectability of discrete event systems[J]. IEEE Transactions on Automatic Control, 2007, 52(12) : 2356 – 2359.

[36] SHU S, LIN F. I-detectability of discrete-event systems[J]. IEEE Transactions on Automation Science and Engineering, 2012, 10(1) : 187 – 196.

[37] SHU S, LIN F. Generalized detectability for discrete event systems[J]. Systems & control letters, 2011, 60(5) : 310 – 317.

[38] SASI Y, LIN F. Detectability of networked discrete event systems[J]. Discrete Event Dynamic Systems, 2018, 28 : 449 – 470.

[39] SABOORI A, HADJICOSTIS C N. Notions of security and opacity in discrete event systems[C] // 2007 46th IEEE Conference on Decision and Control. 2007 : 5056 – 5061.

[40] LIN F. Opacity of discrete event systems and its applications[J]. Automatica, 2011, 47(3) : 496 – 503.

[41] DUBREIL J, DARONDEAU P, MARCHAND H. Supervisory control for opacity[J]. IEEE Transactions on Automatic Control, 2010, 55(5) : 1089 – 1100.

[42] BEN-KALEFA M, LIN F. Supervisory control for opacity of discrete event systems[C] // 2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton). 2011 : 1113 – 1119.

[43] KEROGLOU C, HADJICOSTIS C N. Probabilistic system opacity in discrete event systems[J]. Discrete Event Dynamic Systems, 2018, 28 : 289 – 314.

[44] CARVALHO L K, WU Y-C, KWONG R, et al. Detection and mitigation of classes of attacks in supervisory control systems[J]. Automatica, 2018, 97 : 121 – 133.

[45] SU R. Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations[J]. Automatica, 2018, 94 : 35 – 44.

[46] LAWFORD M, WONHAM W M. Supervisory control of probabilistic discrete event systems[C] // Proceedings of 36th Midwest Symposium on Circuits and Systems. 1993 : 327 – 331.

[47] BERTRAND N, HADDAD S, LEFAUCHEUX E. Foundation of diagnosis and predictability in probabilistic systems[C] // IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'14) : Vol 29. 2014 : 417 – 429.

[48] SHU S, LIN F, YING H, et al. State estimation and detectability of probabilistic discrete event systems[J]. Automatica, 2008, 44(12) : 3054 – 3060.

[49] PANTELIC V, POSTMA S M, LAWFORD M. Probabilistic supervisory control of probabilistic discrete event systems[J]. IEEE Transactions on Automatic Control, 2009, 54(8) : 2013 – 2018.

[50] LEFEBVRE D, HADJICOSTIS C N. Privacy and safety analysis of timed stochastic discrete event systems using Markovian trajectory-observers[J]. Discrete Event Dynamic Systems, 2020, 30(3) : 413 – 440.

[51] HUMAYED A, LIN J, LI F, et al. Cyber-physical systems security—A survey[J]. IEEE Internet of Things Journal, 2017, 4(6) : 1802 – 1831.

[52] ALGULIYEV R, IMAMVERDIYEV Y, SUKHOSTAT L. Cyber-physical systems and their security issues[J]. Computers in Industry, 2018, 100 : 212 – 223.

[53] ASHIBANI Y, MAHMOUD Q H. Cyber physical systems security: Analysis, challenges and solutions[J]. Computers & Security, 2017, 68 : 81 – 97.

[54] BASILIO J C, HADJICOSTIS C N, SU R, et al. Analysis and control for resilience of discrete event systems: Fault diagnosis, opacity and cyber security[J]. Foundations and Trends® in Systems and Control, 2021, 8(4) : 285 – 443.

[55] BRAVE Y, HEYMANN M. Formulation and control of real time discrete event processes[C] // Proceedings of the 27th IEEE Conference on Decision and Control. 1988 : 1131 – 1132.

[56] KOZAK P. Supervisory control of discrete event processes: A real-time extension[J]. Inst. Inform. Theory Automat., Czechoslovak Acad. Sci., Prague, Tech. Rep, 1991 : 26.

[57] BRANDIN B A, WONHAM W M. Supervisory control of timed discrete-event systems[J]. IEEE Transactions on Automatic control, 1994, 39(2) : 329 – 342.

[58] WONG-TOI H, HOFFMANN G. The control of dense real-time discrete event systems[R]. [S.l.] : STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1992.

[59] LIN F, WONHAM W M. Supervisory control of timed discrete-event systems under partial observation[J]. IEEE Transactions on Automatic Control, 1995, 40(3) : 558 – 562.

[60] KATZ R D. Max-Plus (A, B)-Invariant Spaces and Control of Timed Discrete-Event Systems[J]. IEEE Transactions on Automatic Control, 2007, 52(2) : 229 – 241.

[61] ZHANG R, CAI K, GAN Y, et al. Supervision localization of timed discrete-event systems[J]. Automatica, 2013, 49(9) : 2786 – 2794.

[62] RASHIDINEJAD A, RENIERS M, FENG L. Supervisory control of timed discrete-event systems

subject to communication delays and non-fifo observations[J]. IFAC-PapersOnLine, 2018, 51(7): 456 – 463.

[63] MIAO C, SHU S, LIN F. State estimation for timed discrete event systems with communication delays[C] // 2017 Chinese Automation Congress (CAC). 2017: 2721 – 2726.

[64] LAI A, LAHAYE S, GIUA A. State estimation of max-plus automata with unobservable events[J]. Automatica, 2019, 105: 36 – 42.

[65] ZHANG K. State-based opacity of real-time automata[C] // 27th IFIP WG 1.5 International Workshop on Cellular Automata and Discrete Complex Systems (AUTOMATA 2021): Vol 12. 2021: 1 – 15.

[66] LI J, LEFEBVRE D, HADJICOSTIS C N, et al. Observers for a class of timed automata based on elapsed time graphs[J]. IEEE Transactions on Automatic Control, 2021, 67(2): 767 – 779.

[67] BASILE F, CABASINO M P, SEATZU C. Diagnosability analysis of labeled time Petri net systems[J]. IEEE Transactions on Automatic Control, 2016, 62(3): 1384 – 1396.

[68] HE Z, LI Z, GIUA A, et al. Some remarks on "State Estimation and Fault Diagnosis of Labeled Time Petri Net Systems with Unobservable Transitions" [J]. IEEE Transactions on Automatic Control, 2019, 64(12): 5253 – 5259.

[69] ALUR R, COURCOUBETIS C, HALBWACHS N, et al. The algorithmic analysis of hybrid systems[J]. Theoretical computer science, 1995, 138(1): 3 – 34.

[70] ALUR R, COURCOUBETIS C, HENZINGER T A, et al. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems[R]. [S.l.]: Cornell University, 1993.

[71] HENZINGER T A. The theory of hybrid automata[M]. [S.l.]: Springer, 2000.

[72] ALUR R, DILL D. The theory of timed automata[C] // Real-Time: Theory in Practice: REX Workshop Mook, The Netherlands, June 3–7, 1991 Proceedings. 1992: 45 – 73.

[73] ALUR R, PARTHASARATHY M. Decision problems for timed automata: A survey[J]. Departmental Papers (CIS), 2004: 93.

[74] FINKEL O. Undecidable problems about timed automata[C] // Formal Modeling and Analysis of Timed Systems: 4th International Conference, FORMATS 2006, Paris, France, September 25-27, 2006. Proceedings 4. 2006: 187 – 199.

[75] BOUYER P, HADDAD S, REYNIER P-A. Undecidability results for timed automata with silent transitions[J]. Fundamenta Informaticae, 2009, 92(1-2): 1 – 25.

[76] VAN HUNG D, JI W. On the design of hybrid control systems using automata models[C] // Foundations of Software Technology and Theoretical Computer Science: 16th Conference Hyderabad, India, December 18–20, 1996 Proceedings 16. 1996: 156 – 167.

[77] ALUR R, FIX L, HENZINGER T A. Event-clock automata: A determinizable class of timed automata[J]. Theoretical Computer Science, 1999, 211(1-2) : 253 – 273.

[78] LAROUSSINIE F, MARKEY N, SCHNOEBELEN P. Model checking timed automata with one or two clocks[C] // CONCUR 2004-Concurrency Theory: 15th International Conference, London, UK, August 31-September 3, 2004. Proceedings 15. 2004 : 387 – 401.

[79] DIMA C. Real-time automata[J]. Journal of Automata, Languages and Combinatorics, 2001, 6(1) : 3 – 24.

[80] TRIPAKIS S. Fault diagnosis for timed automata[C] // Formal Techniques in Real-Time and Fault-Tolerant Systems: 7th International Symposium, FTRTFT 2002 Co-sponsored by IFIP WG 2.2 Oldenburg, Germany, September 9–12, 2002 Proceedings 7. 2002 : 205 – 224.

[81] BOUYER P, JAZIRI S, MARKEY N. Efficient timed diagnosis using automata with timed domains[C] // Runtime Verification: 18th International Conference, RV 2018, Limassol, Cyprus, November 10–13, 2018, Proceedings 18. 2018 : 205 – 221.

[82] BOUYER P, HENRY L, JAZIRI S, et al. Diagnosing timed automata using timed markings[J]. International Journal on Software Tools for Technology Transfer, 2021, 23 : 229 – 253.

[83] ZHANG D, FENG G, SHI Y, et al. Physical safety and cyber security analysis of multi-agent systems: A survey of recent advances[J]. IEEE/CAA Journal of Automatica Sinica, 2021, 8(2) : 319 – 333.

[84] RASHIDINEJAD A, WETZELS B, RENIERS M, et al. Supervisory control of discrete-event systems under attacks: An overview and outlook[C] // 2019 18th European Control Conference (ECC). 2019 : 1732 – 1739.

[85] HUANG X, DONG J. Reliable control policy of cyber-physical systems against a class of frequency-constrained sensor and actuator attacks[J]. IEEE Transactions on Cybernetics, 2018, 48(12) : 3432 – 3439.

[86] RABEHI D, MESLEM N, RAMDANI N. Secure interval observer for linear continuous-time systems with discrete measurements subject to cyber-attacks[C] // 2019 4th Conference on Control and Fault Tolerant Systems (SysTol). 2019 : 336 – 341.

[87] LIU H, NIU B, QIN J. Reachability analysis for linear discrete-time systems under stealthy cyber attacks[J]. IEEE Transactions on Automatic Control, 2021, 66(9) : 4444 – 4451.

[88] ZHANG W, MAO S, HUANG J, et al. Data-driven resilient control for linear discrete-time multi-agent networks under unconfined cyber-attacks[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2020, 68(2) : 776 – 785.

[89] ZHANG Q, LI Z, SEATZU C, et al. Stealthy attacks for partially-observed discrete event systems[C] // 2018 IEEE 23rd International Conference on Emerging Technologies and Factory

Automation (ETFA) : Vol 1. 2018 : 1161 – 1164.

[90] BARBHUIYA F, AGARWAL M, PURWAR S, et al. Application of stochastic discrete event system framework for detection of induced low rate TCP attack[J]. Isa Transactions, 2015, 58 : 474 – 492.

[91] LIMA P M, ALVES M V, CARVALHO L K, et al. Security against network attacks in supervisory control systems[J]. IFAC-PapersOnLine, 2017, 50(1) : 12333 – 12338.

[92] FRITZ R, ZHANG P. Modeling and detection of cyber attacks on discrete event systems[J]. IFAC-PapersOnLine, 2018, 51(7) : 285 – 290.

[93] LIMA P M, CARVALHO L K, MOREIRA M V. Detectable and undetectable network attack security of cyber-physical systems[J]. IFAC-PapersOnLine, 2018, 51(7) : 179 – 185.

[94] THORSLEY D, TENEKETZIS D. Intrusion detection in controlled discrete event systems[C] // Proceedings of the 45th IEEE Conference on Decision and Control. 2006 : 6047 – 6054.

[95] GÓES R M, KANG E, KWONG R, et al. Stealthy deception attacks for cyber-physical systems[C] // 2017 IEEE 56th Annual Conference on Decision and Control (CDC). 2017 : 4224 – 4230.

[96] WAKAIKI M, TABUADA P, HESPANHA J P. Supervisory control of discrete-event systems under attacks[J]. Dynamic Games and Applications, 2019, 9 : 965 – 983.

[97] WANG Y, PAJIC M. Supervisory control of discrete event systems in the presence of sensor and actuator attacks[C] // 2019 IEEE 58th Conference on Decision and Control (CDC). 2019 : 5350 – 5355.

[98] JAKOVLJEVIC Z, LESI V, PAJIC M. Attacks on distributed sequential control in manufacturing automation[J]. IEEE Transactions on Industrial Informatics, 2020, 17(2) : 775 – 786.

[99] YAO J, YIN X, LI S. On attack mitigation in supervisory control systems: A tolerant control approach[C] // 2020 59th IEEE Conference on Decision and Control (CDC). 2020 : 4504 – 4510.

[100] YOU D, WANG S, ZHOU M, et al. Supervisory control of Petri nets in the presence of replacement attacks[J]. IEEE Transactions on Automatic Control, 2021, 67(3) : 1466 – 1473.

[101] ZHENG S, SHU S, LIN F. Modeling and control of discrete event systems under joint sensor-actuator cyber attacks[C] // 2021 6th International Conference on Automation, Control and Robotics Engineering (CACRE). 2021 : 216 – 220.

[102] LIN L, SU R. Synthesis of covert actuator and sensor attackers[J]. Automatica, 2021, 130 : 109714.

[103] MEIRA-GÓES R, LAFORTUNE S, MARCHAND H. Synthesis of supervisors robust against sensor deception attacks[J]. IEEE Transactions on Automatic Control, 2021, 66(10) : 4990 – 4997.

[104] GIUA A, MAHULEA C, SEATZU C. Decentralized observability of discrete event systems with synchronizations[J]. Automatica, 2017, 85 : 468 – 476.

Acknowledgement

First of all, I would like to express my gratitude to my supervisors, Prof. Dr. Zhiwu Li and Prof. Dr. Alessandro Giua. I sincerely appreciate their guidance throughout my Ph.D. in analysing problems, writing scientific papers and giving academic presentations. I am honoured to have benefited from their outstanding professionalism and insightful view of research, which I will cherish throughout my life. Their great patience and constant encouragement have supported me from the very first day of my Ph.D. studies.

I would also like to acknowledge the generous support of Prof. Carla Seatzu (University of Cagliari, Italy), whose professional academic suggestions and inspiring encouragement have given me much insight into my career. I am grateful to Prof. Dimitri Lefebvre (University of Normandy, France) for his constructive suggestions and writing guidance on all my papers.

I received a lot of help and kindness during my studies at the University of Cagliari. Thanks again to Prof. Alessandro Giua and Prof. Carla Seatzu for all their help and support in my life in Italy. I appreciate Dr. Yin Tong for helping me to settle down when I first came to Italy. I am grateful to Dr. Yihui Hu, Dr. Nan Du and Dr. Hao Lan for their help in my research and the great experience we shared at the University of Cagliari. I would also like to thank Dr. Zohreh Alzahra Sanai Dashti, Dr. Milad Gholami, Dr. Mauro Franceschelli, Dr. Diego Deplano and Dr. Alessandro Pilloni for helping me to integrate into the *Automatica* research group. Special thanks to Ing. Federica Di Prima for her kind administrative assistance.

I would also like to thank Prof. Kai Cai (Osaka Mertopolitan University, Japan) for the opportunity to visit and work with him at Osaka Mertopolitan University. I benefited greatly from his helpful suggestions during our discussions.

I am also grateful to all members of the System Control & Automation Group in Xidian University during the lab time my PhD: Dr. Ding Liu, Dr. Jiafeng Zhang, Dr. Yufeng Chen, Dr. Gaiyun Liu, Dr. Yifan Hou, Dr. Jinwei Guo, Dr. Xubin Ping, Dr. Meng Qin, Dr. Anrong Wang, Dr. Xi Wang, Dr. Xiuyan Zhang, Dr. Xiubin Zhu, Dr. Lan Yang, Dr. Guanghui Zhu, Dr. Zhou He, Dr. Xuya Cong, and Dr. Zhenzhen Hao, Dr. Cong Wang, Dr. Peng Nie, Dr. Tailong Jing, Dr. Jiazhong Zhou, Dr. Xiaoyan Li, Ms. Ye Liang, Mr. JunjunYang, Mr. Qinrui Chen, Mr. Wei Duan, Mr. Yanjun Zhou, Mr. Dajiang Sun, Mr. Jun Li, Mr. Xiaoyu Han, Mr. Ziliang Zhang, Mr. Shaopeng Hu, Ms. Yao Lu, Ms. Menghuan Hu, Mr. Tianyu Liu, Ms.

Wenjie Zhao, Ms. Yulin Zhao, Mr. Kun Peng, Ms. Dan Zhao, Ms. Yanan Zhang, Ms. Yuling Zhang.

I would like to express my gratefulness to my family. My dear parents Ms. Junyan Zhao and Mr. Ruimin Gao offer their unfailing love and unwavering support, which encourages me all the time. My parents-in-law have got my back at all times. I am grateful for the love of my grandparents in my whole life.

I would like to thank my husband, Dr. Chao Gu. He is not only an excellent colleague during my Ph.D., but also a great life partner, sharing a caring and loving bond that has created a lot of growth for me. Thank you for being there with me all the time.

Finally, many thanks to all those who took the time to read this dissertation and gave me a lot of advice that will help me in my future career.

Biography

1. Basic Information

Chao GAO (female) was born in Baoding (Hebei Province, China) in August 3rd, 1992. She received the B.S. degree in communication engineering from Science and technology college, North China Electronic Power University, Baoding, China, in 2014, and the M.S. degree in electronics and communication engineering from North China Electronic Power University, Baoding, China, in 2017. Since 2017 she has been a Ph.D candidate in cotutorship between School of Electro-Mechanical Engineering (SEME) of Xidian University and Department of Electrical and Electronic Engineering (DIEE) of University of Cagliari (UNICA), co-supervised by Prof. Dr. Zhiwu Li (Xidian University) and Prof. Dr. Alessandro Giua (UNICA).

2. Educational Background

2010.09～ 2014.07, Science and technology college, North China Electronic Power University, B.S. Degree in Communication Engineering

2014.09～ 2017.07, North China Electric Power University, M.S. Degree in Electronics and Communication Engineering

2017.09～ Present, in Ph.D cotutorship between University of Cagliari (Electronics and Computer Engineering) and Xidian University (Control Theory and Control Engineering)

3. Academic Publications

[1] **C. Gao**, D. Lefebvre, C. Seatzu, Z. Li, and A. Giua, "Fault Diagnosis of Timed Discrete Event Systems," *The 22nd IFAC World Congress 2023*, 2023. Accepted.

[2] **C. Gao**, D. Lefebvre, C. Seatzu, Z. Li, and A. Giua, "A region-based approach for state estimation of timed automata under no event observation," In *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 799–804, 2020.

[3] **C. Gao**, C. Seatzu, Z. Li, and A. Giua, "Multiple Attacks Detection on Discrete

Event Systems," In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 2352–2357, 2019.

4. ACADEMIC ACTIVITIES

[1] Sep. 2021, AUTOMATICA.IT 2021 Workshop, Virtual Edition, Catania, Italy. (Oral presentation)

[2] Sep. 2020, 2020 IEEE International Conference on Emerging Technology and Factory Automation, Vienna, Austria. (Oral presentation)

[3] Jun. 2020, 2020 DRIEI Ph.D. Program in Electronic and Computer Engineering, Cagliari, Italy. (Oral presentation)

[4] Oct. 2019, 2019 IEEE International Conference on Systems, Man, and Cybernetics, Bari, Italy. (Oral presentation)

[5] Jun. 2019, 2019 DRIEI Ph.D. Program in Electronic and Computer Engineering, Cagliari, Italy. (Oral presentation)

5. ACADEMIC SERVICES

Reviewer for Journal: Automatica; IEEE Transactions on Automatic Control; International Journal of applied mathematics and computer science.

Reviewer for Conference: IEEE International Conference on Systems, Man, and Cybernetics; IFAC Workshop on Discrete Event Systems; European Control Conference; IFAC Symposium on Fault Detection, Supervision and Safety for Technical Process.

6. ACADEMIC PROGRAMS

Host (1/1), 2021.5-2022.4,
Fundamental Research Funds for the Central Universities and the Innovation Fund of Xidian University.

Participant (2/4), 2020.7-2021.6,
Shaanxi Province Natural Science Fundamental Research Program.