January 2018

Detectability Of Fuzzy Discrete Event Systems

Ahmed Mekki
Wayne State University, ao0973@wayne.edu

DETECTABILITY OF FUZZY DISCRETE EVENT SYSTEMS

by

AHMED MEKKI

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2018

MAJOR: ELECTRICAL ENGINEERING

Approved By:

_____

Advisor Date

_____

_____

_____

_____

DEDICATION

This thesis is dedicated to all the members of my family, who provided unlimited moral and spiritual support.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors Professor Hao Ying and Professor Feng Lin for their unlimited moral and technical support of my PhD study and related research, as well as their patience, motivation, and immense technical contributions.

My sincere thanks also go to Professor Le Wang, Professor Ming Dong and Professor Xiaoyan Han for their valuable input and direction.

Last but not the least, I would like to thank my family including my mother, my wife, my children, and my uncle for supporting me throughout the writing of this thesis and my life in general.

Also, my sincere thanks goes to my friend Nabil Elayyan for his moral and technical support.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

The term ‘discrete event system’, or DES, is commonly used in a narrow meaning to refer to a process or simulation where the dynamics of the system is represented as a sequence of operations (arrival, delay, capturing the resource division, etc.) by some of the entities. DES is a special type of dynamic system that can be used to represent clients, documents, calls, data packets, vehicles, etc. These entities are generally acknowledged to be passive, as they do not control their dynamics. However, they are likely to have certain attributes that affect the process of treatment, such as the type of a call, the complexity of work, etc. A DES can be defined as a discrete-state, event-driven system, that is, the evolution of its states depends entirely on the occurrence of asynchronous discrete events over time. Such system can be defined in terms of set of states (including initial state), set of events and a transition function, which defines subsequent state or set of states based on the current state and the accruing event or sequence of events.

1.1 Problem overview and motivation

Knowledge of the system’s state is of great interest to decide actions to be taken in many applications. Numerous investigations in the area of discrete event systems (DES) and their detectability have been conducted. However, only few studies have addressed the detectability of fuzzy discrete event system (FDES). Let’s take the example of an automobile engine and model it as a system that has only 4 distinct states: off, cranking, idling, and running states. It is crucial to know the exact engine’s state to allow or disallow some specific actions. For example, if the engine is idling or running, there will be no need to crank the engine. Therefore, cranking while engine is running or idling should be

disallowed to protect the starter's motor from being damaged. In real-world applications, not all the states are distinct as in the engine example. In fact, they can be in more than one state at the same time depending on subjectivity and human perceptions. Such systems cannot be accurately modeled using conventional DES. This vagueness in real-world problems prompted the extension of DES to FDES.

In many situations, systems or activities can be seen to experience and undergo varying stages or state conditions. It may also be difficult to assess the state condition and ascertain the response to impacting factors at such varying stages or states. For instance, in medical field knowledge of illness stage of a patient or his/her response to a treatment is very important for doctors or care givers to decide how to proceed with the treatment plan. In such system's state, i.e. illness stage, where characteristics and prognoses are not obvious, it is difficult to decide the necessary actions and the appropriate treatment plan to be followed. Hence, the treatment plans for such systems are subjective and dependent on the care givers' perceptions. The best technique for modeling such systems is to utilize the fuzzy logic, which enables the representation of systems in terms of fuzzy states (i.e. patient's state) and fuzzy events (i.e. treatment or surgery). Therefore, FDES can be applied to broader range of applications and systems. In addition, fuzzy logic allows a given system to be in many states at the same time based on individuals' perceptions. Hence, establishing detectability criteria is essential in achieving successful and matured FDES in real-world applications.

A fuzzy state can be represented by a vector where each of its elements represents the possibility of being at a specific state. For example, for a state vector$[0.8 \quad 0.2 \quad 0]$, the system's state can be interpreted as:

- State 1→ 80% possibility
- State 2→ 20% possibility
- State 3→ 0% possibility

To investigate the detectability of FDES, which is the main focus of this research, we need to develop a criterion to address the fuzzy nature of the system and help estimate its current states and the subsequent states as well after the occurrence of an event sequence. In other words, we need to defuzzify the state vectors and check them for detectability. To do so, we introduced the certainty criterion, which can be used for the defuzzification process.

1.2 Literature review

The aim of this literature review is to study, analyze, and emphasize the essential findings in the fields of DES and FDES. It is assumed that the detectability of a DES is of great importance due to the fact that many control applications decide the next action or set of actions to be taken based on the detected state or states of the system.

Control of DES has been extensively investigated over the past decades [3], [9], [12]-[13], [21], [23]; in particular, the supervisory control theory with the concepts of controllability and observability, which were established by (Ramadge and Wonham [20]) and (Lin and Wonham [10]), respectively.

In the supervisory control, as well as many applications, knowledge of the system's state is crucial in action decisions. Therefore, state estimation (detectability) is of great significance in DES and hence has been investigated by many scholars [4], [19], [22], [34].

Shu et al. studied the determination of the current and subsequent state of DES based on sequences of observations (state observations and/or event observations) [29]. These investigations resulted in defining four types of detectabilities for the DES: strong detectability, (weak) detectability, strong periodic detectability, and (weak) periodic detectability. A system is strongly detectable if the current and subsequent states of the system can be uniquely determined after a finite number of observations for all trajectories of the system. On the other hand, a system is considered (weakly) detectable if the current and subsequent states of the system can be uniquely determined after a finite number of observations for some trajectories of the system. Strong periodic detectability and (weak) periodic detectability are readily defined based on periodic determinations of the system's state. The authors derived necessary and sufficient conditions, which can be checked by constructing an observer for the defined detectabilities. In addition, the authors claimed that their findings can be used in feedback control.

The investigation conducted by Shu and Lin [25] demonstrated that the discrete event systems of non-deterministic characteristics were of greater importance than those of deterministic ones. In this study, the authors were able to extend detectability definitions from deterministic systems to non-deterministic ones, which is very important as not all applications can be modeled by deterministic systems. Shu and Lin, in the same study, also developed a polynomial algorithm to check strong detectability by constructing a detector

rather than an observer in order to reduce the complexity. One of their most important achievements is the extension of *D*-detectability, which makes detectability theorems of DES more applicable in solving many problems as it focuses on distinguishing only certain pairs of states in the system after finite number of observations.

Shu et al. introduced the concept of distinguishability in [31] based on strong detectability and strong periodic detectability. Distinguishability is used to characterize detectability for on-line detection and sensor activation.

The study of detectabilities in probabilistic DES (also known as PDES) was carried out by Shu et al. [30]. The researchers performed this study to support the idea that the stability of the system can be identified by the ability to specify the system's states along some trajectories. The authors assumed complete observation for PDES; they first converted the system into a non-deterministic one and then investigated the convergence of the event sequence. They claimed that an event sequence is convergent (and hence detectable) if along this sequence it is more certain that the system is in a particular state. The introduction of probability to the state estimation problem increases the complexity of the system, and as a result, it increases the opportunity of being used in different applications. The authors claimed that construction of an observer would not be sufficient to study detectability in PDES; therefore, they studied the asymptotical behavior of the system.

Shu and Lin continued their work and extended their investigations to include delayed detectability in a DES [26]. In this study, the authors answered the question of whether it is possible to determine the state of a system at the time the ***k1th*** event was

observed after observing ***k1+k2*** observable events. The system is then said to be delayed detectable if determination of the system's state is possible. The authors also developed polynomial algorithms to check delayed detectability. Furthermore, observability and diagnosability were shown as special cases of delayed detectability.

Prior to the investigations of the detectability, many researches carried out extensive studies in the diagnosability, which have been published in [11], [14]-[16], [32]-[33]. These generally describe the ability to check for fault occurrences in a system.

Another important aspect of detectabilities in DES, conducted by Shu and Lin, is the detectability of the initial state of the DES or *I*-detectability [28]. In this study, the authors proved that DES is strongly *I*-detectable if they can determine the initial state of the DES after a finite number of observations for all trajectories of the system and is weakly *I*-detectable if they can determine the initial state for some trajectories of the system. To perform their research, the authors constructed an *I*-detector to avoid exponential complexity. In addition, the authors developed algorithms to check the systems' characteristics and for sublanguage calculations.

The control the DES to enforce detectabilities was investigated by Shu and Lin [27]. In this study, control actions were used to ensure continuous or periodic determination of the current state of a closed-loop system after some observations in all the system's trajectories.

The work reviewed above assumes that all event observations are static; that is, if an event is observable, then all its occurrences are observable. However, this is not consistent with real-world applications, which prompted Shu and Lin to extend their

investigations to include DES with dynamic event observations [24]. In this study, the authors generalized the static event observation into a dynamic event observation, based on which the detectability problem was investigated. The authors developed necessary and sufficient conditions to check for different types of detectabilities: strong, weak, strong weak, and periodic weak detectability by constructing an observer and/or a detector.

In many practical applications, particularly in biomedical applications, the state of a system is not crisp but rather vague. To handle this type of systems, the DES have been extended to the FDES [7]. A FDES is modeled by using fuzzy states and fuzzy events. Matrices are adopted for mathematical representations of fuzzy states and fuzzy events, where the elements of these matrices have values ranging between 0 and 1. Fuzzy operations, such as Max-Min operation, are used to calculate the next state after the occurrence of an event sequence. Such procedures and techniques were further followed and investigated by many researchers [5], [6], [8], [35], and [36].

The research carried out by Mekki et al. [1] addressed the detectability problem for a special case of FDES where all events are observable. The authors first introduced fuzzy discrete event system with constraints, which defines all the event sequences (language) allowed in the system. Furthermore, the authors extended the detectability of DES to *N*-detectability by constructing an *N*-step look-ahead tree (from the initial state) that describes all possible event sequences of length *N* in the system. In addition, the authors proved that *N*-detectability implies detectability in crisp DES. Evolving from what has been established, the authors defined *N*-detectability of FDES and developed necessary algorithms to compute the corresponding fuzzy states for each event sequence defined in

the constraint system and then check computed fuzzy states for *N*-detectability by examining them against certainty criterion, which was developed by the authors for that purpose. Certainty is determined by computing the ratio of the maximal element in each fuzzy state over the sum of all elements in that state and then comparing it against a defined factor (threshold). If the outcome is greater or equal to the threshold, then the fuzzy state is said to be certain; otherwise, the fuzzy state is not certain. Based on the certainty outcome of all fuzzy states (nodes) along an event sequence (trajectory), a branch (trajectory) is classified as follows: detectable, periodically detectable, or not detectable. Based on that, FDES is said to be strongly *N*-detectable if all branches in the system are detectable and (weakly) *N*-detectable if some of the branches are detectable. Similarly, the FDES is said to be strongly periodically *N*-detectable if all branches are strongly periodically detectable and (weakly) periodically *N*-detectable if some of the branches are periodically detectable.

Detectabilities investigated so far in the literature are for FDES with full observation. However, no research has addressed FDES with partial observations. Since in real-world applications not all events are observable (especially in medical field and many engineering applications), it is imperative to investigate detectabilities of FDES with partial observations which is going to be discussed in Chapter 5.

1.3 Research objectives and contributions

This research is aimed to generalize the already defined detectability definitions for DES and extend them to cover systems with substantial vagueness and inaccuracy such as fuzzy discrete event systems (FDES). In fact, detectability definitions of DES systems will be shown as a special case of detectability definitions of FDES.

One of the main contributions of this study is the extension of detectability definitions of a conventional DES to introduce 4 types of detectabilities for partially and fully observable FDES, which are:

1 Strong *N*-detectability

2 Weak *N*-detectability

3 Periodic *N*-detectability

4 Weak periodic *N*-detectability

Also, other contributions of this study are the modeling of real-world applications such as vehicle dynamics control using the new detectability criteria (*N*-detectability), and demonstrating the practical aspect, with real-world filled-in data, to prove the potential applicability of this *N*-detectability in the automotive industry in general and active safety in particular.

1.4 Organization of the dissertation

This dissertation is organized as follows: chapter 2 gives broad introduction to DES, fuzzy logic, FDES, and FDES with constraints. Chapter 3 discusses the existing detectability definitions for DES as well as the extension of DES detectability definitions to introduce *N*-detectability of DES. Chapter 4 discusses *N*-detectability of fully observable and partially observable FDES. Chapter 5 discusses the practical aspect of the developed theorems and shows vehicle dynamic control as a potential application. Chapter 6 shows the conclusion and recommendation for future researches.

CHAPTER 2 DISCRETE EVENT SYSTEMS AND FUZZY DISCRETE EVENT SYSTEMS

2.1 Discrete event systems

As stated in Chapter 1, a DES is modeled in terms of states, events, dynamics, and initial state. Mathematically DES system is modeled as in Eq. (2.1).

$$G = (Q, \Sigma, \delta, q_0) \tag{2.1}$$

where

- Q is the finite set of discrete states
- $q_0 \in Q$ is the initial state
- Σ is finite set of events
- $\delta: q \times \Sigma \rightarrow Q$ is the transition function from current state to subsequent state in the system after the occurrence of a given event or sequence of events

The set of sequences/strings of events that can take place in a discrete event system are referred to as the language of the system $L(G)$, which is defined in Eq. (2.2)

$$L(G) = \{s \in \Sigma^*: \delta(q_0, s) \neq \emptyset\} \tag{2.2}$$

where Σ^* is the Kleene closure of Σ, which is defined as all possible sequences of events in the event set Σ.

The system in Figure 2.1 represents a printing machine that has 3 distinct states: idle (I), working (w), and down (D). The events of the system are: turn switch on (α), turn switch off (β), break down (λ), and repair (μ). I, in this example, is the initial state of the system.

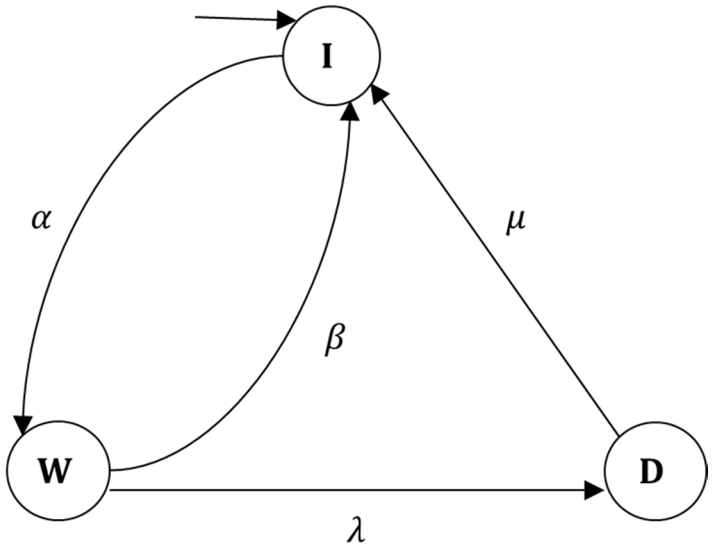

Figure 2.1 Discrete event system for printing machine

The automation in Figure 2.1 can be modeled as follows:

$$G = (Q, \Sigma, \delta, q_0)$$

where

$$Q = \{I, W, D\}$$

$$\Sigma = \{\alpha, \beta, \lambda, \mu\}$$

$$\delta: q \times \Sigma \rightarrow Q$$

$$q_0 = \{I\}$$

The printer machine operates as follows. It starts from the initial state (I) and when the switch is turned 'ON', it will make a transition to state $\delta(I, \alpha)$, which is the working state (W). At working state (W) and upon the occurrence of the event λ (the machine breaks down) the system will make a transition to down state (D) and stay there until the machine is repaired by performing repair (μ). These two transitions can be expressed as follows, respectively:

$$\delta(W, \lambda) = D$$

$$\delta(D, \mu) = I$$

When the system is at the initial state (I) and the event sequence $\alpha\lambda$ occurs, the subsequent state can be obtained as follows.

$$\delta(I, \alpha\lambda) = \delta(\delta(I, \alpha), \lambda) = \delta(W, \lambda) = D$$

Depending on the definition of the transition function δ, the DES can be classified as deterministic or non-deterministic. For instance, if the occurrence of the event $e \in \Sigma$ at a given state $q \in Q$ (the transition $\delta(q, e)$) leads to a unique state in the system, then the system is considered deterministic. In contrast, if the transition $\delta(q, e)$ leads to multiple states, then the system is considered a nondeterministic one. The system shown in Figure 2.1 is considered a deterministic system as we can uniquely determine the subsequent state of the system after the occurrence of a defined event. Nondeterministic DES will be introduced in details in Section 2.1.1.

2.1.1 Nondeterministic discrete event systems

Unlike deterministic DES, the initial state of a nondeterministic DES can itself be a set of states. Furthermore, an occurrence of a given event can cause a transition to multiple states in the system. More generally, the automation of nondeterministic DES can be modeled as:

$$G_{nd} = (Q, \Sigma \cup \{\epsilon\}, \delta_{nd}, Q_0) \tag{2.3}$$

where

- Q, Σ are the same entities defined for the deterministic DES
- $\delta_{nd}: q \times \Sigma \rightarrow 2^Q$ is the nondeterministic transition function

- $Q_0 \subseteq Q$ is the set of the initial states of the system

2^Q is defined as the power set of Q and can be obtained as in Eq. (2.4)

$$2^Q = \{X: X \subseteq Q\} \tag{2.4}$$

The set of the events Σ is divided into two disjointed parts with respect to observability:

- The observable part: Σ_o
- The unobservable part: Σ_{uo}

Mathematically this relation can be expressed as in Eq. (2.5)

$$\Sigma = \Sigma_o \cup \Sigma_{uo} \tag{2.5}$$

Unobservable events are the set of events defined in Σ, but are not observed when they take place. When a sequence or string that contains unobservable events occurs in a system, its observation is described in terms of projection as follows. Given large event set Σ and small event set Σ_o such that $\Sigma_o \subset \Sigma$, the projection of a string is a mapping $P: \Sigma^* \rightarrow \Sigma_o^*$ that erases all events that are not in Σ_o. Formally it is defined as in Eq. (2.6)

$$P(\varepsilon) = \varepsilon \, , P(s,\sigma) = \begin{cases} \mathrm{P(s)}\sigma \text{ if } \sigma \in \Sigma_o \\ \mathrm{P(s)} \text{ if } \sigma \notin \Sigma_o \end{cases} \tag{2.6}$$

where ε denotes the empty string. The projection P is then extended from strings to languages.

The inverse of projection P is denoted by P^{-1} and is given by Eq. (2.7).

$$P^{-1}(s) = \{t \in \Sigma^*: P(t) = s\} \tag{2.7}$$

Nondeterministic systems can always be transformed into an equivalent deterministic one known as "observer" and denoted by G_{obs}, which is defined as in Eq. (2.8).

$$G_{obs} = AC(Q_{obs}, \Sigma_o, \xi, Q_{0,obs}) \tag{2.8}$$

where

- $AC(G)$ is the accessible part of automata G
- $Q_{obs} = 2^Q$ is the power set of Q
- Σ_o is the set of observable events
- ξ is the transition function defined in Eq. (2.10)
- $Q_{0,obs} = UR(q_0)$, is the initial states, which are defined in terms of the unobservable reach of q_0 defined in Eq. (2.9)

For $x \in X$, where $x \subseteq Q$, the unobservable reach is given by

$$UR(x) = \{q \in Q : (\exists q' \in x) q \in \delta_{nd}(q', \varepsilon)\} \tag{2.9}$$

For $x \in X$ and $\sigma \in \Sigma_o$,

$$\xi(x, \sigma) = UR(\{q \in Q : (\exists q' \in x) q \in \delta_{nd}(q', \sigma)\}) \tag{2.10}$$

Example 2.1

Consider the nondeterministic DES shown in Figure 2.2 with the set of states $Q = \{q_0, q_1, q_2, q_3\}$, the set of events $\Sigma = \{\varepsilon, \alpha, \beta\}$, wher ε is the empty string, and the initial state is $Q_0 = \{q_0, q_1\}$.

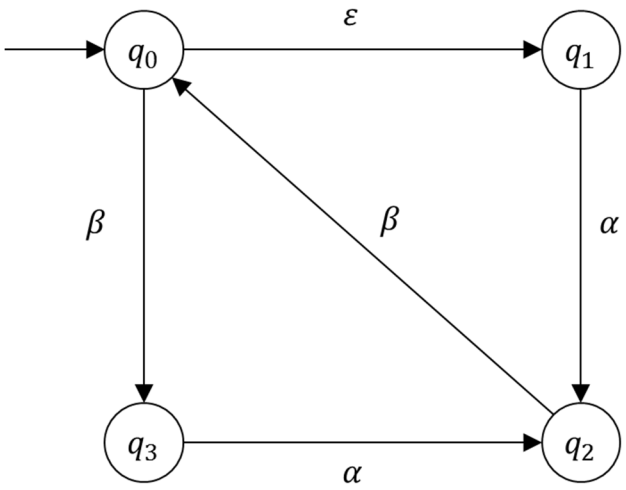

Figure 2.2 A nondeterministic DES

The equivalent observer G_{obs} for the nondeterministic DES in Figure 2.2 is represented by the automata shown in Figure 2.3.

If the system is at the initial state q_0 and the event β is observed, then the next state is obtained as follows:

$$q_{next} = \xi(q_0, \beta) = UR(\{q \in Q : (\exists q' \in q_0) q \in \delta_{nd}(q', \beta)\}) = q_3$$

q_3 in this example is a unique state. On the other hand, if the event sequence $\alpha\beta$ is observed, then the next state cannot be uniquely determined as it falls in the set $\{q_0, q_1\}$.

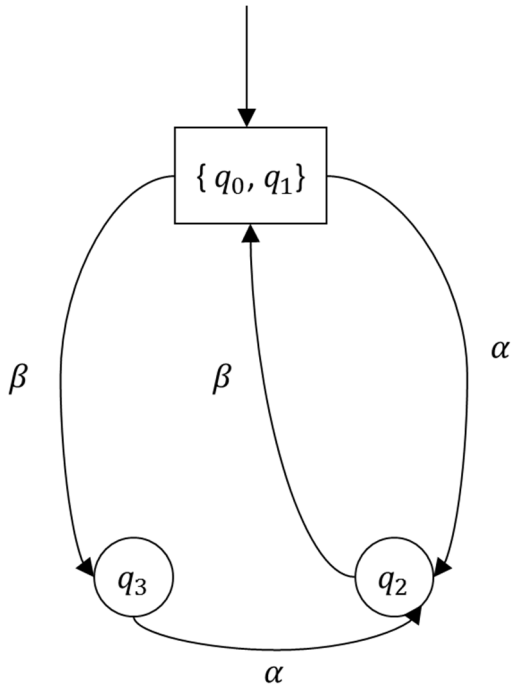

Figure 2.3 Equivalent observer for DES of Figure 2.2

2.2 Fuzzy logic

Fuzzy logic was first introduced in 1965 by Lotfi A. Zadeh [38], professor for computer science at the University of California in Berkeley. Fuzzy logic is a departure from classical two-valued sets and logic, that uses "soft" linguistic (e.g. large, hot, tall) system variables and a continuous range of truth values in the interval [0,1], rather than strict binary (True or False) decisions and assignments. Formally, fuzzy logic is a structured, model-free estimator that approximates a function through linguistic input/output associations.

Fuzzy rule-based systems apply these methods to solve many types of real-world problems, especially where a system is difficult to model, controlled by a human operator

or expert, or where ambiguity or vagueness is common. A typical fuzzy system consists of a rule base, membership functions and an inference procedure [37].

Some fuzzy logic applications include:

- Control (Robotics, Automation, Tracking, and Consumer Electronics)
- Information Systems (DBMS, Info. Retrieval)
- Pattern Recognition (Image Processing, Machine Vision)
- Decision Support (Adaptive HMI, Sensor Fusion)

Some of the key benefits of fuzzy design are:

- Simplified and reduced development cycle
- Ease of implementation
- Can provide more user-friendly and efficient performance

Fuzzy logic has been used as a powerful tool for the controlling of subway systems and complex industrial processes, as well as for household and entertainment electronics, diagnosis systems and other expert systems with substantial vagueness.

Fuzzy logic is basically a multivalued logic that allows intermediate values to be defined between conventional evaluations like yes/no, true/false, black/white, 0/1 etc. Notions like rather warm or pretty cold can be formulated mathematically and processed by computers. In this way a more human-like way of thinking can be applied in computer programming and processing.

Fuzzy logic has many operations that can be performed on fuzzy sets including union, intersection, complement, algebraic product, algebraic sum, etc. [37]. In this section,

we give more emphasis to the operations that are widely used in this thesis. Let's first define the fuzzy sets and then define some of the operations that can be performed on it.

A fuzzy set can be regarded as an extension of classical sets and can be defined as any set that allows different degree of membership for its elements. The main distinction of fuzzy sets is that its elements have varying degrees of membership in the set.

2.2.1 Mathematical Concept

A fuzzy set Z in the universe S can be defined as a set of ordered pairs that can be given by Eq. (2.11).

$$Z = \{(y, \mu_Z(y)) \mid y \in S\} \tag{2.112}$$

where $\mu_Z(y)$ is the degree of membership of y with values ranging from 0 to 1.

To best illustrate the concept, let's consider youngness example. Let set S (the universe of discourse) is the set of people. A fuzzy subset YOUNG is used to define the degree of youngness of each person in S. Let's assign a degree of membership in the fuzzy subset YOUNG. The easiest way to do this is with a membership function based on the person's age as illustrated in Table 2.1.

Person	Age	Degree of youngness
Ali	25	1.00
Chin	39	0.10
Smith	34	0.60
Nadia	65	0.00
Khan	31	0.90

Table 2.1 Degree of membership in the set S

Table 2.1 can be expressed in terms of Eq. (2.11) as follows:

$$S = \{Ali, Chin, Smith, Nadia, Khan\}$$

$$Z = \{(Ali, 1), (Chin, 0.1), (Smith, 0.6), (Nadia, 0), (Khan, 0.9)\}$$

The fuzzy set YOUNG can be obtained from:

$$young(x) = \{1, if\ age(x) \leq 30,$$

$$\frac{40 - age(x)}{10}, if\ 30 < age(x) \leq 40,$$

$$0, otherwise.\}$$

The membership function can be represented as in Figure 2.4.

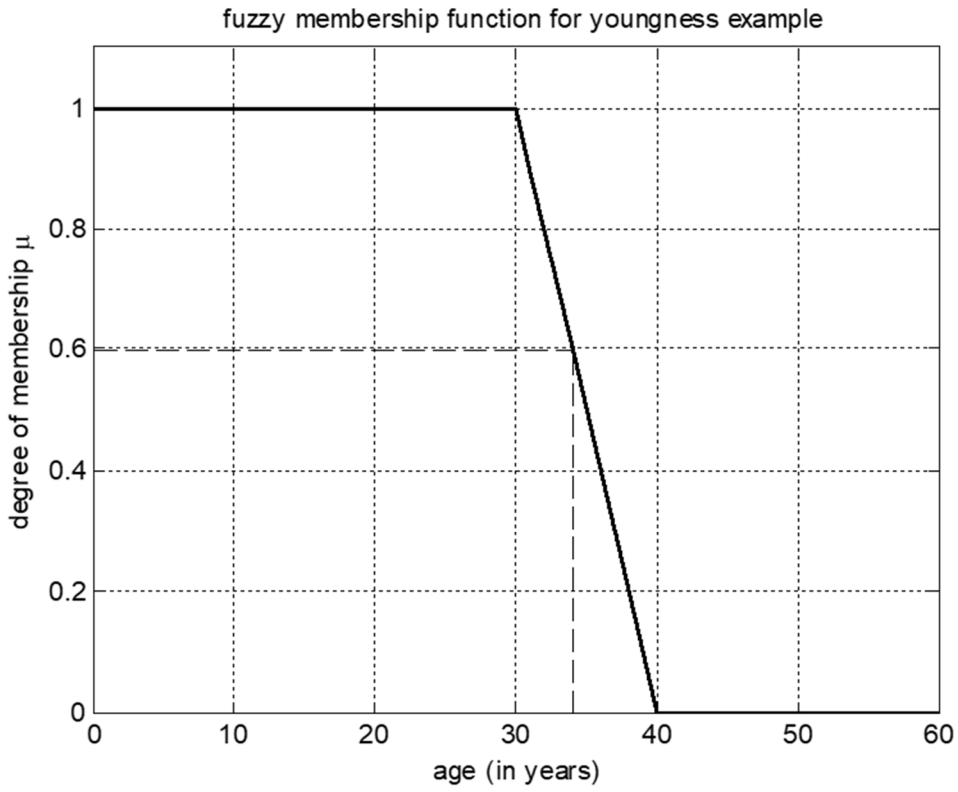

Figure 2.4 Membership function that determines the degree of belonging of member x in S to fuzzy set Z

From Figure 2.4 and with the given definition, we can say that the degree of the truth of the statement “Smith is YOUNG” is 0.6.

2.2.2 Fuzzy operations

Union:

The membership function of the union of fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the maximum of two membership functions. Union operations on fuzzy sets is equivalent to OR operations in Boolean algebra.

$$\mu_{A\cup B} = \max(\mu_A, \mu_B)$$

Intersection:

The membership function of the intersection of fuzzy sets A and B with membership functions μ_A and μ_B respectively is defined as the minimum of two membership functions. Intersection operations on fuzzy sets is equivalent to AND operations in Boolean algebra.

$$\mu_{A\cap B} = \min(\mu_A, \mu_B)$$

Composition of fuzzy relationship:

Fuzzy relation in different product space can be combined with each other by the operation called “Composition”. There are many composition methods in use, e.g. max-product, Max-Min, and Max-Average methods. Max-Min composition is widely used in fuzzy logic applications and is adopted in our computer program to compute the fuzzy state estimate after the occurrence of a fuzzy event sequence.

Max-Min composition:

Take the product spaces $\bar{R} = \{((x,y), \mu_{\bar{R}}(x,y)) \mid (x,y) \in X \times Y\}$ and $\bar{S} = \{((y,z), \mu_{\bar{S}}(y,z)) \mid (y,z) \in Y \times Z\}$. The Max-Min composition\operation of $\bar{R}$ and $\bar{S}$ defines how *X* and *Z* are related, which denoted by $\bar{R} \circ \bar{S}$.

$$\bar{R} \circ \bar{S} = \{(x,z), \mu_{\bar{R}\circ\bar{S}}(x,z) \mid x \in X, y \in Y, z \in Z\}$$

where $\mu_{\bar{R}\circ\bar{S}}(x,z) = \vee \{\mu_{\bar{R}}(x,y) \wedge \mu_{\bar{S}}(y,z)\}$

To explain how the operation Max-Min is performed, take the below two relations R and S respectively:

$\bar{R}$	y_1	y_2	y_3
x_1	0.1	0.3	0.7
x_2	0.9	0.2	0.4
x_3	0	0.9	0.5

$\bar{S}$	z_1	z_2	z_3
y_1	0	0.2	0.5
y_2	0.4	0.6	1
y_3	0	0	1

$$\bar{R} \circ \bar{S} = \begin{bmatrix} 0.1 & 0.3 & 0.7 \\ 0.9 & 0.2 & 0.4 \\ 0 & 0.9 & 0.5 \end{bmatrix} \circ \begin{bmatrix} 0 & 0.2 & 0.5 \\ 0.4 & 0.6 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.3 & 0.7 \\ 0.2 & 0.2 & 0.5 \\ 0.4 & 0.6 & 0.9 \end{bmatrix}$$

2.3 Fuzzy discrete event systems

Conventional discrete event systems (DES) failed to effectively model and solve many real-world problems such as modeling and treatment of HIV and cancers as well as many other applications from different fields due to their imprecise and subjective nature. For instance, the description of human health, i.e. “good” or “bad” is subjective and

depends on the doctor's or the care giver's perceptions. The patient's response to a treatment or medication (good or bad) can also be vague and subjective upon the patient's mood and feelings. Even the transition from one state to another (good to bad or vice versa) is vague as well. This prompted many scholars to pursue alternative options to address the matter and include fuzzy logic in the modeling process of DES. Fuzzy discrete event systems (FDES) and its modeling techniques have been introduced since the early 2000's by Lin and Ying [7], where their main focus was to establish the frame work for the modeling of complex problems using the FDES in order to effectively model and provide solutions to real-world problems with substantial vagueness, impreciseness, and inherent subjectivity such as biomedical applications. Unlike crisp DES, the use of nodes and arcs to graphically represent the FDES automata is not suitable. Instead, vectors and matrices were found to be the best tools for mathematical representations of system's states and system's events respectively. All the states of the FDES are modeled by vectors whose elements range from 0 to 1 ([0, 1]). All the events are modeled by square matrices that have sizes equivalent to the number of states in the FDES. Each element in an event matrix has a value ranging from 0 to 1.

Furthermore, the supervisory control problem of the FDES with partial observation have been investigated in [36], where the observability, co-observability, and normality of the DES have been extended and redefined to include FDES.

The extension of parallel composition of the DES in order to define parallel composition for the FDES has been introduced in [6], where max-product and max-min

automata were considered to model the FDES. The same investigation resulted in defining a uniform criteria to check for fuzzy controllability for the FDES.

Definitions

Fuzzy state:

A fuzzy state is a state in DES where the system can simultaneously be in any state within the system with different degree of possibility (membership).

Fuzzy event:

A fuzzy event is an event that can cause transition in the system from any state within the system to any state within the same system with a different degree of possibility (membership).

In [7], the FDES Automata is modeled as in Eq. (2.12).

$$\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{q}_0) \tag{2.12}$$

where $\tilde{Q}$, $\tilde{\Sigma}$, $\tilde{\delta}$, and $\tilde{q}_0$ are state space, set of fuzzy events, the transition function, and the initial state respectively. $\tilde{Q}$ and $\tilde{\Sigma}$ describe the fuzziness of the systems as follows:

$$\tilde{Q} = [0,1]^n \tag{2.13}$$

$$\tilde{q} \in [0,1]^n \tag{2.14}$$

$$\tilde{\Sigma} = \{\tilde{\sigma}_1, \tilde{\sigma}_2, \dots, \tilde{\sigma}_m\} \tag{2.15}$$

$$\tilde{\sigma} \in [0,1]^{n*n} \tag{2.16}$$

The transition from one state to another after the occurrence of a fuzzy event is obtained by $\tilde{\delta}(\tilde{q}, \tilde{\sigma}) = \tilde{q} \circ \tilde{\sigma}$, where $\circ$ is either the Max-Min or Product-Max operation of fuzzy inference.

Finite automata in Figure 2.5 describes the stages of the lower back injuries and the corresponding treatment plans to address such conditions.

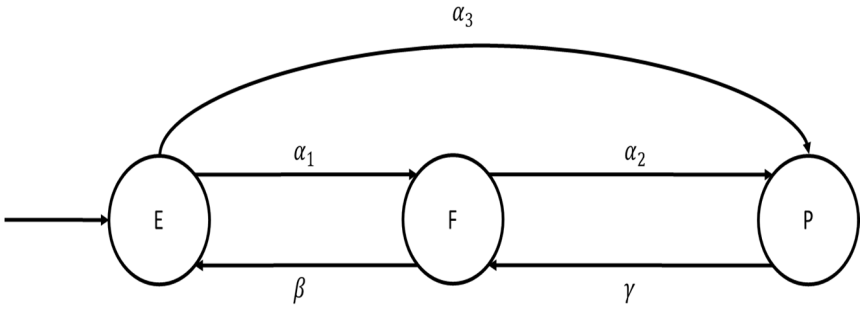

Figure 2.5 Finite automata describing lower back conditions

Lower back conditions are modeled as:

- Excellent (E)
- Fair (F)
- Poor (P)

While the severity of lower back injuries are modeled by the following events:

- Light injury (α_1)
- Moderate injury (α_2)
- Severe injury (α_3)

On the other hand, treatment plans are modeled by the events:

- Physical therapy (β)
- Surgery (γ)

The lower back of a healthy person is described as in excellent condition (E) and when he/she experiences a light injury in the lower back (α_1), his/her lower back condition deteriorates from excellent (E) to fare (F). With physical therapy (β), the lower back

condition can be improved to reach excellent condition (E) again. With a moderate injury (α_2) while in fair condition (F), a person's lower back condition deteriorates to poor (P), which needs surgical intervention (γ) to reach fair condition (F) and further physical therapy (β) to recover back to excellent condition (E). When a person experiences sever injury (α_3) in lower back while his/her lower back condition is excellent (E), his/her lower back condition deteriorates to poor (P). At this state, a surgery (γ) is needed to improve the condition of the lower back to fair (F) followed by physical therapy (β) to reach excellent condition (E).

To effectively model the system in Figure 2.5, vectors and matrices are adopted to represent fuzzy states and fuzzy events as follows:

$$E = [0.9 \quad 0.1 \quad 0]$$

$$F = [0.2 \quad 1 \quad 0.1]$$

$$P = [0 \quad 0.1 \quad 1]$$

$$\alpha_1 = \begin{bmatrix} 0 & 0.85 & 0 \\ 0.1 & 0 & 0.1 \\ 0 & 0.1 & 0 \end{bmatrix}$$

$$\alpha_2 = \begin{bmatrix} 0 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.9 \\ 0 & 0.1 & 0 \end{bmatrix}$$

$$\alpha_3 = \begin{bmatrix} 0.2 & 0.2 & 0.8 \\ 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}$$

$$\beta = \begin{bmatrix} 0.2 & 0.2 & 0.1 \\ 1 & 0 & 0.2 \\ 0 & 0.1 & 0 \end{bmatrix}$$

$$\gamma = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \\ 00.1 & 0.9 & 0.1 \end{bmatrix}$$

If the lower back condition of a person is excellent (E) and that person experiences a severe injury (α_3) in his/her lower back, then the new state of the lower back condition can be obtained by

$$\tilde{q}_{new} = \tilde{\delta}(\tilde{q}, \tilde{\sigma}) = [0.9 \quad 0.1 \quad 0] \circ \begin{bmatrix} 0.2 & 0.2 & 0.8 \\ 0.1 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix} = [0.2 \quad 0.2 \quad 0.8]$$

where $\circ$ is the Max-Min product operator.

The above result can be interpreted as follows:

- 20% → Excellent (E)
- 20% → Fair (F)
- 80% → Poor (P)

This is a typical situation of a real-world problem. Depending on doctor's perception, a specific treatment plan will be followed.

2.4 Fuzzy discrete event system with constraints

In Section 2.3, we showed the effectiveness and practicality of utilizing the FDES in modeling real-world problems with inherent subjectivity and impreciseness. However, FDESs have infinite and continuous state space, which means any sequence of events can occur in the system, which is not the case for many engineering, medicine, and other areas of applications. For example, in a fuzzy system an element of a state vector can take an infinite value between 0 and 1, which results in an infinite set of data that increases the complexity of the modeled system and hence becomes impossible to manage in real-world applications with the current technologies and available resources. Therefore, a crisp DES is introduced to act as a constraint to the fuzzy system. In that way, only sequences of

events that are permissible in the defined constraint system can take place in the fuzzy system. In other words, the language generated by the constraint system dictates the language generated by the FDES.

To model the constraint system, a nondeterministic automata as in Eq. (2.17) is used.

$$G = (Q, \Sigma \cup \{\varepsilon\}, \delta, q_0) \tag{2.17}$$

Where

- Q is the state space
- Σ is the event set corresponding to $\tilde{\Sigma}$
- ε is the empty string
- $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$ is the (nondeterministic) transition function
- $q_0 \subseteq Q$ is the set of initial states

Similar to $\tilde{G}, G$ can be represented in terms of vectors and matrices. If the number of states in G is $m = |Q|$, then a state is represented by an m-dimensional row vector whose elements are in the set $\{0,1\}$ and an event is represented by a $m \times m$ matrix whose elements are in the set $\{0,1\}$.

By combining the constraint system G and the FDES $\tilde{G}$, a FDESwC is obtained as defined in Eq. (2.18).

$$FDESwC = (G, \tilde{G}) \tag{2.18}$$

Example 2.2

This example illustrates the formulation and representation of a FDESwC. For $\tilde{G}$:

Suppose that $\tilde{Q} = [0,1]^n$ with $n = 3$, $\tilde{\Sigma} = \{\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}\}$ where

$$\tilde{\alpha} = \begin{bmatrix} 0.1 & 0.9 & 0.2 \\ 0 & 0.8 & 0.1 \\ 0.3 & 0.75 & 0.1 \end{bmatrix}$$

$$\tilde{\beta} = \begin{bmatrix} 0.1 & 0.1 & 0.8 \\ 0.8 & 0.1 & 0.1 \\ 0.9 & 0 & 0.1 \end{bmatrix}$$

$$\tilde{\gamma} = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.2 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.6 \end{bmatrix}$$

The initial state $\tilde{q}_0 = [0.9 \quad 0.1 \quad 0]$

Transition function is defined as:

$$\tilde{\delta}(\tilde{q}, \tilde{\sigma}) = \tilde{q} \circ \tilde{\sigma}$$

where $\circ$ is the Max-Min operator.

The system illustrated in Figure 2.6 represents the constraint system $G = (Q, \Sigma \cup \varepsilon, \delta, q_0)$, where

$$Q = \{1,2,3,4\}$$

$$\Sigma = \{\alpha, \beta, \gamma\}$$

$$q_0 = \{1\} = [1\ 0\ 0\ 0]$$

ε is the empty string

δ is the transition function

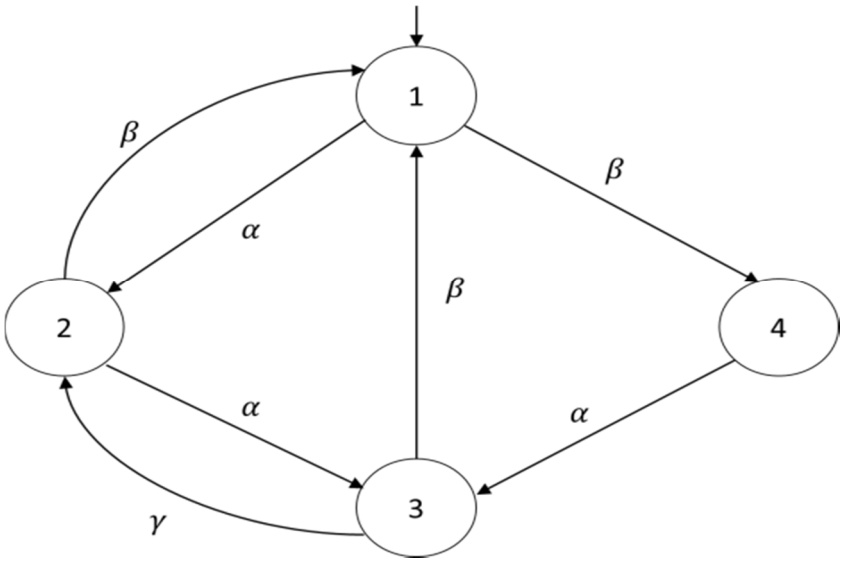

Figure 2.6 Constraint System for Example 2.2

From Figure 2.6, if an event causes a transition from a state *i* to a state *j* in the system, then the corresponding matrix entry $[a_{ij}] = 1$, otherwise $[a_{ij}] = 0$. This way we can obtain the event matrices for the constraint system as follows:

$$\alpha = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\beta = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\gamma = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

If the event α occurs at the initial state, the subsequent state in the constraint system can be obtained as follows:

$$q_{new} = q_0 \times \alpha = [1\ 0\ 0\ 0] \times \begin{bmatrix} 0\ 1\ 0\ 0 \\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0 \\ 0\ 0\ 1\ 0 \end{bmatrix} = [0\ 1\ 0\ 0]$$

where $\times$ is the product operator.

The corresponding transition in the fuzzy system is obtained as follows:

$$\tilde{q}_{new} = \tilde{q}_0 \circ \tilde{\alpha} = [0.9 \quad 0.1 \quad 0] \circ \begin{bmatrix} 0.1 & 0.9 & 0.2 \\ 0 & 0.8 & 0.1 \\ 0.3 & 0.75 & 0.1 \end{bmatrix} = [0.1 \quad 0.9 \quad 0.1]$$

where $\circ$ is Max-Min operator.

After the occurrence of the event $\tilde{\alpha}$ at the initial state, the subsequent state of the fuzzy system is:

- 10% → state 1
- 90% → state 2
- 10% → state 3

DES can be shown as a special case of the FDESwC If the below conditions are satisfied:

- $m = n$
- $\sigma = \tilde{\sigma}$ (in terms of matrix)
- $q_0 = \tilde{q}_0$ (in terms of vector)

Then $FDESwC$ and G become the same system.

CHAPTER 3 DETECTABILITIES OF DISCRETE EVENT SYSTEMS

The detectability problem for DES has been investigated since the mid 2000's. In [29], Shu et al. focused on determining the current state and the subsequent states of a system based on event observations. They defined four types of detectabilities: strong, weak, strong periodic, and weak periodic detectabilities. They derived the necessary and sufficient conditions for these detectabilities by constructing an observer. The system considered in [29] is a deterministic system, which means if a system is at a given state and a given event takes place, then the subsequent state can be uniquely determined. However, not all the systems are deterministic in real-world applications. Therefore, in [25] Shu and Lin extended the detectability definitions of DES to include nondeterministic DESs.

Furthermore, different type of detectabilities have been investigated by many scholars including: *D*-detectability [25], *I*-detectability [28], and delayed detectability [26]. These investigations included different type of discrete event systems, such as deterministic and nondeterministic systems [25], probabilistic discrete event systems [30], and discrete event systems with dynamic event observations [24].

To make detectability definitions more realistic and extend them to DES with partial observations, nondeterministic DES was considered where the system's state can't uniquely be determined. Four types of detectabilities were defined for nondeterministic DES: (weak) detectability, strong detectability, (weak) periodic detectability, and strong periodic detectability [25], [29]. Detectability was extended from deterministic systems to nondeterministic systems. Such a generalization is necessary because there are many systems that need to be modeled as nondeterministic DES. In many applications, we need

to determine the current state of a system and subsequent states based on event observations.

3.1 Extension of detectabilities of discrete event systems

In many biomedical and engineering applications, knowing system state is important to decide next set of actions to be taken. The ability to know the system's state can be referred to as detectability. In this research we extend the detectability definitions for DES to define *N*-detectability for DES. Then, we redefine *N*-detectability to include fuzzy discrete event systems with constraints. We cover both fully observable and partially observable systems

To make detectability meaningful, the following assumptions were made [25], [29]:

Assumptions

1. G is deadlock free, that is, for any state of the system, at least one event is defined at that state:

$$(\forall q \in Q)(\exists \sigma \in \Sigma) f(q, \sigma) \neq \emptyset$$

2. No loops in G contain only unobservable events:

$$\neg(\exists q \in Q)(\exists s \in \Sigma_{uo}^{*}) s \neq \varepsilon \ \wedge q \in f(q, s)$$

Under these assumptions, four types of detectabilities were defined in [25], [29]:

1. Strong Detectability: A DES *G* is strongly detectable if one can determine the current and subsequent states of the system after a finite number of observations for all the trajectories of the system. That is,

$$(\exists n \in N)(\forall s \in L(G)(\forall s' \leq s)|P(s')| > n \Rightarrow |R(q_0, P(s'))| = 1$$

2. Weak Detectability: A DES G is weakly detectable if one can determine the current and subsequent states of the system after a finite number of observations for some trajectories of the system. That is,

$$(\exists n \in N)(\exists s \in L(G)(\forall s' \leq s)|P(s')| > n \Rightarrow |R(q_0, P(s'))| = 1$$

3. Strong Periodic Detectability: A DES G is strongly periodically detectable if one can determine the current state of the system periodically after a finite number of observations for all the trajectories of the system. That is,

$$(\exists n \in N)(\forall s \in L(G)(\forall s' \leq s)(\exists s'' \in \Sigma^*)s's'' \leq s \wedge |P(s'')|$$
$$< n \wedge |R(q_0, P(s'\,s''))| = 1)$$

4. Weak Periodic Detectability: A DES G is weakly periodically detectable if one can determine the current state of the system periodically after a finite number of observations for some trajectories of the system. That is,

$$(\exists n \in N)(\exists s \in L(G)(\forall s\hat{}' \leq s)(\exists s'' \in \Sigma\hat{}\,*)s\hat{}'\,s\hat{}'' \leq s \wedge |P(s\hat{}'')|$$
$$< n \wedge |R(q_0, P(s'\,s''))| = 1)$$

In nondeterministic DES, some events may not observable. Therefore, we may not know the exact state of the system after observing an event sequence $t \in P(L(G))$. However, we can determine the set of states in which the system is in. This set of states is called the state estimate (SE), which is used to reformulate the detectability definitions in [25], [29].

Formally, if $t \in P(L(G))$ is observed, then the state estimate $SE(t)$ is given by Eq. (3.1)

$$SE(t) = \{q \in Q: (\exists s \in L(G))\delta(q_0, s) = q \wedge P(s) = t\} \tag{3.1}$$

If $SE(t)$ is a singleton, that is, the number of states in *SE* is equal to 1 (denoted as $|SE(t)| = 1$), then $SE(t)$ is said to be certain and hence the state of the system can be exactly determined. Mathematically for DES represented by matrices and vectors, certainty of a state estimate is defined as in Eq. (3.2).

$$cert(SE) = \frac{Max(e_1, e_2, \dots, e_n)}{\sum_i^n e_i} \geq 1 \tag{3.2}$$

where $e \in SE$ and n is the number of elements in the state estimate vector.

For example, if $SE(t) = [0\ 0\ 1\ 0]$, then $cert(SE(t)) = \frac{Max(0,0,1,0)}{0+0+1+0} = 1$ and hence $SE(t)$ is certain. On the other hand, if $SE(t) = [0\ 0\ 1\ 1]$, then $cert(SE(t)) = \frac{Max(0,0,1,1)}{0+0+1+1} = 0.5$ and consequently $SE(t)$ is considered not certain.

Using state estimates, detectability definitions can be reformulated as follows:

1. Strong Detectability: A DES *G* is strongly detectable if and only if, for all the trajectories of the system, the current and future state estimates of the system are certain after a finite number of observations.
2. Weak Detectability: A DES *G* is weakly detectable if and only if, for some trajectories of the system, the current and future state estimates of the system are certain after a finite number of observations.
3. Strong Periodic Detectability: A DES *G* is strongly periodically detectable if and only if, for all the trajectories of the system, the current state estimates of the system are certain periodically after a finite number of observations.

4. Weak Periodic Detectability: A DES G is weakly periodically detectable if and only if, for some trajectories of the system, the current state estimates of the system are certain periodically after a finite number of observations.

States involved in loops are of particular interest because the observer can stay in these states forever. Therefore, let us denote the set of all loops in the observer as

$$Loop = \{(z,u) \in Z \times \Sigma_o^* : |u| \geq 1 \wedge \xi(z,u) = z\}$$

In [25], it was proven that a nondeterministic DES G is strongly detectable with respect to P if and only if in observer G_{obs}

$$(\forall (z,u) \in Loop)(\forall w \in \Sigma_o^*)\xi(z,w) \in Z_m$$

where Z_m is the set of certain states.

That is, any state reachable from any loop in G_{obs} is in Z_m.

Also, it was proven that a nondeterministic DES G is weakly detectable with respect to P if and only if in observer G_{obs}

$$(\exists (z,u) \in Loop)(\forall w \leq u)\xi(z,w) \in Z_m$$

That is, there are loops in G_{obs} that are entirely within Z_m.

For periodic detectability, it was proven that a nondeterministic DES G is strongly periodically detectable with respect to P if and only if in observer G_{obs}

$$(\forall (z,u) \in Loop)(\exists w \leq u)\xi(z,w) \in Z_m$$

That is, all loops in G_{obs} must include at least one state that belongs to Z_m.

Also, it was proven that a nondeterministic DES G is weakly periodically detectable with respect to P if and only if in observer G_{obs}

$$(\exists (z,u) \in Loop)(\exists w \leq u)\xi(z,w) \in Z_m$$

That is, there are loops in G_{obs} which include at least one state that belongs to Z_m.

3.2 *N*-detectability of discrete event systems

In crisp DES, a state estimate can involve only a finite number of states (it is bounded by 2^N, where *N* is the number of states in the system), which are all possible states that the system can be in simultaneously. States within a state estimate range from $[1\ 0\ 0\ \ldots 0]$ to $[1\ 1\ 1\ \ldots 1]$. This is not true in fuzzy discrete event systems, where an element in a state estimate can take unlimited number of degrees of membership to each state in the system, given the fact that each element in the state estimate can have a value ranging from 0 to 1. This imposes a significant problem for defining fuzzy detectability. To alleviate the problem and properly extend detectability of DES to detectability of FDES, we will construct an *N*-step look-ahead tree and investigate the detectability of each branch in the tree. We will first investigate *N*-detectabilities for crisp discrete event systems and then generalize it to include fuzzy discrete event systems.

Consider all sequences of events with the number of observed event being less than or equal to the depth N and construct *N*-step look-ahead tree. Denote the tree as in Eq. (3.3).

$$Tree(G, N) = (X, \Sigma_o, \xi, x_o) \tag{3.3}$$

where

- N is the depth or the length of observable string
- ξ is the transition function
- X is the set of nodes in the tree
- Σ_o is the set of observable events

- x_o the root of the tree

Nodes in X can be partitioned according to the levels: $X = X_0 \cup X_1 \cup X_2 \cup \ldots \cup X_N$, where X_i is the set of level *i* nodes. In particular, X_0 has only one node, that is, the root node. X_N contains all the leaf nodes and can be reached after the occurrence of an event sequence of *N* observable events. Each leaf node in X_N corresponds to a branch or a trajectory in the *N*-Step look-ahead tree. Denote the set of branches or trajectories by Eq. (3.4).

$$Trj(G, N) = \{trj_1, trj_2, \ldots, trj_k\} \tag{3.4}$$

where k is the maximum number of branches in the *N*-step look-ahead tree, $k \leq |\Sigma|^N$.

By considering the assumptions above, all nodes in the tree, except the leaf nodes, have at least one continuation (transition).

Each node in $x \in (X - X_0)$ corresponds to an observed string, denoted by $\xi^{-1}(x)$. Note that $\xi^{-1}(x) \in P(L(G))$. Hence, the state estimate for $\xi^{-1}(x)$, denoted by $\mathrm{SE}(\xi^{-1}(x))$, is defined according to Equation (3.1). We say that a node $x \in X$ is certain if $\mathrm{SE}(\xi^{-1}(x))$ is certain, that is, $|\mathrm{SE}(\xi^{-1}(x))| = 1$.

Example 3.1

The following example illustrated by Figure 3.1 represents an automaton for DES with 3 states and 3 events.

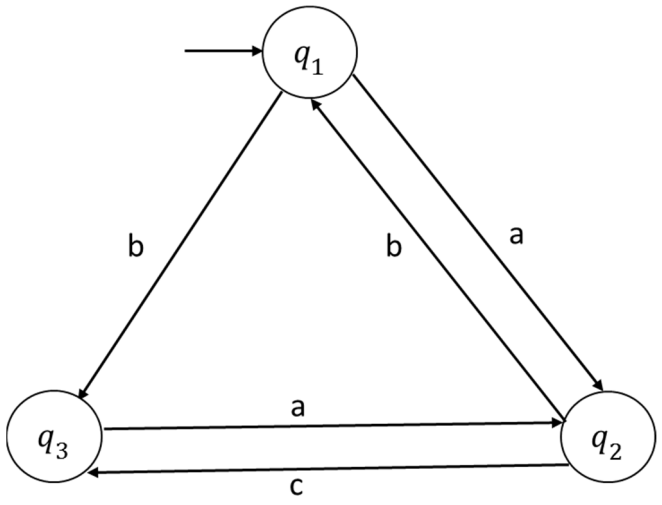

Figure 3.1 Example 3.1 automata with 3 states and 3 events

Given $\Sigma = \{a, b, c\}$, $\Sigma_o = \{a, b\}$, and $\Sigma_{uo} = \{c\}$

where

$$q_0 = [1 \quad 0 \quad 0]$$

$$q_2 = [0 \quad 1 \quad 0]$$

$$q_0 = [0 \quad 0 \quad 1]$$

$$a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

For a string $t = aba$ where $t = P(s)$ which is the observable string of s .Therefore $P^{-1}(t) = \{aba, abaaba, abaabaaba, \ldots abac, abaabac, \ldots\} = (aba)^n + (aba)^n c$, where$n \geq 1$.

$$SE(t) = \delta(q_0, s) \;=\; \delta(q_0, (aba)^n) \;\cup \delta(q_0, (aba)^n c)$$

$$= Logical([0 \quad 1 \quad 0] + [0 \quad 0 \quad 1]) = [0 \quad 1 \quad 1]$$

where $Logical(x) = \begin{cases} 0, x = 0 \\ 1, x \geq 1 \end{cases}$

N-step tree realization:

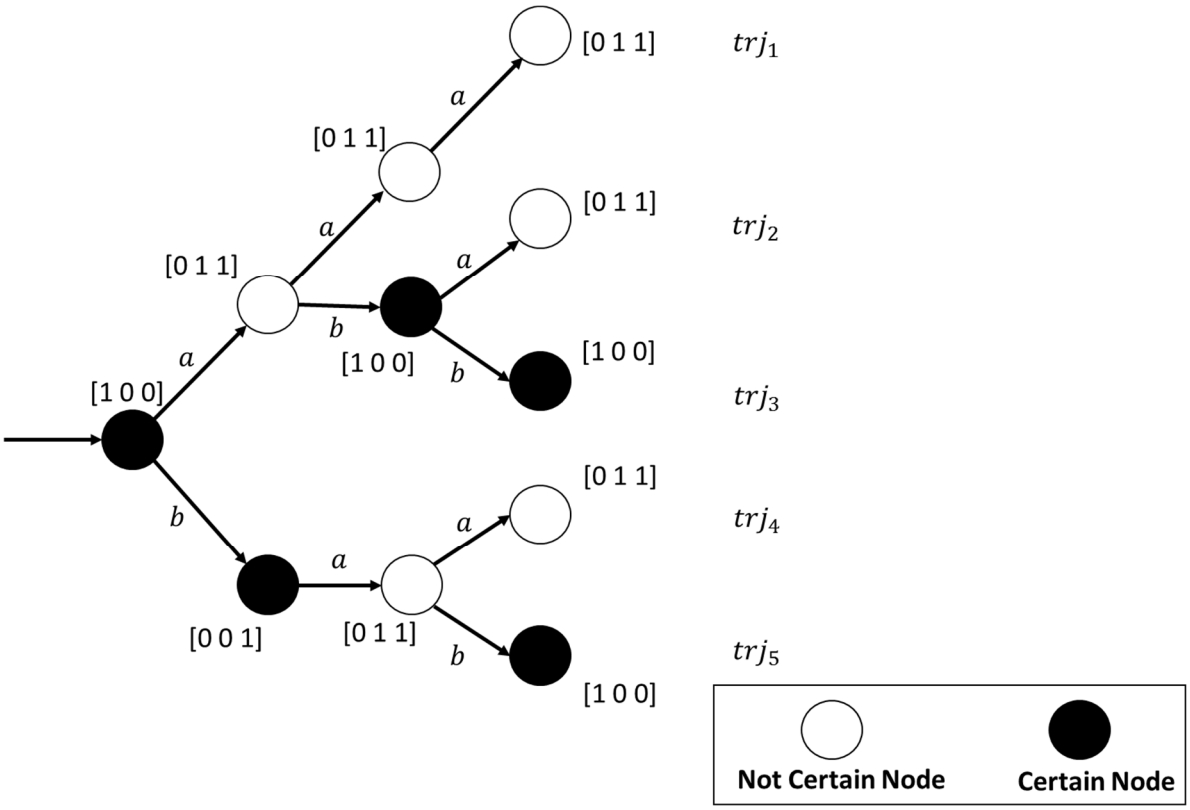

Figure 3.2 N-step look-ahead tree for Example 2.2

The tree in Figure 3.2 is constructed from the initial state q_0. The root of the tree x_0 is defined in terms of the unobservable reach ($UR(.)$) that can be obtained by Eq. (3.5).

$$x_0 = UR(q_0) = \{q \in Q: (\exists q' \in q_0) q \in \delta(q', \varepsilon)\} \tag{3.5}$$

If an event σ occurs at any of the states $q' \in x_0$, the tree extends as in Eq. (3.6).

$$\xi(x_0, \sigma) = UR(\{q \in Q: (\exists q' \in x_0) q \in \delta(q', \sigma)\}) \tag{3.6}$$

To construct the N-step look-ahead tree in Figure 3.2, we will start with the root node that can be obtained as defined in Eq. (3.5). Since there is no unobservable event defined in the initial state q_1, the root node x_0 is obtained as

$$x_0 = UR(q_1) = \{q_1\}$$

which is the state vector $[1 \quad 0 \quad 0]$.

Upon the observation of the event a at the initial state q_1, the look-ahead tree extends as defined in Eq. (3.6)

$$q_{next} = \xi(x_0, a) = UR\big(\delta(q_1, a)\big) = UR(q_2)$$

Since q_2 has the unobservable event c defined, then q_{next} is obtained as below:

$$q_{next} = q_2 \cup \delta(q_2, c) = q_2 \cup q_3 = [0 \quad 1 \quad 0] + [0 \quad 0 \quad 1] = [0 \quad 1 \quad 1]$$

Similarly, the look-ahead tree can be extended to all branches.

After setting $N = 3$, the constructed *N*-step look-ahead tree in Figure 3.2 has 5 trajectories, each of them consists of 3 nodes. These trajectories are:

$$trj_1 = \{[0 \quad 1 \quad 1], [0 \quad 1 \quad 1], [0 \quad 1 \quad 1]\}$$

$$trj_2 = \{[0 \quad 1 \quad 1], [1 \quad 0 \quad 0], [0 \quad 1 \quad 1]\}$$

$$trj_3 = \{[0 \quad 1 \quad 1], [1 \quad 0 \quad 0], [1 \quad 0 \quad 0]\}$$

$$trj_4 = \{[0 \quad 0 \quad 1], [0 \quad 1 \quad 1], [0 \quad 1 \quad 1]\}$$

$$trj_5 = \{[0 \quad 0 \quad 1], [0 \quad 1 \quad 1], [1 \quad 0 \quad 0]\}$$

After Appling certainty criterion on each node of the tree, we can conclude that the solid filled nodes in Figure 3.2 are certain while the others are not.

3.3 Branch detectability in discrete event systems

In *N*-step look-ahead tree, a branch detectability is defined as follows:

- Detectable: a branch said to be detectable if all the last *D* nodes within the branch are certain
- Periodically detectable: a branch said to be periodically detectable if some the last *D* nodes within the branch are certain
- Not detectable: a branch said to be not detectable if all the last *D* nodes within the branch are not certain

In the above example, with *D=50%* only trj_3 and trj_5 is detectable.

Based on branch detectability of DES, *N*-detectabilities for DES are defined as follows:

1. Strong *N*-Detectability: A DES G is strongly *N*-detectable with respect to D if all the branches in $Trj(G,N)$ are detectable with respect to D.
2. Weak *N*-Detectability: A DES G is weakly *N*-detectable with respect to D if some branches in $Trj(G,N)$ are detectable with respect to D.
3. Strong Periodic *N*-Detectability: A DES G is strongly periodically *N*-detectable with respect to D if all the branches in $Trj(G,N)$ are periodically detectable with respect to D.
4. Weak Periodic N-Detectability: A DES G is weakly periodically N-detectable with respect to D if some branches in $Trj(G,N)$ are periodically detectable with respect to D.

By extending *N*-Detectability of DES to fuzzy discrete event system with constraints, detectabilities of FDES will be established.

Theorem 3.1

For a DESG, let $N > 2^{|Q|}/(1-D)$. G is strongly detectible (or detectable, strongly periodically detectable, periodically detectable) if and only if G is strongly *N*-detectable (or *N*-detectable, strongly periodically *N*-detectable, periodically *N*-detectable).

We prove the result for strong detectability. Proofs for others are similar.

It is proved in [25] that G is strongly detectable if and only if in the observer of G, all the states accessible from any loop are certain. From the construction of $Trj(G,N)$, all the nodes in the tree correspond to some states in the observer of G.

Since the maximal number of states in the observer of G is $2^{|Q|}$, any trajectory of length $2^{|Q|}$ will already reach a loop in the observer of G.

(Only if) Assume that G is strongly detectable. Then for any trajectory of length $N > 2^{|Q|}/(1-D)$, the last D of its nodes are at least $2^{|Q|}$ transitions away from the initial state, regardless of D value ($0\% < D < 100\%$). Therefore, they must be accessible from a loop and hence are certain. In other words, G is strongly *N*-detectable with respect to D.

(If) Assume that G is not strongly detectable. Then there exists at least one loop in the observer of G whose states are not all certain. Let b be the trajectory starting from the initial state of the observer, reaching the loop and then moving along the loop until it reaches the length of $N > 2^{|Q|}/(1-D)$. Clearly, for b, the last D of its nodes are in the loop and some of them are not certain. Therefore, G is not strongly *N*-detectable with respect to D.■

Example 3.2

Consider the automation shown in Figure 3.1 to represent G with $\Sigma = \{a, b, c\}$, $\Sigma_o = \{a, b\}$, and $\Sigma_{uo} = \{c\}$. To better demonstrate theorem 3.1, we first enumerate the automation as in Figure 3.3 and then construct the observer for DES G as shown in Figure 3.4 to check for detectability.

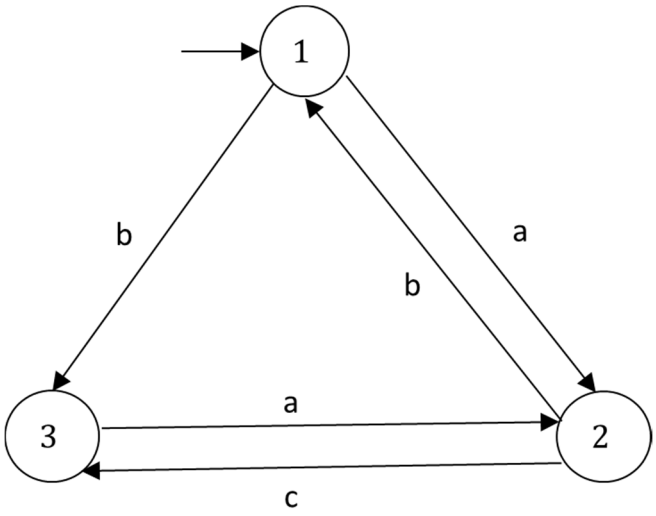

Figure 3.3 Example 3.2 automata with 3 enumerated states

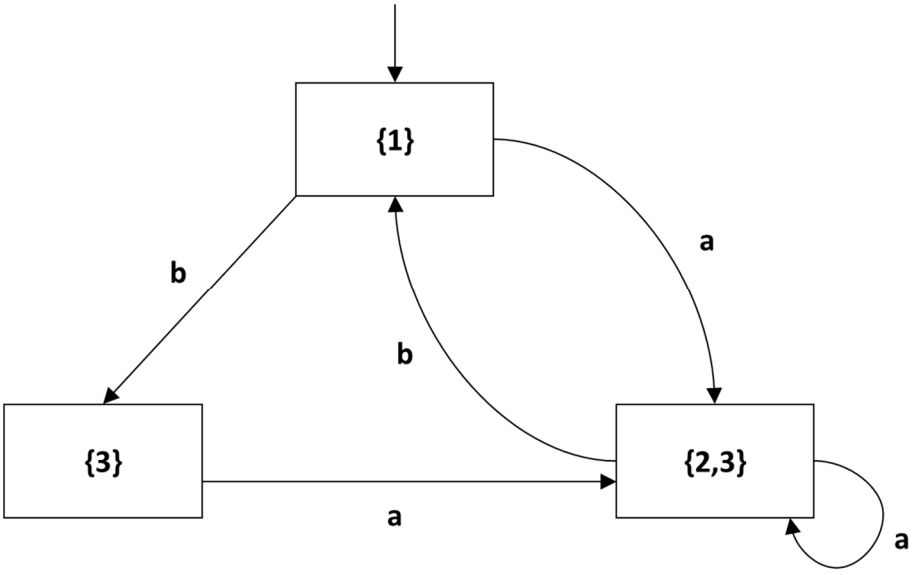

Figure 3.4 Observer automata for Figure 3.3

The observer in Figure 3.4 has 3 states, which means for any event sequence of a length longer than $2^{|Q|}$(which is equal to 8), the system will make a loop within itself. If the system is strongly/weakly detectable, then any node within/accessible from a loop is singleton for all/some trajectories of the system. If the system is strongly/weakly periodically detectable, then some nodes within a loop are singleton for all/some trajectories of the system.

For instance, at the initial state 1, if the event sequence $babaaaabbaba$ occurs, the system will make a transition to states $\{2,3\}$, which is not singleton. If we continue down this path to the event sequence $babaaaabbababb$, then the system will arrive at state 3, which itself is singleton.

For *N*-detectability, let's extend the branches of the tree in Figure 3.2 to reach the depth of 16 observable events or more ($D = 50\%$). Let's take trj_5 in Figure 3.2 as an example and extend it to the 16th observable event as in the event sequence $babaaaabbababbaa$. The last 50% nodes of trj_5 are included in the below set of nodes:

$$trj_{5_{>50\%}} = \begin{Bmatrix} [0 & 0 & 1], [0 & 1 & 1], [1 & 0 & 0], [0 & 1 & 1], \\ [1 & 0 & 0], [0 & 0 & 1], [0 & 1 & 1], [0 & 1 & 1] \end{Bmatrix}$$

Each member of the above set of nodes is a state estimate that corresponds to a state in the observer. As can be noticed, some state estimates are certain and some are not, which is the same conclusion we reached using the observer.

Similarly, we can show that for a given *N*, if the DES *G* strongly detectible (or detectable, strongly periodically detectable, periodically detectable), then it is strongly *N*-

detectable (or N-detectable, strongly periodically N-detectable, periodically N-detectable) and vice versa.

CHAPTER 4 DETECTABILITIES OF FUZZY DISCRETE EVENT SYSTEMS

4.1 Detectabilities of fully observable fuzzy discrete event systems

In Section 2.3, we introduced fuzzy discrete event systems and presented a model using fuzzy automaton as follows.

$$\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{q}_0)$$

The elements of $\tilde{G}$ are defined as follows. $\tilde{Q} = [\,0,1]^n$ is the state space, where n is the dimension of the state space. In other words, a state is represented by an n-dimensional (row) vector. When the system starts, it is in its initial states, which is denoted by $\tilde{q}_0 \in \tilde{Q}$. As events occur, the system will change states. The set of events are denoted by $\tilde{\Sigma}$. An event $\tilde{\sigma}$ is represented by an $n \times n$ matrix with elements in the interval [0, 1] (real numbers between 0 and 1). How the system changes from one state to another after the occurrence of an event is specified by the transition function $\tilde{\delta}: \tilde{Q} \times \tilde{\Sigma} \to \tilde{Q}$. In other words, if at the current state $\tilde{q}$, event $\tilde{\sigma}$ occurs, then the next state is $\tilde{\delta}(\tilde{q}, \tilde{\sigma})$. We let $\tilde{\delta}(\tilde{q}, \tilde{\sigma}) = \tilde{q} \circ \tilde{\sigma}$, where $\circ$ denotes a fuzzy reasoning operator. For example, we can let $\circ$ be the max-product or Max-Min operator. The theoretical results in this study, including all the theorems, are true for any fuzzy reasoning operator.

The fuzzy discrete event systems that are considered in the literature so far are systems without constraints. In a fuzzy discrete event system without constraints, any sequences of events can occur in the system. In many applications, not all sequences of events can occur; rather, only some sequences of events can occur. To make our results

more applicable, in this research, we consider fuzzy discrete event systems with constraints. We use a nondeterministic automaton to specify constraints of the system [7]:

$$G = (Q, \Sigma \cup \{\varepsilon\}, \delta, q_0).$$

where the elements Q,Σ,δ, q_0 are the same elements defined in Eq.(2.17)

The transition function δ can be extended to strings $\delta: Q \times (\Sigma \cup \{\varepsilon\})^* \to 2^Q$ in the usual way [3]. We say that $\delta(q, s)$ is defined, denoted by $\delta\ (q, s)!$, if $\delta(q, s) \neq \emptyset$. G can also be represented by vectors and matrices. Let the number of states in G be $m = |Q|$. Then a state can be represented by a vector $q \in \{0,1\}^m$and an event can be represented by a $m \times m$ matrix with elements in the se t $\{0,1\}$.

As in Eq. (2.18), a fuzzy discrete event system with constraints is then given by

$$FDESwC = (\tilde{G}, G)$$

$FDESwC$ is constrained in the sense that only sequences of events in $L(G)$ can occur in the fuzzy system.

To study *N*-detectability for the fully observed *FDESwC*, we need to construct an *N*-step look ahead tree from the initial state of the constraint system q_0. This tree is denoted by

$$Tree(G, N) = (X, \Sigma, \xi, x_0)$$

Where

- X is the set of the nodes of the tree
- Σ is the set of the events
- ξ is the transition function
- x_0 is the root of the tree and can be obtained by Eq. (3.5)

For any event sequence occurs at the initial states, the tree extends as in Eq. (3.6).

In addition to $Tree(G,N)$ construction, the extension of *N*-detectability to $FDESwC$ requires construction of a fuzzy tree $FTree(G,N) = (X, \Sigma, \xi, x_0, f)$, where X, Σ, ξ, and x_0 are the same entities as in $Tree(G,N)$ defined for the constraint system. The additional element f maps each node $x \in X$ to the corresponding fuzzy state $f(x)$. $f(x)$ is represented by a row vector whose elements are in [0,1]. x_0 is given by

$$\tilde{x}_0 = f(x_0) = \tilde{q}_0$$

if $x' = \xi(x, \sigma)$, then

$$f(x') = f(x) \circ \tilde{\sigma}$$

where "$\circ$" is a fuzzy reasoning operator (e.g., Max-Min operator).

The fuzzy node $f(x)$ in the fuzzy tree $FTree(G,N)$ is calculated by merging all the fuzzy states that can be reached after observing an event sequence at the initial states. This way we calculate all nodes in $FTree(G,N)$ and construct the fuzzy tree's branches described by

$$Trj(G,N) = \{trj_1, trj_2, \dots, trj_k\}$$

For the node $f(x) = [z_1\ z_2 \dots z_n]$, where $z_n \in [0,1]$, $f(x)$ is certain with respect to τ if the following condition is true:

$$\frac{max\{z_1, z_2, \dots, z_n\}}{\sum_{i=1}^{n} z_i} \geq \tau$$

where τ is a threshold in [0,1].

Example 4.1

Consider the FDES system defined in terms of fuzzy state set $\tilde{Q}$ ={$\tilde{q}_1, \tilde{q}_2, \tilde{q}_3$} and fuzzy event set $\tilde{\Sigma} = \{\tilde{a}, \tilde{b}, \tilde{c}\}$, where

$$\tilde{a} = \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix}$$

$$\tilde{b} = \begin{bmatrix} 0.1 & 0.1 & 0 \\ 0.95 & 0 & 0 \\ 0.9 & 0.2 & 0.15 \end{bmatrix}$$

$$\tilde{c} = \begin{bmatrix} 0.1 & 0.1 & 0 \\ 0.85 & 0 & 0 \\ 0.7 & 0.2 & 0.1 \end{bmatrix}$$

The constraint system is illustrated by the diagram of Figure 4.1. Take threshold τ =0.67, N =4, and $q_0 = [1\ 0\ 0]$.

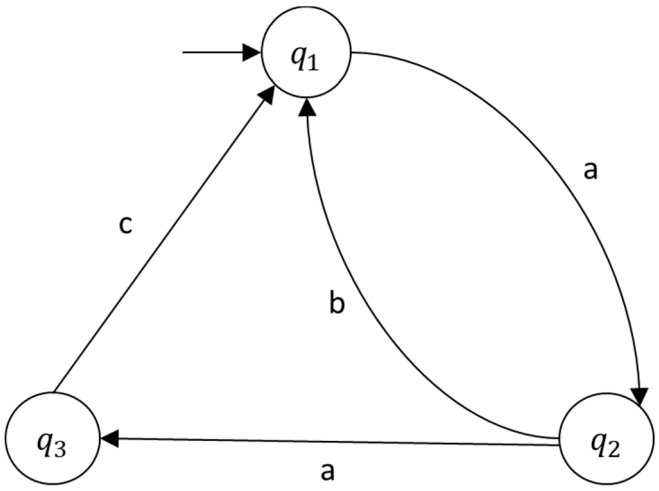

Figure 4.1 Constraint system for Example 4.1

Event matrices for the constraint system can be obtained from Figure 4.1 as:

$$a = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$b = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$c = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

From given constraint system shown in Figure 4.1, the language generated by G with $N \leq 4$ is

$$L(G) = \{\varepsilon, a, aa, ab, aac, aba, aaca, abaa, abab\}$$

Similarly, the fuzzy system will be constrained by the constraint system shown in Figure 4.1, that is

$$L(\tilde{G}) = \{\varepsilon, \tilde{a}, \tilde{a}\tilde{a}, \tilde{a}\tilde{b}, \tilde{a}\tilde{a}\tilde{c}, \tilde{a}\tilde{b}\tilde{a}, \tilde{a}\tilde{a}\tilde{c}\tilde{a}, \tilde{a}\tilde{b}\tilde{a}\tilde{a}, \tilde{a}\tilde{b}\tilde{a}\tilde{b}\}$$

The realized fuzzy tree $Tree(G, N) = (X, \Sigma_o, \xi, x_o)$ is shown by the diagram of Figure 4.2

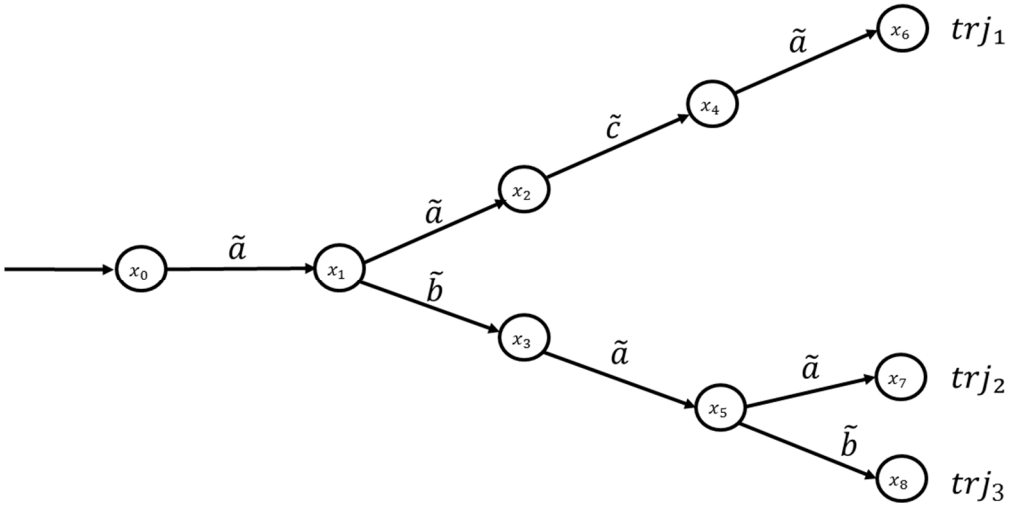

Figure 4.2 Trajectories of Example 4.1

From Figure 4.2.

$$Trj(G,N) = \{trj_1, trj_2, trj_3\}$$

trj_1 consists of x_1, x_2, x_4 and x_6 nodes and can be represented by the following matrix:

$$trj_1 = \begin{bmatrix} x_1 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix}$$

Similarly:

$$trj_2 = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix}$$

$$trj_3 = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_8 \end{bmatrix}$$

Since the system is fuzzy in nature and all events' matrices have elements ranging between 0 and 1(fuzzy events), elements in trajectory matrix (nodes) i.e. x_i is the state estimate which corresponds to the current state of the system after observation of a fuzzy event sequence represented by $\xi^{-1}(x_i)$, and hence:

$$x_1 = SE(\xi^{-1}(x_1)) = SE(\tilde{a}) = \xi(q_0, \tilde{a}) = [1 \quad 0 \quad 0] \circ \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix}$$

$$= [0.1 \quad 0.95 \quad 0]$$

$$x_2 = SE\left(\xi^{-1}(x_2)\right) = SE(\tilde{a}\tilde{a}) = \xi(q_0, \tilde{a}\tilde{a})$$

$$= [1 \quad 0 \quad 0] \circ \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix} = [0.2 \quad 0.2 \quad 0.9]$$

$$x_4 = SE\left(\xi^{-1}(x_4)\right) = SE(\tilde{a}\tilde{a}\tilde{c}) = \xi(q_0, \tilde{a}\tilde{a}\tilde{c})$$

$$= [1 \quad 0 \quad 0] \circ \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix}$$

$$\circ \begin{bmatrix} 0.1 & 0.1 & 0 \\ 0.85 & 0 & 0 \\ 0.7 & 0.2 & 0.1 \end{bmatrix} = [0.7 \quad 0.2 \quad 0.1]$$

$$x_6 = SE\left(\xi^{-1}(x_6)\right) = SE(\tilde{a}\tilde{a}\tilde{c}\tilde{a}) = \xi(q_0, \tilde{a}\tilde{a}\tilde{c}\tilde{a})$$

$$= [1 \quad 0 \quad 0] \circ \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix}$$

$$\circ \begin{bmatrix} 0.1 & 0.1 & 0 \\ 0.85 & 0 & 0 \\ 0.7 & 0.2 & 0.1 \end{bmatrix} \circ \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.1 & 0.1 & 0 \end{bmatrix} = [0.2 \quad 0.7 \quad 0.2]$$

where "$\circ$" is a fuzzy operation and can be product operation or Max-Min operation. In this example Max-Min operation was used which resulted in below trajectories:

$$trj_1 = \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.2 & 0.2 & 0.9 \\ 0.7 & 0.2 & 0.1 \\ 0.2 & 0.7 & 0.2 \end{bmatrix}$$

$$trj_2 = \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.95 & 0.1 & 0 \\ 0.1 & 0.95 & 0.1 \\ 0.2 & 0.2 & 0.9 \end{bmatrix}$$

$$trj_3 = \begin{bmatrix} 0.1 & 0.95 & 0 \\ 0.95 & 0.1 & 0 \\ 0.1 & 0.95 & 0.1 \\ 0.95 & 0.1 & 0.1 \end{bmatrix}$$

In the above trajectory matrices, the order of the row corresponds to the node level of the trajectory, for example, the third row in trj_3 corresponds to x_5 node in Figure 4.2 ,

which is level 3 node and the forth one corresponds to level 4 node (x_8) and in this case it is the leaf of trj_3 as illustrated in Figure 4.2.

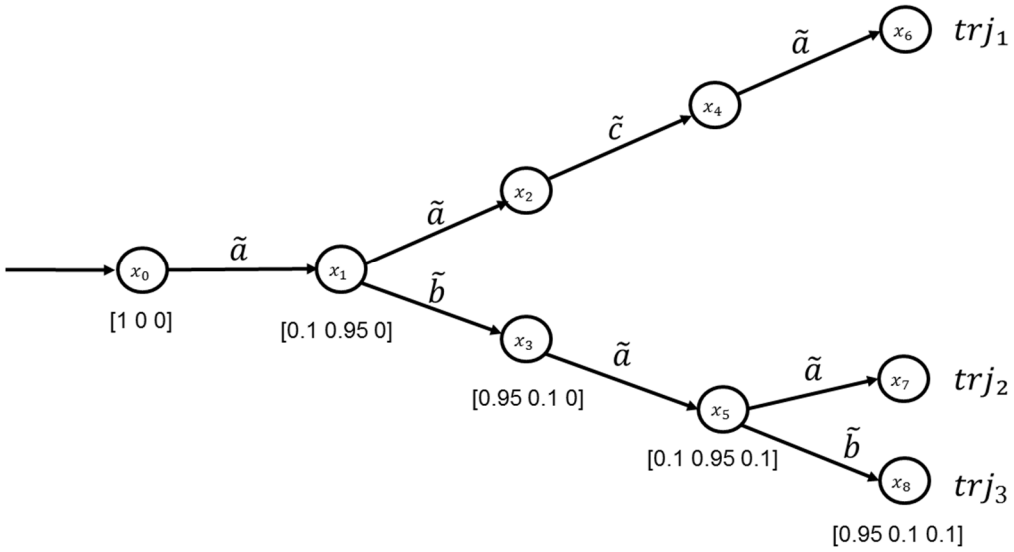

Figure 4.3 Estate estimates for nodes in trj3 of Example 4.1

A branch $b \in Trj(G, N)$ is regarded as detectable with respect to D and τ if all of the last D (which is a percentage number) nodes are certain with respect to τ. Also, b is periodically detectable with respect to D and τ if some of the last D nodes are certain with respect to τ.

Consequently, *N*-detectabilities of $FDESwC$ are defined as follows:

1. Strong *N*-Detectability: A $FDESwC(G, \tilde{G})$ is strongly *N*-detectable with respect to D and τ if all the branches in $Trj(G, N)$ are detectable with respect to D and τ.

2. Weak *N*-Detectability: A $FDESwC(G, \tilde{G})$ is weakly *N*-detectable with respect to D and τ if some branches in $Trj(G, N)$ are detectable with respect to D and τ.
3. Strong Periodic *N*-Detectability: A$FDESwC(G, \tilde{G})$ is strongly periodically *N*-detectable with respect to D and τ if all the branches in $Trj(G, N)$ are periodically detectable with respect to D and τ.
4. Weak Periodic *N*-Detectability: A $FDESwC(G, \tilde{G})$ is weakly periodically *N*-detectable with respect to D and τ if some branches in $Trj(G, N)$ are periodically detectable with respect to D and τ.

For better accuracy the branch length *N* should be long enough to allow the system to make at least one loop around itself. This way the certainty criterion can be applied to more nodes along a certain path and consequently increase the chance of finding more certain nodes. By choosing $N \geq 2^{|Q|}/(1 - D)$, the chance of finding a given branch detectable will increase. On the other hand, *D* identifies the set of nodes along a branch to be checked for certainty. The bigger is the *D*, the more likely to find certain nodes along the branch and consequently improve the branch detectability.

The threshold τ is very important in the defuzzification process. It directly affects the certainty outcome for a given node and the branch detectability as well. Example 4.2 clearly shows how increasing the threshold will decrease the possibility for a given node to be certain. Therefore, τ should be application dependent and its selection should be determined in collaboration with experts in that field.

Construction of a look-ahead tree for real-world applications requires reliable and efficient computer programs to handle its lengthy calculations and complexity. Therefore, a Computer program has been developed to construct a look-ahead tree for both the constraint system and the fuzzy system. In addition, the computer program is used to calculate the state estimate of each node in the tree and finally check it for detectability. Matrix computations techniques and algorithms were adopted to calculate estate estimates and hence apply developed criterion do decide detectability classes.

4.2 Implementations and computer programming

A computer program is developed using M-script language to perform calculations for state estimate in addition to the construction of the look-ahead tree for both the constrained system and fuzzy system and the determination of system detectability.

As stated in above sections, FDESwC is actually a composition of a crisp system, which regarded as a constraints system, and the fuzzy system. The crisp system has the same number of events as in FDES but not necessary to have the same number of states. This way, FDES will allow only the event sequences that are permissible in the constraint system.

To extend detectability definitions from conventional DES to FDES the below parameters were introduced:

1- Detectability threshold

   Detectability Threshold is the threshold ranging between 0 and 1 and based on which the certainty of the state estimate can be determined.

2- Detectability Checker

Detectability checker is the function that inspects the elements of a fuzzy state estimate and compares it against detectability threshold to decide membership of each state within state estimate.

$$D_{ch} = \begin{cases} 1 \; if \; e \geq thr \\ 0 \; otherwise \end{cases}$$

where e is the state's element and thr is the threshold.

3- Master Event (ME)

Master event is the term used to reference to all possible combinations of events that can be generated by FDES system (Kleene closure). Since the master of event is infinite, it is imperative to restrict the ME that can be generated by a system. Defining a depth (size of string) and enforcing constraints are the procedures adopted to make ME finite.

Algorithm 4.1 is developed to perform the followings:

- Calculates the maximum number of trajectories that can be constructed in the *N*-step look-ahead tree of the constraint system based on the number of the system's states and the given depth *N*
- Predicts the next event to take place and calculates the state estimate
- Constructs all trajectories in *N*-step look-ahead tree for FDES
- Checks all the nodes in *N*-step look-ahead fuzzy tree for certainty based on the depth and the given threshold
- Determines the detectability classification for all the trajectories of the fuzzy tree
- Determines the detectability classification for the FDES

To explain the next occurring event prediction algorithm, let's consider the constraints system of Figure 4.1 , where the event set $\Sigma = \{a, b, c\}$ is stored in an array whose first, second and third elements are the events a,b, and c respectively. The master event ME is constructed as follows:

The set of the events of length 1

$$E_1 = \{a, b, c\}$$

The set of the events of length 2

$$E_2 = \{aa, ab, ac, ba, bb, bc, ca, cb, cc\}$$

The set of the vents of length 3

$$E_3 = \{aaa, aab, aac, aba, abb, abc, aca, acb, acc,$$
$$baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc,$$
$$caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc\}$$

This way ME is expanded to *N* number of events. The event sequences in the set E_N represent the trajectories of the system, which are of a great importance in the definition of *N*-detectabilities.

The maximum number of trajectories that can be constructed for a given system is obtained by

$$Trj_{max} = |\Sigma|^N$$

For $N = 3$ and $|\Sigma| = 3$

$$Trj_{max} = 3^3 = 27$$

The number of trajectory groups $No\_TrjGroups$ is equivalent to the size of the event set.

In this example

$$No\_TrjGroups = |\Sigma| = 3$$

The below tables represent the trajectory groups of the system for $N = 3$. To obtain the trajectory group Trj_group for a given trajectory, the trajectory index Trj_{index} is divided by the number of trajectories within the trajectory group $No\_TrjInGroup$, which is obtained by

$$Trj\_group = Trj_{index}/No\_TrjInGroup$$

where

$$No\_TrjInGroup = Trj_{max}/|\Sigma|$$

In this example $No\_TrjInGroup = \frac{27}{3} = 9$

If the Trj_group is a fraction, then it will be rounded to the closest higher integer. The number of times an event is repeated at the same level (depth) in a trajectory group is referred to as the repeat factor RpF, which is obtained by

$$RpF = \frac{No\_TrjInGroup}{|\Sigma|^{(depth-1)}}$$

To calculate the event index recursively, the level transition element Lvl_Trns is calculated to be used in next level (depth) calculations and is obtained by

$$Lvl\_Trns = rem(Trj_{index}, RpF)$$

where

$rem(A, B)$ is the remainder of the operation $\frac{A}{B}$.

In this example, for trj_{12}, the trajectory group Trj_group and Lvl_Trns are obtained as follows:

$$Trj\_group = 12/9 = 1.333 \approx 2$$

For depth=1

$$Lvl\_Trans = rem(12,9) = 3$$

The first event in an event sequence is always the same for a given Trajectory group.

In this example the index of the first element is obtained as follows

$$Event\_Index = Trj_{group} = 2$$

which is event b

Table 4.2 shows that event b is always the first event in all event sequences in the table.

For depth=2

$$RpF = \frac{No\_TrjInGroup}{|\Sigma|^{(depth-1)}}$$

In this example, for trj_{12}

$$RpF = \frac{9}{|3|^{(2-1)}} = 3$$

Based on the RpF, Lvl_Trns calculate in previous level, and the $depth$, the event index can be obtained as follows

if $\frac{Lvl\_Trns\_priv}{RpF} > 0$

$$Event\_Index = \frac{Lvl\_Trns\_priv}{RpF}$$

else

$$Event\_Index = |\Sigma|$$

In the case of trj_{12}, the second event is obtained by

$$\frac{Lvl\_Trns\_priv}{RpF} = \frac{3}{3} = 1$$

which the index for the event a.

This way we can continue along the trajectory and calculate the third deeper event as follows

$$RpF = \frac{9}{|3|^{(3-1)}} = 1$$

$$\frac{Lvl\_Trns\_priv}{RpF} = \frac{3}{1} = 3$$

which is the index for the event c.

Based on this algorithm, the indices for trj_{12} are 2-1-3 which correspond to the event sequence bac as shown in the highlighted column of Table 4.4.

	trj_1	trj_2	trj_3	trj_4	trj_5	trj_6	trj_7	trj_8	trj_9
depth=1	a	a	a	a	a	a	a	a	a
depth=2	aa	aa	aa	ab	ab	ab	ac	ac	ac
depth=3	aaa	aab	aac	aba	abb	abc	aca	acb	acc

Table 4.1 Trajectory group 1

	trj_{10}	trj_{11}	trj_{12}	trj_{13}	trj_{14}	trj_{15}	trj_{16}	trj_{17}	trj_{18}
depth=1	b	b	b	b	b	b	b	b	b
depth=2	ba	ba	ba	bb	bb	bb	bc	bc	bc
depth=3	baa	bab	bac	bba	bbb	bbc	bca	bcb	bcc

Table 4.2 Trajectory group 2

	trj_{19}	trj_{20}	trj_{21}	trj_{22}	trj_{23}	trj_{24}	trj_{25}	trj_{26}	trj_{27}

depth=1	c	c	c	c	c	c	c	c	c
depth=2	ca	ca	ca	cb	cb	cb	cc	cc	cc
depth=3	caa	cab	cac	cba	cbb	cbc	cca	ccb	ccc

Table 4.3 Trajectory group 3

	trj_{10}	trj_{11}	trj_{12}	trj_{13}	trj_{14}	trj_{15}	trj_{16}	trj_{17}	trj_{18}
depth=1	2	2	2	2	2	2	2	2	2
depth=2	21	21	21	22	22	22	23	23	23
depth=3	211	212	213	221	222	223	231	232	233

Table 4.4 Event indices for trajectory group 2

Algorithm 4.1

Input: $\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{q}_0)$, $G = (Q, \Sigma \cup \{\varepsilon\}, \delta, q_0)$,$N$, τ;

Output: *SD*, *D*, *SPD*, *PD*;

1: Construct $Tree(G, N) = (X, \Sigma, \xi, x_0)$:

$X_0 = \{x_0\} = \{UR(q_0)\}$;

for n=1, 2,…, N do begin

$X_n = \phi$;

for all $x \in X_{n-1}$ and $\sigma \in \Sigma$ do begin

if $(\exists q \in x)\delta(q, \sigma)!$, then

$\xi(x, \sigma) = UR(\{q \in Q: (\exists q' \in x)q \in (q', \sigma)\})$;

$X_n = X_n \cup \{\xi(x_0, \sigma)\}$;

end;

end;

$X = X_0 \cup X_1 \cup X_2 \cup \ldots X_N$;

2: Compute all fuzzy states:

$f(x_0) = \tilde{q}_0$;

for n=1, 2,…, N do begin

for all $x \in X_{n-1}$ and $\sigma \in \Sigma$ do begin

if $x' = \xi(x, \sigma)$!, then

$f(x') = f(x) \circ \tilde{\sigma}$;

end;

end;

3: Check all branches for detectability:

for $x \in X_N$ do

$b(x) = (x_0, \sigma_1, x_1, \ldots, x_{N-1}, \sigma_{N-1}, x_N = x)$,

$(\forall i = 1, \ldots, n-1) x_{i+1} = \xi(x_i, \sigma_i)$;

for $i = [(1 - 0.01D)N], \ldots, N$, do begin

if $x_i = [z_1, z_2, \ldots, z_n] \Lambda [\frac{\max\{z_1, z_2, \ldots, z_n\}}{\sum_{i=1}^{n}(z_n)} \geq \tau$

then $d(x_i) = True$,

else $d(x_i) = False$;

end;

if $(\forall i = [(1 - 0.01D)N], \ldots, N) d(x_i) = True$,

then $fuzzydet(x) = True$,

else $fuzzydet(x) = False$;

if $(\exists i = [(1 - 0.01D)N], \ldots, N) d(x_i) = True$

then $perifuzzydet(x) = True$,

else $perifuzzydet(x) = False$;

end;

4: Check *N*-detectabilities:

if $(\forall x \in X_N) fuzzydet(x) = True$

Then $SD = True$,

else $SD = False$;

if $(\exists x \in X_N) fuzzydet(x) = True$

Then $D = True$,

else $D = False$;

if $(\forall x \in X_N) perifuzzydet(x) = True$

Then $SPD = True$,

else $SPD = False$;

if $(\exists x \in X_N) perifuzzydet(x) = True$

Then $PD = True$,

else $PD = False$;

5: Stop.

Example 4.2

In this example, we consider the following fuzzy discrete event system with constraints $FDESwC = (\tilde{G}, G)$. All events that take place in the system are considered to be fully observable.

For the fuzzy automaton $\tilde{G}$, $\tilde{Q}$ = $[0,1]^n$ with $n = 3$, $\tilde{\Sigma} = \{\tilde{\alpha}, \tilde{\beta}, \tilde{\gamma}\}$ where

$$\tilde{\alpha} = \begin{bmatrix} 0.1 & 0.9 & 0.2 \\ 0 & 0.2 & 0.1 \\ 0.3 & 0.1 & 0.1 \end{bmatrix}$$

$$\tilde{\beta} = \begin{bmatrix} 0.1 & 0.1 & 0.8 \\ 0 & 0.7 & 0.1 \\ 0.9 & 0 & 0.1 \end{bmatrix}$$

$$\tilde{\gamma} = \begin{bmatrix} 0.1 & 0.1 & 0.8 \\ 0 & 0.7 & 0.1 \\ 0.9 & 0 & 0.1 \end{bmatrix}$$

$$\tilde{q}_0 = [0.9\ 0.1\ 0]$$

$$\tilde{\delta}(\tilde{q}, \tilde{\sigma}) = \tilde{q} \circ \tilde{\sigma}$$

where $\circ$ is the Max-Min operator.

The constraint automaton $G = (Q, \Sigma \cup \{\varepsilon\}, \delta, q_0)$ is shown in Figure 4.4, where $Q = \{1, 2, 3, 4\}$, $\Sigma = \{\alpha, \beta, \gamma\}$, $q_0 = [1\ 0\ 0\ 0]$, and δ is given below.

$$\alpha = \begin{bmatrix} 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 1 \\ 0\ 0\ 0\ 1 \\ 0\ 0\ 0\ 0 \end{bmatrix}$$

$$\beta = \begin{bmatrix} 0\ 0\ 1\ 0 \\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0 \\ 1\ 0\ 0\ 0 \end{bmatrix}$$

$$\gamma = \begin{bmatrix} 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0 \\ 0\ 0\ 1\ 0 \end{bmatrix}$$

In general, n and m may be different.

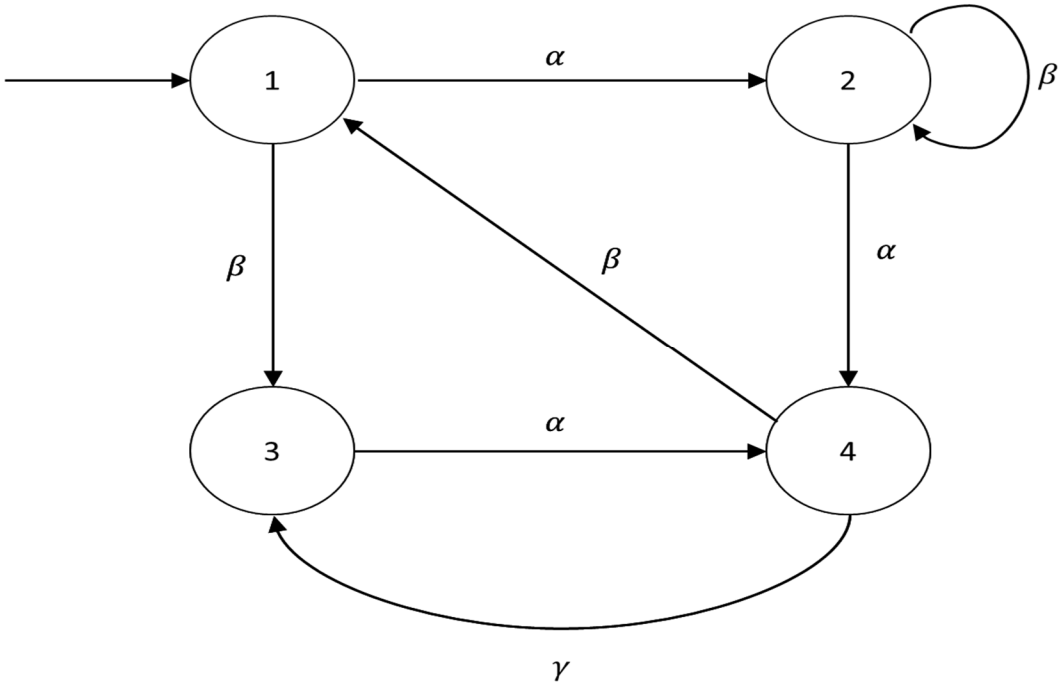

Figure 4.4 Constraint System G in Example 4.2

Figure 4.4 shows the constructed look-ahead tree for the constraint system G with $N = 3$.

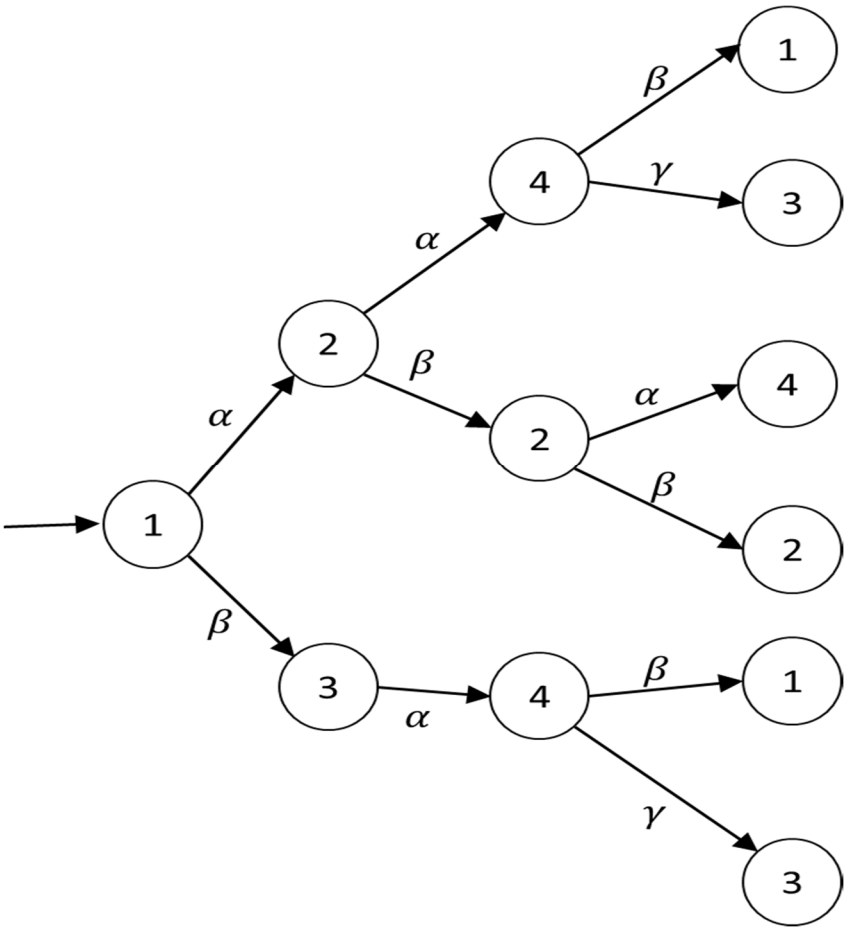

Figure 4.5 N-step look-ahead tree Tree (G, N) for Example 4.2

After calculating all the corresponding fuzzy nodes for the tree in Figure 4.5, the fuzzy tree $FTree(G,N)$ shown in Figure 4.6 is constructed.

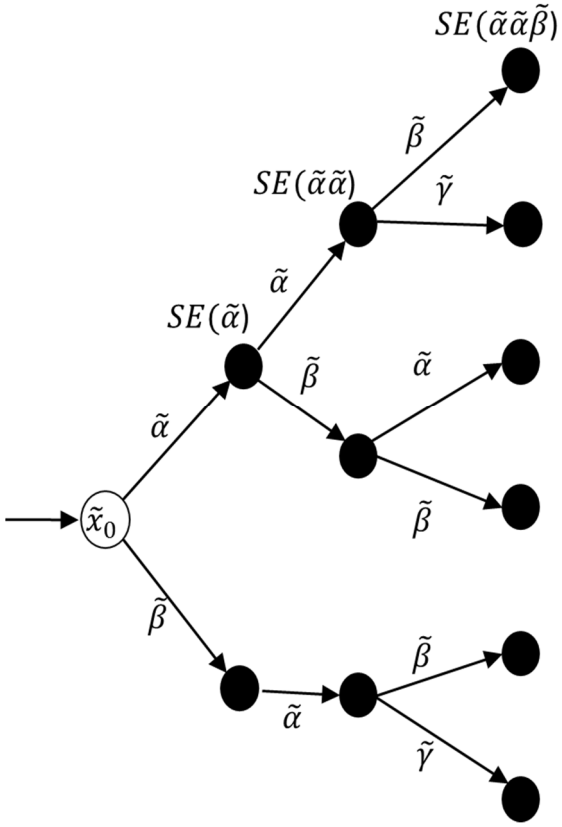

Figure 4.6 Fuzzy tree FTree (G, N) for Example 4.2

For the system in Example 4.2 with threshold $\tau = 0.6$, $D = 50\%$, and $N = 7$, the m-script in Figure 4.6 was executed and 51 trajectories were obtained as shown in Table 4.5.

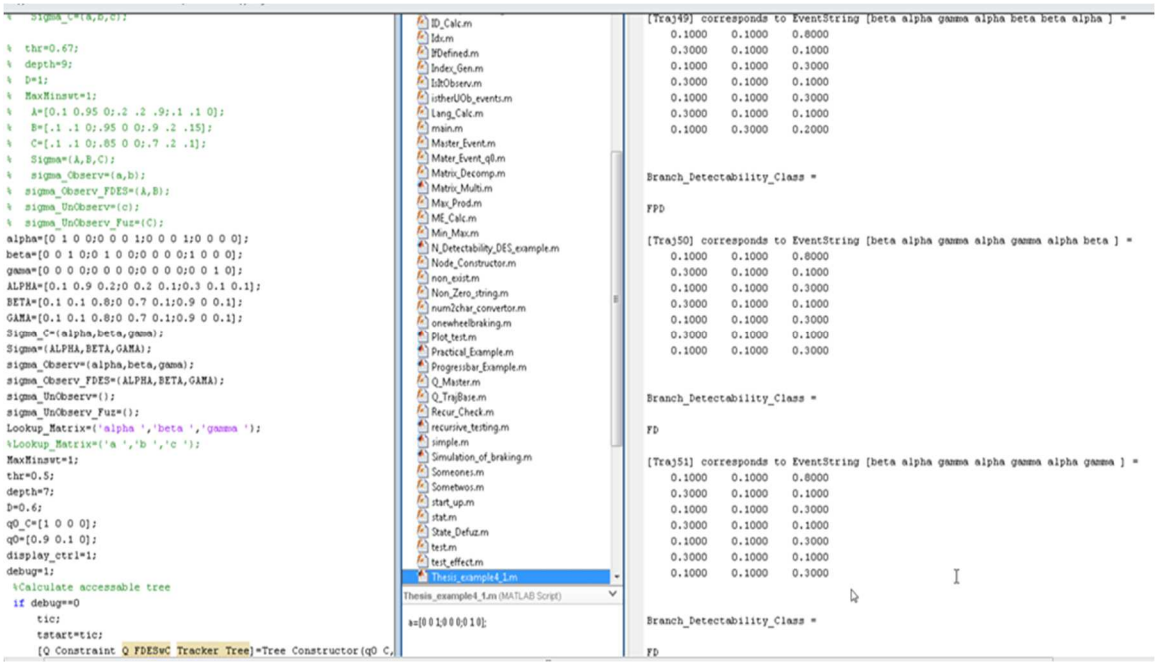

Figure 4.7 M-script for Example 4.2

From the results in Table 4.5, trajectory 51 is detectable, Trajectory 33 is periodically detectable, and trajectory 1 and 32 are not detectable. Since we have at least one trajectory is detectable and at least 1 trajectory is periodically detectable, the FDESwC is both weakly detectable and weakly periodically detectable. On the other hand, the system is not strongly detectable nor strongly periodically detectable because there is at least one trajectory i.e. Trajectory 1 that is not detectable.

Trajectory 1 $[\alpha\alpha\beta\alpha\alpha\beta\alpha]$	Trajectory 32 $[\alpha\beta\beta\beta\alpha\gamma\alpha]$	Trajectory 33 $[\alpha\beta\beta\beta\beta\alpha\beta]$	Trajectory 51 $[\beta\alpha\gamma\alpha\gamma\alpha\gamma]$
0.1 0.9 0.2	0.1 0.9 0.2	0.1 0.9 0.2	0.1 0.1 0.8
0.2 0.2 0.1	0.2 0.7 0.1	0.2 0.7 0.1	0.3 0.1 0.1
0.1 0.2 0.2	0.1 0.7 0.2	0.1 0.7 0.2	0.1 0.1 0.3

0.2 0.2 0.1	0.2 0.7 0.1	0.2 0.7 0.1	0.3 0.1 0.1
0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.7 0.2	0.1 0.1 0.3
0.2 0.2 0.1	0.2 0.2 0.1	0.2 0.2 0.1	0.3 0.1 0.1
0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.1 0.3
Branch is: ND	Branch is: ND	Branch is: PD	Branch is: D

Table 4.5 Sample trajectories for Example 4.2

In general, selection of D and τ should be application dependent. In Example 4.2, if we select deferent values for D and τ, we will get different branch detectability and consequently different overall system detectability. To illustrate the effect of tuning these 2 parameters, let's keep $D = 50\%$ and change τ to 0.4. The results show that all of the 51 trajectories constructed for the fuzzy *N*-step tree are detectable with respect to D and τ and consequently the system is strongly detectable with respect to $D = 50\%$ and $\tau = 0.40$. Table 4.6 shows only 4 trajectories as a sample of the m-script outcome.

Trajectory 1 $[\alpha\alpha\beta\alpha\alpha\beta\alpha]$	Trajectory 32 $[\alpha\beta\beta\beta\alpha\gamma\alpha]$	Trajectory 33 $[\alpha\beta\beta\beta\beta\alpha\beta]$	Trajectory 51 $[\beta\alpha\gamma\alpha\gamma\alpha\gamma]$
0.1 0.9 0.2	0.1 0.9 0.2	0.1 0.9 0.2	0.1 0.1 0.8
0.2 0.2 0.1	0.2 0.7 0.1	0.2 0.7 0.1	0.3 0.1 0.1
0.1 0.2 0.2	0.1 0.7 0.2	0.1 0.7 0.2	0.1 0.1 0.3
0.2 0.2 0.1	0.2 0.7 0.1	0.2 0.7 0.1	0.3 0.1 0.1
0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.7 0.2	0.1 0.1 0.3
0.2 0.2 0.1	0.2 0.2 0.1	0.2 0.2 0.1	0.3 0.1 0.1

0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.1 0.3
Branch is: D	Branch is: D	Branch is: D	Branch is: D

Table 4.6 Sample trajectories for Example 4.2 with D=50% and change τ=0.4

Let's change D to be 60% and select τ to be 0.5. The results in Table 4.7 are obtained. As can be notice in Table 4.7, trajectory 32 is now periodically detectable with respect to *D=60%* and $\tau = 0.5$ compares to the same trajectory in Table 4.5 .

Trajectory 1 $[\alpha\alpha\beta\alpha\alpha\beta\alpha]$	Trajectory 32 $[\alpha\beta\beta\beta\alpha\gamma\alpha]$	Trajectory 33 $[\alpha\beta\beta\beta\beta\alpha\beta]$	Trajectory 51 $[\beta\alpha\gamma\alpha\gamma\alpha\gamma]$
0.1 0.9 0.2	0.1 0.9 0.2	0.1 0.9 0.2	0.1 0.1 0.8
0.2 0.2 0.1	0.2 0.7 0.1	0.2 0.7 0.1	0.3 0.1 0.1
0.1 0.2 0.2	0.1 0.7 0.2	0.1 0.7 0.2	0.1 0.1 0.3
0.2 0.2 0.1	0.2 0.7 0.1	0.2 0.7 0.1	0.3 0.1 0.1
0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.7 0.2	0.1 0.1 0.3
0.2 0.2 0.1	0.2 0.2 0.1	0.2 0.2 0.1	0.3 0.1 0.1
0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.2 0.2	0.1 0.1 0.3
Branch is: ND	Branch is: PD	Branch is: PD	Branch is: D

Table 4.7 Sample trajectories for Example 4.2 with D=60% and change τ=0.6

As can be seen in Table 4.8, different values of D and τ results in different branch detectability outcome.

D	τ	D Branches %	PD Branches %	ND Branches %
40%	0.4	100.00%	0.00%	0.00%

40%	0.45	23.53%	9.80%	66.67%
50%	0.5	13.73%	23.53%	62.74%
50%	0.6	13.73%	11.76%	74.51%
60%	0.7	1.96%	5.88%	92.16%
60%	0.8	0.00%	0.00%	100%

Table 4.8 Branch detectability based on different D and τ and N=7

4.3 Detectabilities of partially observable fuzzy discrete event systems

In this section we introduce partially observable fuzzy discrete event systems with constraints ($POFDESwC$) and investigate their detectabilities.

Consider the language generated by the DES G as defined in Eq. (1.3). Since the constraint system G under study is partially observable, only observable sequences of events can be observed, which are defined in terms of the projection $P(L(G))$, where $P: \Sigma^* \rightarrow \Sigma_o^*$ is defined in Eq. (1.4). For example, if an event string $s = \alpha\beta\gamma\beta$ occurs, where both α and β are observable events and γ is an unobservable event, then only $P(s)$ can be observed, and $P(s) = P(\alpha\beta\gamma\beta) = \alpha\beta\beta$.

Unlike the conventional DES, FDES have infinite and continuous state space. Therefore *N*-detectabilities are more appropriate than detectabilities in investigating the detectabilities in FDES, and we will consider it here.

For the case of partial observation, we construct an *N*-step look-ahead tree under partial observation denoted by Eq. (4.1).

$$POTree(G, N) = (Y, \Sigma_o, \zeta, y_0) \tag{4.1}$$

where

- Y is the set of nodes
- Σ_o is the set of observable events
- ζ is the transition function
- y_0 is the root of the tree

The construction of the partial observation tree $POTree(G, N)$ in Eq. (4.1) requires extension of the constraint automata G from the initial states to all branches that have up to N observable events. This tree is denoted by $Tree(G, < N >) = (x, \Sigma, \xi, x_0)$ that is constructed in the same way as $Tree(G, N) = (x, \Sigma, \xi, x_0)$, except its branches have N observable events and hence may have a length exceeding N . We then replace all the unobservable events by empty string ε. Furthermore, all the nodes separated by the empty string (ε) in $Tree(G, < N >)$ are combined (merged) to construct corresponding nodes in the partial observation tree $POTree(G, N)$. Accordingly, y_0 is defined in terms of the unobservable reach of the initial states x_0 as in Eq. (4.2).

$$y_0 = UR(x_0) = \{x \in X: (\exists x' \in x_0) x \in \xi(x', \varepsilon)\} \tag{4.2}$$

If an event $\sigma \in \Sigma_o$ is observed in any state $x' \in y_0$, then the tree is extended by σ as expressed in Eq. (4.3).

$$\zeta(y_0, \sigma) = UR\{x \in X: (\exists x' \in y_0)(\exists s \in \Sigma^*)\, \sigma = P(s) \wedge x \in \xi(x', s)\} \tag{4.3}$$

Similar to fully observed FDES, the extension of *N*-detectability to $POFDESwC$ also requires construction of a fuzzy tree $FTree(G, < N >) = (X, \Sigma, \xi, x_0, f)$ where X, Σ, ξ, and x_0 are the same entities as in $Tree(G, < N >)$ defined for the constraint system. The additional element f maps each node $x \in X$ to the corresponding fuzzy state $f(x)$. $f(x)$ is represented by a row vector whose elements are in $[0,1]$. x_0 is given by

$$\tilde{x}_0 = f(x_0) = \tilde{q}_0$$

if $x' = \xi(x, \sigma)$, then

$$f(x') = f(x) \circ \tilde{\sigma}$$

where "$\circ$" is a fuzzy reasoning operator (e.g., Min-Max operator).

The fuzzy partial observation tree is then obtained by

$$FPOTree(G, N) = (Y, \Sigma_o, \zeta, y_0, h)$$

where Y, Σ_o , ζ, and y_0 are the same entities as in $POTree(G, N)$. The additional element h maps each node $y \in Y$ to the corresponding fuzzy state $h(y)$ as follows:

$$h(y) = \odot_{x \in y} f(x)$$

where $\odot$ is a fuzzy OR operator (e.g., Max() operator).

The fuzzy node $h(y)$ in the partial observation tree $FPOTree(G, N)$ is calculated by merging all the fuzzy states that can be reached unobservably after observing an event sequence at the initial states.

For example, if a node $\tilde{y} \in \tilde{Y}$ in $FPOTree(G, N)$ is constructed by merging nodes $\tilde{x}_1$, $\tilde{x}_2$, and $\tilde{x}_3$ in $FTree(G, < N >)$, then the corresponding fuzzy state of node y is calculated as follows:

$$h(y) = f(x_1) \cup f(x_2) \cup f(x_3)$$

For $f(y) = [z_1\ z_2 \ldots z_n]$, where $z_n \in [0,1]$, $f(y)$ is certain with respect to τ if the following condition is true:

$$\frac{max\{z_1, z_2, \ldots, z_n\}}{\sum_{i=1}^{n} z_i} \geq \tau$$

Accordingly, a branch $b \in Trj(G,N)$ is regarded as detectable with respect to D and τ if all of the last D (which is a percentage number) nodes are certain with respect to τ. Also, b is periodically detectable with respect to D and τ if some of the last D nodes are certain with respect to τ.

Consequently, *N*-detectabilities of $POFDESwC$ are defined as follows:

1. Strong *N*-Detectability: A $POFDESwC(G,\tilde{G})$ is strongly *N*-detectable with respect to D and τ if all the branches in $Trj(G,N)$ are detectable with respect to D and τ.
2. Weak *N*-Detectability: A $POFDESwC(G,\tilde{G})$ is weakly *N*-detectable with respect to D and τ if some branches in $Trj(G,N)$ are detectable with respect to D and τ.
3. Strong Periodic *N*-Detectability: A $POFDESwC(G,\tilde{G})$ is strongly periodically *N*-detectable with respect to D and τ if all the branches in $Trj(G,N)$ are periodically detectable with respect to D and τ.
4. Weak Periodic N-Detectability: A $POFDESwC(G,\tilde{G})$ is weakly periodically N-detectable with respect to D and τ if some branches in $Trj(G,N)$ are periodically detectable with respect to D and τ.

In addition to the features of Algorithm 4.1, algorithm 4.2 is capable of constructing the partial observation tree for both the constraints and fuzzy systems and then check the fuzzy system for detectability based on given parameters.

Algorithm 4.2

Input: $\tilde{G} = (\tilde{Q},\tilde{\Sigma},\tilde{\delta},\tilde{q}_0)$, $G = (Q,\Sigma \cup \{\varepsilon\},\delta,q_0)$,$N$, τ;

Output: *SD*, *D*, *SPD*, *PD*;

1: Construct $Tree(G, < N >) = (X, \Sigma, \xi, x_0)$:

$X_0 = \{x_0\} = \{UR(q_0)\}$;

for n=1, 2,…, N do begin

$X_n = \phi$ (the empty set);

for all $x \in X_{n-1}$ and $\sigma \in \Sigma$ do begin

if $(\exists q \in x)\delta(q, \sigma)!$, then

$\xi(x, \sigma) = UR(\{q \in Q: (\exists q' \in x)q \in \delta(q', \sigma)\})$;

$X_n = X_n \cup \{\xi(x, \sigma)\}$;

end;

end;

$X = X_0 \cup X_1 \cup X_2 \cup \ldots X_N$;

2: Construct $POTree(G, N) = (Y, \Sigma_o, \zeta, y_0)$:

$Y_0 = \{y_0\} = \{UR(x_0)\} = \{x \in X: (\exists x' \in x_0)x \in \xi(x', \varepsilon)\}$;

for n=1, 2,…, N do begin

$Y_n = \phi$ (the empty set);

for all $y \in Y_{n-1}$ and $\sigma \in \Sigma_o$ do begin

if $(\exists q \in y)\delta(q, \sigma)!$, then

$\zeta(y, \sigma) = UR(\{x \in X: (\exists x' \in y)(\exists s \in \Sigma^*)\, \sigma = P(s) \wedge x \in \xi(x', s)\})$;

$Y_n = Y_n \cup \{\zeta(y, \sigma)\}$;

end;

end;

$Y = Y_0 \cup Y_1 \cup Y_2 \cup \ldots Y_N$;

3: Construct $FPOTree(G, N) = (Y, \Sigma_o, \zeta, y_0, h(y))$:

$h(y_0) = \odot_{x \in y_0} f(x)$;

for n=1, 2,…, N do begin

for all $y \in Y_{n-1}$ and $\sigma \in \Sigma_o$ do begin

if $y' = \zeta(y, \sigma)$!, then

$\widetilde{y'} = h(y) \circ \tilde{\sigma} = (\odot_{x \in y} f(x)) \circ \tilde{\sigma}$;

end;

end;

4: Check all branches for detectability:

for $y \in Y_N$ do

$b(y) = (y_0, \sigma_1, y_1, \ldots, y_{N-1}, \sigma_{N-1}, y_N = y)$,

$(\forall i = 1, \ldots, n-1) y = \zeta(y_i, \sigma_i)$;

for $i = [(1 - 0.01D)N], \ldots, N$, do begin

if $y = [z_1, z_2, \ldots, z_n] \wedge [\frac{\max\{z_1, z_2, \ldots, z_n\}}{\sum_{i=1}^{n}(z_n)} \geq \tau$

then $d(y_i) = True$,

else $d(y_i) = False$;

end;

if $(\forall i = [(1 - 0.01D)N], \ldots, N) d(y_i) = True$,

then $fuzzydet(y) = True$,

else $fuzzydet(y) = False$;

if $(\exists i = [(1 - 0.01D)N], \dots, N) d(y_i) = True$

then $perifuzzydet(y) = True$,

else $perifuzzydet(y) = False$;

end;

5: Check fuzzy *N*-detectabilities:

if $(\forall y \in Y_N) fuzzydet(y) = True$

Then $SD = True$,

else $SD = False$;

if $(\exists y \in Y_N) fuzzydet(y) = True$

Then $D = True$,

else $D = False$;

if $(\forall y \in Y_N) perifuzzydet(y) = True$

Then $SPD = True$,

else $SPD = False$;

if $(\exists y \in Y_N) perifuzzydet(y) = True$

Then $PD = True$,

else $PD = False$;

6: Stop.

Example 4.3

This example illustrates the formulation and representation of a *POFDESwC*. For $\tilde{G}$, suppose that $\tilde{Q} = [0,1]^n$ with $n = 3$, $\tilde{\Sigma} = \tilde{\Sigma}_o \cup \tilde{\Sigma}_{uo}$ with $\tilde{\Sigma}_o = \{\tilde{\alpha}, \tilde{\beta}\}$ and $\tilde{\Sigma}_{uo} = \{\tilde{\gamma}\}$

where

$$\tilde{\alpha} = \begin{bmatrix} 0.1 & 0.9 & 0.2 \\ 0 & 0.8 & 0.1 \\ 0.3 & 0.75 & 0.1 \end{bmatrix}$$

$$\tilde{\beta} = \begin{bmatrix} 0.1 & 0.1 & 0.8 \\ 0.8 & 0.1 & 0.1 \\ 0.9 & 0 & 0.1 \end{bmatrix}$$

$$\tilde{\gamma} = \begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.2 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.6 \end{bmatrix}$$

$$\tilde{q}_0 = [0.9\ 0.1\ 0]$$

$$\tilde{\delta}(\tilde{q}, \tilde{\sigma}) = \tilde{q} \circ \tilde{\sigma}$$

where $\circ$ is the Max-Min operator. The system illustrated in Figure 4.8 represents the constraint system $G = (Q, \Sigma \cup \varepsilon, \delta, q_0)$, where $Q = \{1,2,3,4\}$, $\Sigma_o = \{\alpha, \beta\}$, $\Sigma_{uo} = \{\gamma\}$, $q_0 = \{1\} = [1\ 0\ 0\ 0]$, $D = 50\%$, and $\tau = 0.67$.

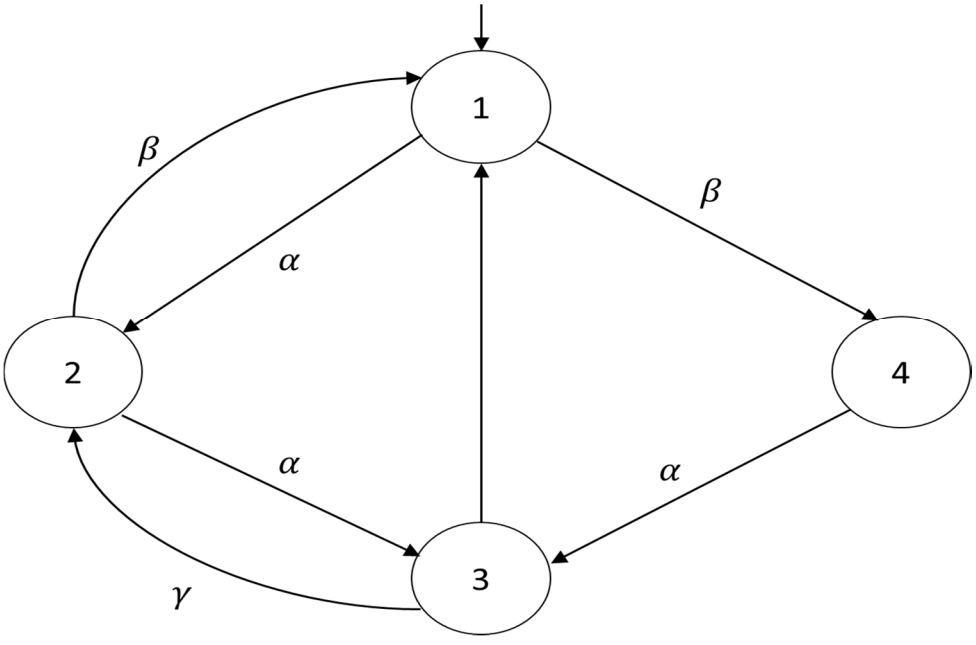

Figure 4.8 Constraint system for Example 4.3

This example illustrates how to construct $Tree(G, < N >)$ and $POTree(G, N)$ for the system in Figure 4.8 with N =3. Figure 4.9 shows $Tree(G, < N >)$. If a transition at any given node includes an unobservable event, the branch is further extended until *N* observable events are reached. In addition, if the last node has an unobservable event defined, that unobservable transition will be included in the tree.

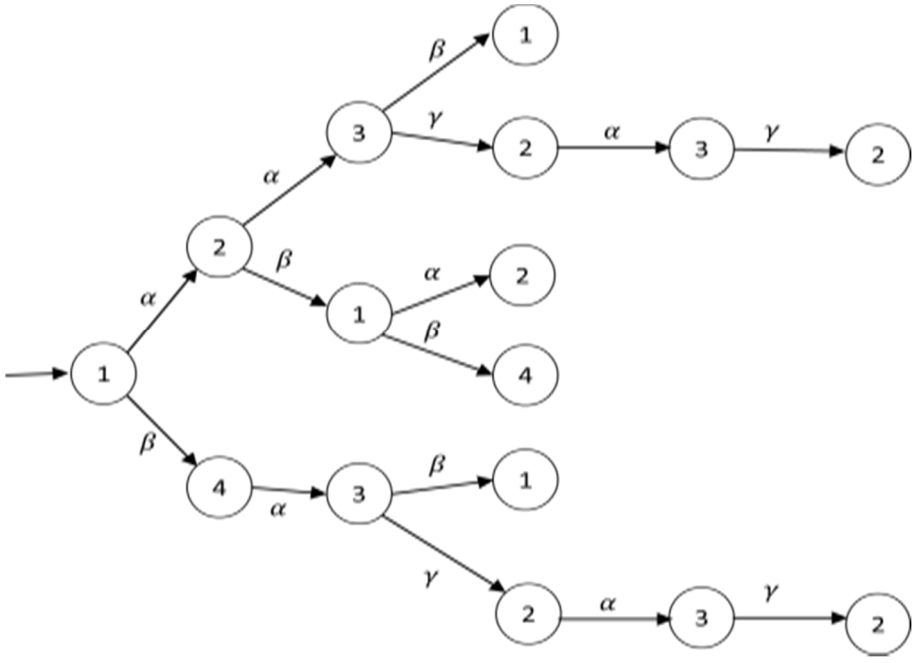

Figure 4.9 Tree (G,<N>) for Example 4.3

The partial observation tree $POTree(G, N)$ can be derived from $Tree(G, < N >)$ by combining all the nodes connected by unobservable events, as shown in Figure 4.9.

Each node y in the constructed partial observation tree corresponds to a state estimate of $t \in P(L(G))$:

$$y = SE(t) = \zeta(y_0, t)$$

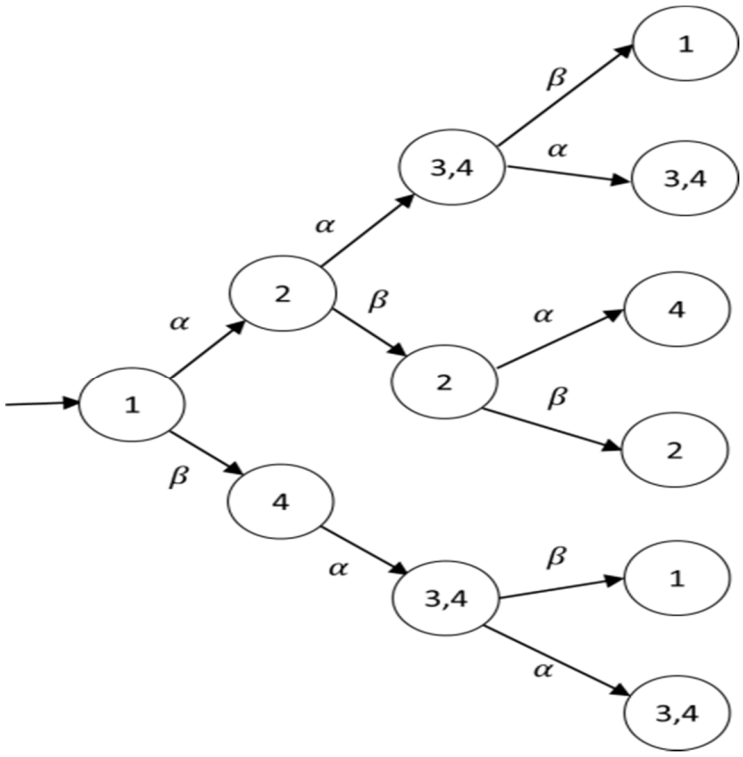

Figure 4.10 POTree (G, N) for Example 4.3

If the node y is singleton $(\text{i.e.}, |y| = 1)$, then y is certain and hence $SE(t)$ is certain. Each sequence of events with N observable events forms a branch in the partial observation tree. A branch is denoted by

$$Trj(G, N) = \{trj_1, trj_2, \ldots trj_k\}$$

where b_k is the *kth* branch in $POTree(G, N)$.

An *N*-step tree with $N = 9$ is considered, which represents the length of observable sequences of events in the system. Then all the nodes in all possible branches of the system and their corresponding fuzzy states are calculated and further checked for certainty to determine branch detectability. With the aid of a computer program, 230 branches were constructed. Only 6 branches were selected and shown in Table 4.9 and Table 4.10 . As an example, Trajectory 115 corresponds to the event sequence $\alpha\beta\alpha\beta\alpha\beta\alpha\beta\alpha$.

Trajectory 115 $[\alpha\beta\alpha\beta\alpha\beta\alpha\beta\alpha]$	Trajectory 116 $[\alpha\beta\alpha\beta\alpha\beta\alpha\beta\beta]$	Trajectory 117 $[\alpha\beta\alpha\beta\alpha\beta\beta\alpha\beta]$
0.10 0.90 0.20	0.10 0.90 0.20	0.10 0.90 0.20
0.80 0.10 0.10	0.80 0.10 0.10	0.80 0.10 0.10
0.10 0.80 0.20	0.10 0.80 0.20	0.10 0.80 0.20
0.80 0.10 0.10	0.80 0.10 0.10	0.80 0.10 0.10
0.10 0.80 0.20	0.10 0.80 0.20	0.10 0.80 0.20
0.80 0.10 0.10	0.80 0.10 0.10	0.80 0.10 0.10
0.10 0.80 0.20	0.10 0.80 0.20	0.10 0.10 0.80
0.80 0.10 0.10	0.80 0.10 0.10	0.30 0.75 0.20
0.10 0.80 0.20	0.10 0.10 0.80	0.75 0.10 0.30
Branch is: FD	Branch is: FD	Branch is: FPD

Table 4.9 Sample trajectories for Example 4.3

Trajectory 159 $[\beta\alpha\beta\alpha\alpha\alpha\alpha\beta]$	Trajectory 228 $[\beta\alpha\alpha\beta\beta\alpha\alpha\alpha]$	Trajectory 167 $[\beta\alpha\beta\alpha\beta\alpha\alpha]$
0.10 0.10 0.80	0.10 0.10 0.80	0.10 0.10 0.80
0.30 0.75 0.20	0.30 0.75 0.20	0.30 0.75 0.20
0.75 0.10 0.30	0.20 0.75 0.20	0.20 0.10 0.30
0.30 0.75 0.20	0.75 0.10 0.20	0.30 0.75 0.20
0.20 0.75 0.20	0.20 0.10 0.75	0.75 0.10 0.30
0.20 0.75 0.20	0.30 0.75 0.20	0.30 0.75 0.20

0.20 0.75 0.20	0.20 0.75 0.20	0.75 0.10 0.30
0.20 0.75 0.20	0.20 0.75 0.20	0.30 0.75 0.20
0.75 0.10 0.20	0.20 0.75 0.20	0.20 0.75 0.20
Branch is: FPD	Branch is: FPD	Branch is: FND

Table 4.10 Sample trajectories for Example 4.3

For each branch in the tables above, the ith row describes the fuzzy state after the observation of the ith event. The jth column represents the membership of the fuzzy state to the jth state of the FDES. The number of rows in each table corresponds to the depth of the branch *N*.

The highlighted rows shown in the above tables represent the last D nodes of each trajectory, and they are inspected for certainty. The branch detectability of each branch is shown at the bottom of the table.

Based on our criteria for detectability, the system in Example 4.1 is both *N*-detectable and periodically *N*-detectable because the results show at least one branch is detectable and that at least one branch is periodically detectable. On the other hand, since there is at least one branch that is not detectable or periodically detectable, $POFDESwC(G, \tilde{G})$ is not strongly detectable nor strongly periodically detectable.

The following theorem shows that *N*-detectabilities for crisp DES are special cases of *N*-detectabilities for $POFDESwC(G, \tilde{G})$ when the conditions for the DES to be a special case of FDES are satisfied.

Theorem 4.1

For a $POFDESwC(G, \tilde{G})$ and a DES G, if $m = n$, $\sigma = \tilde{\sigma}$ (in terms of the matrix), and $q_0 = \tilde{q}_0$ (in terms of the vector), then G is strongly *N*-detectable (*N*-detectable, strongly periodically *N*-detectable, and periodically *N*-detectable) with respect to D if and only if $POFDESwC(G, \tilde{G})$ is strongly *N*-detectable (*N*-detectable, strongly periodically *N*-detectable, and periodically *N*-detectable, respectively) with respect to D and $\tau > 0.5$.

Proof:

Consider any partially observation tree $POTree(G, N) = (Y, \Sigma, \zeta, y_0)$. If it can be proven that for all nodes $y \in Y$ in the partially observation tree, y is certain if and only if $h(y)$ is certain with respect to $\tau > 0.5$, then it can be concluded that *N*-detectability of the constraint system implies *N*-detectability of the FDES and vice versa.

Let $y = \zeta(y_0, t)$ and $h(y) = [z_1, z_2, \dots, z_n]$, to prove the theorem, it is necessary to show that $|y| = 1$ if and only if

$$\frac{max\{z_1, z_2, \dots, z_n\}}{\sum_{i=1}^{n} z_i} > 0.5$$

Since $m = n$, $\sigma = \tilde{\sigma}$ (in terms of the matrix), and $q_0 = \tilde{q}_0$ (in terms of the vector), $q_j \in y$ (where$1 \leq j \leq n$) if and only if $z_j = 1$ and hence the following is true:

$$|y| = 1 \Leftrightarrow \sum_{i=1}^{n} z_i = 1$$

$$\Leftrightarrow \frac{max\{z_1, z_2, \dots, z_n\}}{\sum_{i=1}^{n} z_i} > 0.5$$

CHAPTER 5 APPLICATION OF DETECTABILITY OF FUZZY DISCRETE EVENT SYSTEMS

In control field and many other engineering applications, the knowledge of the current state of the system is important to allow or disallow the occurrences of certain events in a system.

In previous sections we laid out the frame work of detectability criteria for fuzzy discrete event system with constraints. In this section, we show how to model a real-world application using the developed theorem. In addition, we demonstrate the value-added features of such modeling in terms of robustness and cost savings.

5.1 Vehicle dynamics control

One of the potential applications for the detectability of FDES is the control of vehicle dynamics in modern automobiles. Automobiles are widely used in our daily life and they are driven in different weather and road conditions as well as at different speeds, which leads to safety hazards that are often fatal. Most fatal car accidents occur due to loss of driver's control of a vehicle, which may lead to driving in unintended directions and possibly colliding with another vehicle, highway walls, or any other object on the road, or even rolling over pedestrians. To address these hazards, automobile manufactures came up with electronic controls for vehicle dynamics to assist drivers to avoid potential accidents and also improve comfort.

A real vehicle equipped with a brake control module that has a typical ESP (Electronic Stability Program) system has been used to take measurements in order to identify the suitability of FDES in vehicle dynamics control applications.

The calibration setup shown in Figure 5.1 is used for data acquisition from the brake ECU and vehicle busses such as CAN (Controller Area Network). In such setup, INCA (INtegrated Calibration and Acquisition Systems) tool is used as a measuring, calibration, and diagnostic system that provides comprehensive measuring support, supports in all essential tasks during control unit calibration, evaluates the measured data, and documents and stores measurement and calibration results. Quantities derived from measurements and calibration variables can be calculated and displayed online.

INCA tools are used for ECU development and test as well as for validation and calibration of electronically controlled systems in the vehicle, on the test bench, or in a virtual environment on the PC. They are comprised of hardware that connects to the vehicle's ECUs and a user interface software that runs in a PC to communicate to the hardware and the connected ECU.

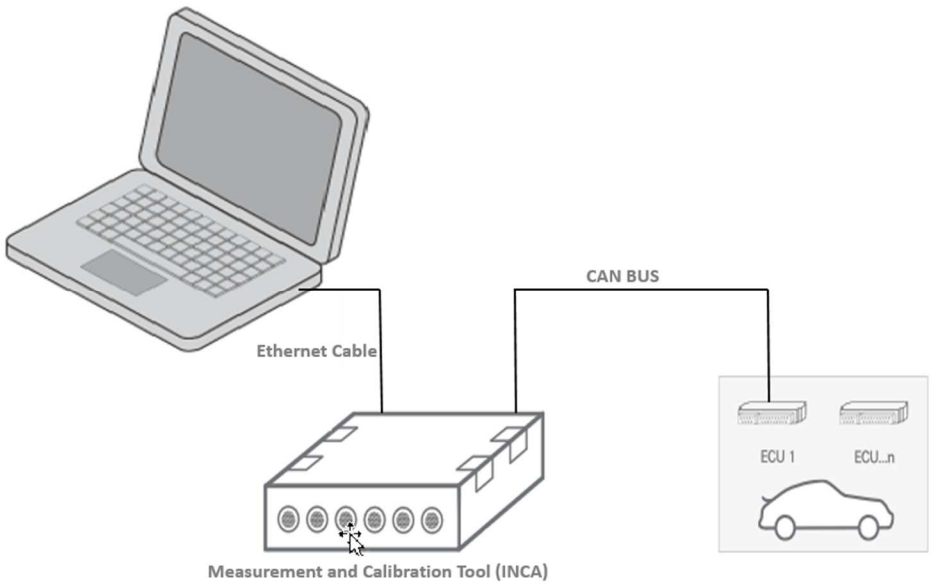

Figure 5.1 Vehicle's measurement and calibration setup

During track testing, which continued for about 5 minutes, the following sensors and actuation signals were captured and stored:

1. Wheel speed sensors
2. Steering angle sensor
3. Brake pressure at the wheel
4. Lateral acceleration sensor
5. Yaw rate sensor
6. ABS (Anti-lock Brake System) flag
7. TCS (Traction Control System) flag
8. VDC (Vehicle Dynamic Control) flag

Sensors' data and actuation signals were captured as the vehicle underwent various maneuvers and driving scenarios. These Maneuvers included the followings:

1. Double lane change maneuver
2. Applying high torque while driving on a surface that has a low friction coefficient
3. Applying full brake while driving at 60 mile/hour or more
4. Applying dynamic deceleration using APB (Automatic Park Brake) button only while driving at high speed (>60 mile/hour)
5. Applying a large steering angle on a road surface that has a low friction coefficient
6. Applying a small steering angle on a road surface that has a high friction coefficient

This experiment showed that the vehicle can be in ABS, TCS, and VDC states at the same time. This means the wheels are locked, the wheels are slipping, and the vehicle is understeering or oversteering at the same time, which is a typical fuzzy situation. Therefore, vehicle dynamic control is a good application for the detectability of FDES as the knowledge of vehicle state (wheels locked, wheels are slipping, or vehicle is understeering/oversteering) is crucial to decide the right reaction to stabilize the vehicle and improve the comfort level.

Figure 5.3 shows the automation of a system that represents vehicle dynamics in terms of stability control. The system is modeled with the following assumptions [17-18]:

1. The vehicle is front wheel drive.

2. Spring and dampers effects are ignored.
3. Longitudinal and lateral load transfers are ignored.
4. Rolling and pitching motion dynamics with yaw motions are ignored.

Six different discrete states are involved, including normal, which is the initial state, oversteering, understeering, rollover, spinning, and skidding. When the vehicle is in the normal state, it is assumed to be driven forward in the intended direction with full wheel traction on the road's surface. When the driver wants to change direction, he or she will turn the steering wheel towards the intended direction. However, the front wheel may slide in the lateral direction, putting the vehicle in the understeering state, or the rear wheels may slide laterally to move the vehicle into the oversteering state, depending on vehicle speed, road conditions, wheel conditions, and some other factors. If the linear wheel speed is faster or slower than the vehicle speed, then the vehicle is considered to be in the spinning or skidding state, respectively. All these states with the exception of the normal state are not desirable, and hence the controller's or driver's intervention is necessary to correct the situation.

The transition from one state to another takes place after observing some events. For example, when the vehicle is driven in a given direction and the steering wheel angel, measured by a steering angle sensor, is greater or smaller than the yaw angle measured by a yaw rate sensor, the vehicle transitions from the normal state to the understeering or oversteering state, respectively. When the vehicle is sliding laterally and with the lateral force being greater than the overturning resisting force, the vehicle moves to the rollover state. The normal state is the desired state for the vehicle to be in. However, if any of the

other states is detected, intervention of an automatic controller or driver is needed to rectify the situation.

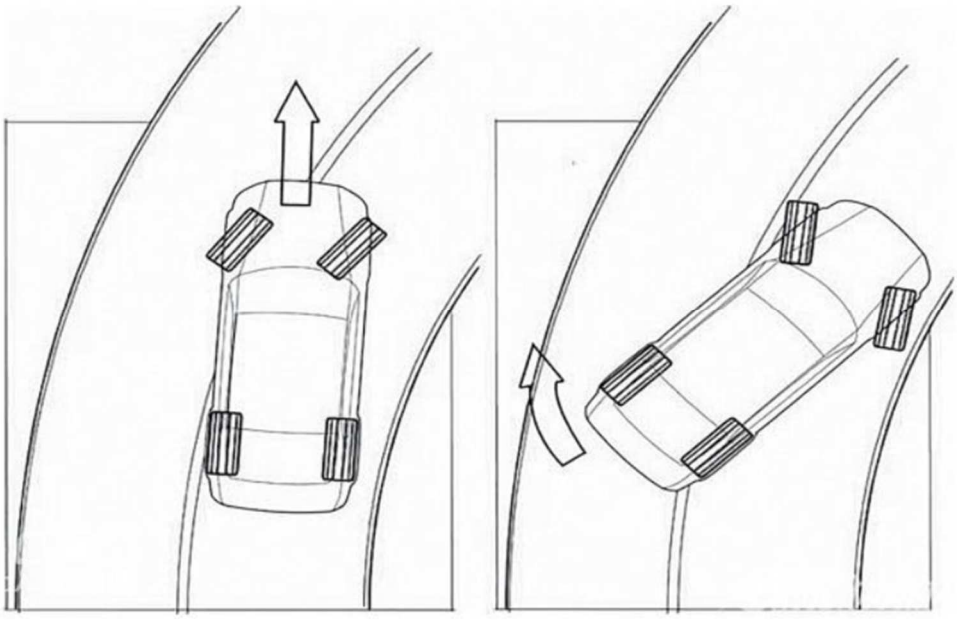

Figure 5.2 Left side shows understeered vehicle and right side shows oversteered vehicle [40]

The 6 states are modeled as 6 DES states with the same names (Figure 5.3). The following events are captured in the DES model that cause state transition:

$E1$: occurs when the driver requests a large steering angle δ on a road and friction coefficient μ is low.

$E2$: occurs when the driver requests a small steering angle δ on a road and μ is high.

$E3$: occurs when the vehicle control overreacts in mitigating oversteering or understeering situation.

ABS: Anti-lock braking system (ABS) is enabled.

TCS: Traction Control System (TCS) is enabled.

VDC: Vehicle Dynamic Control (VDC) is enabled.

$E4$: occurs when the driver requests a large torque.

$E5$: occurs when the driver requests a large braking pressure while driving forward.

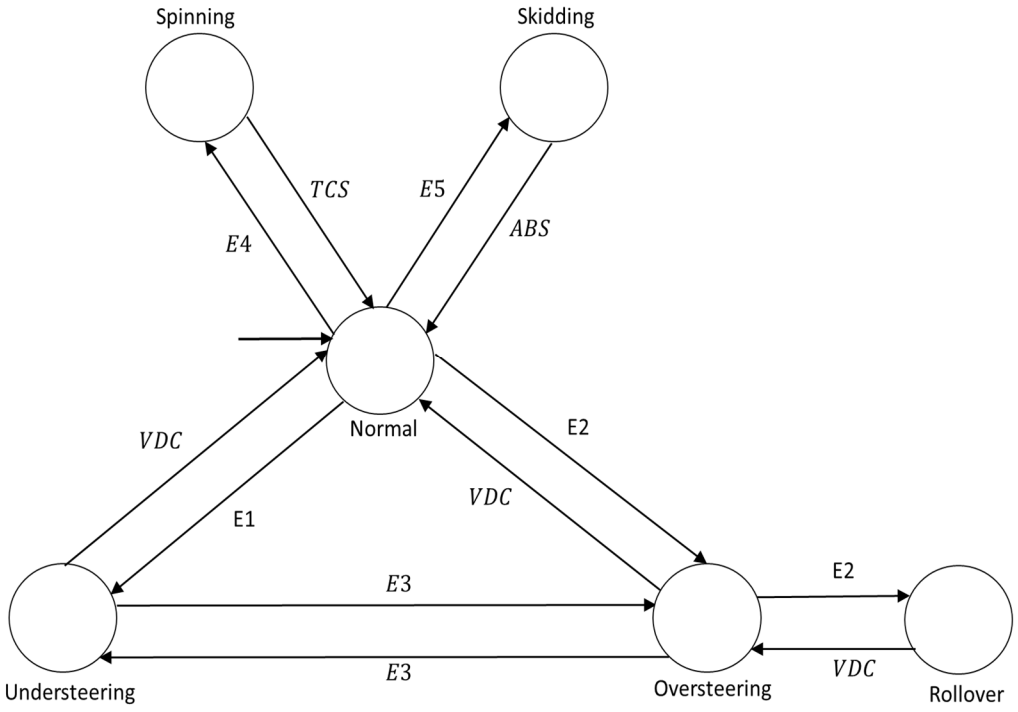

Figure 5.3 DES representing stability control for vehicle dynamics

When loss of steering is detected while cornering, i.e. oversteering or understeering state is detected, braking is applied to individual wheels, such as the outer front wheel to counter oversteering, or the inner rear wheel to counter understeering.

Typically, a vehicle on the road can be in multiple states at the same time. For example, when a vehicle is driven on a slippery road or on a curve, varying multiple states can be detected at the same time, including normal, spinning/skidding, or

oversteering/understeering states. Therefore, application of our new detectability theorems for the FDES model is particularly suitable for representing such state conditions to trigger corrective reactions.

Vehicle dynamic control relies heavily on the following high precision devices: steering wheel angle sensor, yaw rate sensor, acceleration sensor, and a set of wheel speed sensors. If any of these sensors is faulty, then the control will be degraded, and understeering and oversteering rectifications will be switched off. Therefore, detectability of $POFDESwC(G, \tilde{G})$ is suitable in such applications.

To model the vehicle dynamic control system using FDESwC, the automation shown in Figure 5.3 is considered as the constraint system. Let the states be enumerated as: normal=1, oversteering=2, understeering=3, rollover=4, spinning=5, and skidding=6. Then

$Q = \{1,2,3,4,5,6\}$, $\Sigma = \{E1, E2, E3, ABS, TCS, VDC, E4, E5\}$, $q_0 = \{1\} =$ [1 0 0 0 0 0], and δ: $\delta \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$

Accordingly, the crisp events can be modeled as:

$$E1 = \begin{bmatrix} 0&0&1&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix}, E2 = \begin{bmatrix} 0&1&0&0&0&0 \\ 0&0&0&1&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix},$$

$$E3 = \begin{bmatrix} 0&0&0&0&0&0 \\ 0&0&1&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix}, ABS = \begin{bmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 1&0&0&0&0&0 \end{bmatrix},$$

$$TCS = \begin{bmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix}, E4 = \begin{bmatrix} 0&0&0&0&1&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix},$$

$$VDC = \begin{bmatrix} 0&0&0&0&0&0 \\ 1&0&0&0&0&0 \\ 1&0&0&0&0&0 \\ 0&1&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix}, E5 = \begin{bmatrix} 0&0&0&0&0&1 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \end{bmatrix}.$$

It is not necessary that the number of states in the FDES match that in the constraint system. To demonstrate this, $\tilde{G}$ is defined as $\tilde{G} = (\tilde{Q}, \tilde{\Sigma}, \tilde{\delta}, \tilde{q}_0)$, where $\tilde{Q} = [0,1]^n$ with $n = 5$. Rollover state is not included in the FDES and hence fuzzy event matrix $[a_{ij}]$ defines only the transition from state i to state j, where both i and j are in the set of states {normal, oversteering, understeering, spinning, skidding}. Members of event set $\tilde{\Sigma}$ are chosen in collaborative effort with experts in the field of active safety system. Each element of a given fuzzy event matrix ($[a_{ij}]$) is determined based on the experts' best evaluation of the possibility for the vehicle's state to transition from the current state i to state j after the occurrence of that fuzzy event. Accordingly, the event matrices of the fuzzy system are filled in as follows:

$$\widetilde{E1} = \begin{bmatrix} 0.1&0.1&0.9&0.1&0.1 \\ 0.1&0.1&0.0&0.1&0.1 \\ 0.1&0.1&0.0&0.0&0.1 \\ 0.1&0.1&0.1&0.0&0.1 \\ 0.1&0.1&0.1&0.1&0.1 \end{bmatrix}, \widetilde{E2} = \begin{bmatrix} 0.1&0.9&0.2&0.1&0.1 \\ 0.1&0.4&0.3&0.1&0.1 \\ 0.1&0.3&0.2&0.1&0.1 \\ 0.1&0.1&0.1&0.1&0.1 \\ 0.1&0.1&0.1&0.1&0.1 \end{bmatrix},$$

$$\widetilde{E3} = \begin{bmatrix} 0.1\ 0.0\ 0.1\ 0.1\ 0.1 \\ 0.2\ 0.0\ 0.8\ 0.1\ 0.1 \\ 0.2\ 0.8\ 0.2\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \end{bmatrix}, \widetilde{ABS} = \begin{bmatrix} 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.9\ 0.1\ 0.1\ 0.1\ 0.1 \end{bmatrix},$$

$$\widetilde{TCS} = \begin{bmatrix} 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.9\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \end{bmatrix}, \widetilde{VDC} = \begin{bmatrix} 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.9\ 0.1\ 0.1\ 0.1\ 0.1 \\ 0.9\ 0.9\ 0.1\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.0\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \end{bmatrix},$$

$$\widetilde{E4} = \begin{bmatrix} 0.1\ 0.4\ 0.4\ 0.9\ 0.1 \\ 0.1\ 0.5\ 0.3\ 0.1\ 0.1 \\ 0.1\ 0.2\ 0.2\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.2\ 0.1\ 0.1 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \end{bmatrix}, \widetilde{E5} = \begin{bmatrix} 0.1\ 0.4\ 0.4\ 0.1\ 0.9 \\ 0.1\ 0.2\ 0.2\ 0.1\ 0.2 \\ 0.1\ 0.3\ 0.2\ 0.1\ 0.2 \\ 0.1\ 0.2\ 0.3\ 0.1\ 0.4 \\ 0.1\ 0.1\ 0.1\ 0.1\ 0.1 \end{bmatrix}.$$

The fuzzy initial state, threshold τ, and depth N are chosen as:

$\tilde{q}_0 = [0.9\ 0.2\ 0.1\ 0.0\ 0.0]$, $\tau = 0.67$, and $N = 8$.

Based on the given inputs, a fuzzy tree with 1701 trajectories is constructed. Each trajectory in the constructed tree corresponds to a specific maneuver or driving scenario. Table 5.1, Table 5.2, and Table 5.3 show some sample fuzzy trajectories with detectability classifications. Around 20% of the overall constructed trajectories have been analyzed in collaboration with experts in the field. This analysis showed that branch detectability classifications are somehow consistent with both design assumptions and experts' anticipations especially in driving scenarios that include longitudinal accelerations and decelerations.

Based on the evaluation of all trajectories in the fuzzy tree, the system is weakly periodically detectable because there are some trajectories classified as PD. Moreover, the system in not strongly detectable nor weakly detectable.

Trajectory 1365 [E4 TCS E4 TCS E4 TCS E1 VDC]	Trajectory 1366 [E4 TCS E4 TCS E4 TCS E2 E2]
0.10 0.40 0.40 0.90 0.10	0.10 0.40 0.40 0.90 0.10
0.90 0.10 0.10 0.10 0.10	0.90 0.10 0.10 0.10 0.10
0.10 0.40 0.40 0.90 0.10	0.10 0.40 0.40 0.90 0.10
0.90 0.10 0.10 0.10 0.10	0.90 0.10 0.10 0.10 0.10
0.10 0.40 0.40 0.90 0.10	0.10 0.40 0.40 0.90 0.10
0.90 0.10 0.10 0.10 0.10	0.90 0.10 0.10 0.10 0.10
0.10 0.10 0.90 0.10 0.10	0.10 0.90 0.20 0.10 0.10
0.90 0.90 0.10 0.10 0.10	0.10 0.40 0.30 0.10 0.10
Branch is PD	Branch is PD

Table 5.1 Detectabilities of vehicle dynamics example

Trajectory 1626 [E5 ABS E4 TCS E2 E2 VDC E2]	Trajectory 1660 [E5 ABS E5 ABS E1 E3 VDC E1]
0.10 0.40 0.40 0.10 0.90	0.10 0.40 0.40 0.10 0.90
0.90 0.10 0.10 0.10 0.10	0.90 0.10 0.10 0.10 0.10
0.10 0.40 0.40 0.90 0.10	0.10 0.40 0.40 0.10 0.90
0.90 0.10 0.10 0.10 0.10	0.90 0.10 0.10 0.10 0.10
0.10 0.90 0.20 0.10 0.10	0.10 0.10 0.90 0.10 0.10
0.10 0.40 0.30 0.10 0.10	0.20 0.80 0.20 0.10 0.10
0.40 0.30 0.10 0.10 0.10	0.80 0.20 0.10 0.10 0.10

0.10 0.40 0.30 0.10 0.10	0.10 0.10 0.80 0.10 0.10
Branch is ND	Branch is PD

Table 5.2 Detectabilities of vehicle dynamics example

Trajectory 1674 [E5 ABS E5 ABS E2 E3 E3 E2]	Trajectory 1701 [E5 ABS E5 ABS E5 ABS E5 ABS]
0.10 0.40 0.40 0.10 0.90	0.10 0.40 0.40 0.10 0.90
0.90 0.10 0.10 0.10 0.10	0.90 0.10 0.10 0.10 0.10
0.10 0.40 0.40 0.10 0.90	0.10 0.40 0.40 0.10 0.90
0.90 0.10 0.10 0.10 0.10	0.90 0.10 0.10 0.10 0.10
0.10 0.90 0.20 0.10 0.10	0.10 0.40 0.40 0.10 0.90
0.20 0.20 0.80 0.10 0.10	0.90 0.10 0.10 0.10 0.10
0.20 0.80 0.20 0.10 0.10	0.10 0.40 0.40 0.10 0.90
0.10 0.40 0.30 0.10 0.10	0.90 0.10 0.10 0.10 0.10
Branch is ND	Branch is PD

Table 5.3 Detectabilities of vehicle dynamics example

Table 5.3 shows that trajectory 1674 is not detectable (ND) due to the fact that all the highlighted nodes failed the certainty check and hence mapping to crisp states led to inconclusive states (corresponding crisp states are not singleton). In other words, detection of system's state is not possible. We observe that after the skidding correction, the vehicle has turned while braking (ABS event), which led to oversteering. This triggered oversteering overreaction, which, as a result, led to understeering. Further understeering overreaction followed by the occurrence of oversteering event (E2) should have led to

rollover state, which is not included in the fuzzy system. This explains why the last fuzzy node of this trajectory ([0.1 0.4 0.3 0.1 0.1]) is not certain.

By analyzing trajectory 1701 in Table 5.3, we observe that the vehicle was driven in longitudinal direction and a sudden brake has been applied, causing the vehicle to skid. This led to the triggering of ABS multiple times to stabilize the vehicle. This is consistent with design assumptions and hence the trajectory is classified as PD. As a result, the FDES is weakly periodically *N*-detectable, which means the system's state can be periodically detected in some trajectories. Therefore, triggering of control actions can be taken with some degree of confidence.

CHAPTER 6 CONCLUSION AND FUTURE WORK

6.1 Conclusion

This research focused on fuzzy detectabilities of fuzzy discrete event systems with full and partial observation. The main contributions highlighted by the study included:

1- Introduction of fuzzy discrete event system with constraints (FDESwC)

2- Reformulation of existing definitions of detectability for conventional DES to be defined in terms of the state estimate

3- Introduction of *N*-detectability for DES

4- Extension of *N*-detectability for DES to FDES with full observation and under partial observation.

5- Derive necessary and sufficient conditions for N-detectability of FDES with full observation and under partial observation.

6- Develop algorithms to check N-detectability of FDES with full observation and under partial observation.

7- Apply the results to vehicle dynamics control.

6.2 Future work

Future development for the application of fuzzy detectabilities of fuzzy discrete event systems to lung cancer treatment decision making is of greater beneficial potentials. Because patients' conditions and treatment outcomes are often fuzzy, they can be most effectively modeled as fuzzy discrete event systems. The theory developed in this research can then be used to develop algorithms and computer tools for optimal decision making for lung cancer treatments. Also, the control of vehicle dynamics is a potential application for

fuzzy *N*-detectability. Driving in an unintended direction due to road and weather conditions is one of the major causes of accidents and has claimed many lives. A research by the National Highway Traffic Safety Administration（NHTSA） in the United States shows that accidents due to loss of vehicle control are the second most dangerous form of accident in the Unites States, after head-on collisions. In the year 2000 alone, 9882 people lost their lives. 8,146 of those were killed in accidents involving only a single vehicle [39]. While the majority of these fatal accidents involved loss of control of a vehicle, it is clear that an active safety system capable of preventing accidents due to understeering, oversteering, and rollover will save lives, and as such is worthy of investigation.

In addition to cost savings, application of *N*-detectability in vehicle dynamics control may lead to reducing the number of high precision sensors needed in vehicles today to capture the vehicle's orientation and intended direction.

Furthermore, fuzzy systems can be combined with artificial inelegance and deep learning algorithms to develop self-learning systems that can enhance vehicle stability especially in self-driving automobiles.

Also, computations of state estimates for deep trajectories seemed to be a bottle-neck and hence any improvement in the algorithm used for such calculation will be of a great interest to optimize the use of resources and hence improve run time performance.

APPENDIX

Example 3.1 code:

```
%*************************************************************************
****
% Constraint System Parameters
%*************************************************************************
****
a=[0 1 0;0 1 0;0 1 0];
b=[0 0 1;1 0 0;0 0 0];
c=[0 0 0;0 0 1;0 0 0];
%Event matrix of the constraint system
Sigma_C={a,b,c};
%Observable event matrix of the constraint system
sigma_Observ={a,b};
%Observable event matrix of the constraint system
sigma_UnObserv={c};
%Initial state of the constraint system
q0_C=[1 0 0];
%*************************************************************************
****
% Fuzzy System Parameters
%*************************************************************************
****
A=[0.1 0.95 0;.2 .2 .9;.1 .1 0];
B=[.1 .1 0;.95 0 0;.9 .2 .15];
C=[.1 .1 0;.85 0 0;.7 .2 .1];
%Event matrix of the fuzzy system
Sigma={A,B,C};
%Observable event matrix of the fuzzy system
sigma_Observ_FDES={A,B};
%Observable event matrix of the fuzzy system
sigma_UnObserv_Fuz={C};
%Initial state of the fuzzy system
q0=[0.9 0.1 0];
Lookup_Matrix={'a ','b ','c '};
%Fuzzy resoning oprator, if set to 0 product operator will be used
MaxMinswt=1;
%Threshold
thr=0.67;
%Depth (N)
depth=7;
%The last D% nodes of a branch
D=0.5;
display_ctrl=1;
debug=1;
%*************************************************************************
****
% Calculate accessible tree for both constraint and fuzz system
%*************************************************************************
****
```

```
if debug==0
tic;
tstart=tic;
[Q_Constraint Q_FDESwC Tracker_Tree]=Tree_Constructor(q0_C,q0,...
Sigma_C,Sigma,sigma_Observ,sigma_Observ_FDES,...
sigma_UnObserv,sigma_UnObserv_Fuz,depth,MaxMinswt);
fprintf('%s t=%f\n','Trj_Constructor Calculations Done !',...
toc(tstart));
else
[Q_Constraint Q_FDESwC Tracker_Tree]=Tree_Constructor(q0_C,q0,...
Sigma_C,Sigma,sigma_Observ,sigma_Observ_FDES,...
sigma_UnObserv,sigma_UnObserv_Fuz,depth,MaxMinswt);
end
cert=cell(1,size(Q_FDESwC,1));
for i=1:size(Q_FDESwC,1)
%Extract Nodes for each Trajectory in the system
k=int2str(i);
B_temp=Tracker_Tree{i};
Event_String=num2char_convertor(B_temp,Lookup_Matrix);
A=cell2mat(Q_FDESwC{i});
B=cell2mat(Q_Constraint{i});
%strcat('Traj',k)=A
temp_para=strcat('Traj',k);
%Check for Detectability against defined critrion
rslt=certainity(A,thr,depth,D,i);
%display trajectories' strings and corresponding Nodes
%comment out if not needed
if display_ctrl==1
fprintf('[%s] corresponds to EventString [%s] = \n',temp_para,...
Event_String);
disp(B);
disp(A);
if rslt==1
Branch_Detectability_Class='FD'
elseif rslt==2
Branch_Detectability_Class='FPD'
elseif rslt==0
Branch_Detectability_Class='FND'
end
end
%Store Detectability classifications in array each element is the
%detectability clasification of a branch\trajectory
cert{1,i}=rslt;
end
Detectability_Matrix = cell2mat(cert);
%Detectability_Matrix=[0,0,0]
if Allones(Detectability_Matrix)==1
Detectability_Class='FSD';
elseif Someones(Detectability_Matrix)==1
Detectability_Class='FD';
elseif Alltwos(Detectability_Matrix)==1
Detectability_Class='FSPD';
elseif Sometwos(Detectability_Matrix)==1
Detectability_Class='FPD';
```

```
 else
 Detectability_Class='ND';
 end
```

Tree constructor code:

```
function [POT_Constraint POT_FDES Event_TrackerBaseTrj]=...
 Tree_Constructor(q0_C,q0_FUZ,sigma,sigma_FDES,sigma_Observ,...
 sigma_Observ_FDES,sigma_UnObserv,sigma_UnObserv_Fuz,...
 depth,MaxMinswt);
% Check_Unobservability(sigma_UnObserv,gama);
%Caluclate the number of events in Constained System, which is the same
in
%Fuzzy System
 No_Ev=size(sigma,2);
 %Calculate maximum number of Trajectories in the System
 No_Trj=power(No_Ev,depth);
%initialization of the cell array Event_Matrix
%Event_MatrixBaseTrj = cell(No_Trj,1);
Event_TrackerBaseTrj = cell(No_Trj,1);
POT_Constraint=cell(No_Trj,1);
POT_FDES=cell(No_Trj,1);
%Q_NodeTree=cell(No_Trj,1);
% for i=1:No_Ev
% sigma{i};
% end
%calculate Trajectory's nodes
No_TrjInGroup=No_Trj/No_Ev;
No_Trj_start=1;
No_Trj_end=No_Trj;
cnt=0;
dep=0;
%h = waitbar(0,'Initializing waitbar...');
for trj=No_Trj_start:No_Trj_end
 %perc=trj/No_Trj_end;
 %waitbar( perc/100,h,sprintf('%d%% along...',perc))
 %Event_Matrix_t=cell(depth,1);
 POT_Constraint_t=cell(depth,1);
 POT_FDES_t=cell(depth,1);
 %Q_Matrix_t=cell(depth,1);
 Event_Tracker=cell(depth,1);
 %strg=sigma{1}
 %Event_Matrix{};
 skip=0;
 %Initialize current state of constraint system and
 %fuzzy system [0 0 .. 0]
 q_current=zeros(1,size(q0_C,2));
 q_currentFUZ=zeros(1,size(q0_FUZ,2));
 %detemine the groupe the trajectory falls in, which determins the
first
 %string of event
 %Trj_Group = ceil(trj/No_TrjInGroup);
 %claculates the index of the trajectory within trajectory group
% if(mod(trj,No_TrjInGroup)==0)
```

```
% Trj_InternalIndex = No_TrjInGroup;
% else
% Trj_InternalIndex = mod(trj,No_TrjInGroup);
% end
 Event_Index=ceil(trj/No_TrjInGroup);
 first_strg=sigma{Event_Index};
 sigma_e=first_strg;
 first_strg_FDES=sigma_FDES{Event_Index};
 sigma_eFUZ=first_strg_FDES;
 temp=sigma_e;
 temp_FDES=sigma_eFUZ;
 %check if the string is defined from intial state
 %if (IfDefined(q0_C,temp,0))
 if any(q0_C*temp)
 %check if segma_e is observable or not
 if IsItObserv(sigma_e,sigma_Observ)
 %if due event is observable, then move to next state
 q_current=logical(q0_C*sigma_e);
 q_currentFUZ=FzOperation(q0_FUZ,sigma_eFUZ,MaxMinswt);
 %check for unobservable reach
 if istherUOb_events(q_current,sigma_UnObserv)
 [q_current_t q_currentFUZ_t]= UnObsevReach_Calc...
 (q_current,q_currentFUZ,sigma_UnObserv,...
 sigma_UnObserv_Fuz,MaxMinswt);
 q_current=q_current|q_current_t;
 q_currentFUZ=max(q_currentFUZ,q_currentFUZ_t);
 end
 %push calculated node in to the tree
 POT_Constraint_t{1,1}=double(q_current);
 POT_FDES_t{1,1}=double(q_currentFUZ);
 %track event causing trasnsition
 Event_Tracker{1,1}=Event_Index;
 else
 %calculate unobservable reach caused by current
 % unobservable event
 q_unobser=logical(q0_C*sigma_e);
 q_unobservFUZ=FzOperation(q0_FUZ,sigma_eFUZ,MaxMinswt);
 %combine current state with unobservable reach
 q_current=q_current | q_unobser;
 q_currentFUZ=max(q_currentFUZ,q_unobservFUZ);
 %check for further observable reach from calculated combined
 %states
 for j=1:size(sigma_Observ)
 %if IfDefined(q_current,sigma_Observ{j},0)
 if any(q_current*sigma_Observ{j})
 q_current=logical(q_current*sigma_Observ{j});
 q_currentFUZ=FzOperation(q_currentFUZ,...
 sigma_Observ_FDES{j},MaxMinswt);
 %check for unobservable reach
 if istherUOb_events(q_current,sigma_UnObserv)
 [q_current_t q_currentFUZ_t]= UnObsevReach_Calc...
 (q_current,q_currentFUZ,sigma_UnObserv,...
 sigma_FDES,MaxMinswt);
 q_current=q_current|q_current_t;
```

```
 q_currentFUZ=max(q_currentFUZ,q_currentFUZ_t);
 end
 POT_Constraint_t{1,1}=double(q_current);
 POT_FDES_t{1,1}=double(q_currentFUZ);
 Event_Tracker{1,1}=j;
 break;
 end
 end
 end
 rem_elem = rem(trj,No_TrjInGroup);
 Event_Index=ceil(trj/No_TrjInGroup);
% Event_Tracker{1,1}=Trj_Group;
% POT_Constraint_t{1,1}=q_current;%double(q_next);
% POT_FDES_t{1,1}=double(q_currentFUZ);
else continue ;
 end
 %for NE=1:No_Ev
 for dep=2:depth
 %repeat_factor "RpF" calculations
 RpF=No_TrjInGroup/power(No_Ev,dep-1);
 %calculation of event index within the event matrix
 if(ceil(rem_elem/RpF)==0)
 Event_Index =No_Ev;
 else
 Event_Index =ceil(rem_elem/RpF);
 end
 rem_elem = rem(trj,RpF);
% temp=temp*sigma{Event_Index};
% temp_FDES=temp_FDES*sigma_FDES{Event_Index};
 % Calculate next string of event in contraint system
 %temp=FzOperation(temp,sigma{Event_Index},0);
 %calculate current due event
 sigma_e=sigma{Event_Index};
 sigma_eFUZ=sigma_FDES{Event_Index};

 %calculate sequence of event
 temp=temp*sigma_e;
 % Check if the string of event is part of L(G)
 %if (IfDefined(q0_C,temp,0))
 if any(q0_C*temp)
 %check if segma_e is observable or not
 if IsItObserv(sigma_e,sigma_Observ)
 %if due event is observable, then move to next state
 q_current=logical(q_current*sigma_e);
 q_currentFUZ=FzOperation(q_currentFUZ,...
 sigma_eFUZ,MaxMinswt);
 %Event_Tracker{dep,1}=Event_Index;
 %check for unobservable reach
 if istherUOb_events(q_current,sigma_UnObserv)
 [q_current_t q_currentFUZ_t]= UnObsevReach_Calc...
 (q_current,q_currentFUZ,sigma_UnObserv,...
 sigma_FDES,MaxMinswt);
 q_current=q_current|q_current_t;
```

```
q_currentFUZ=max(q_currentFUZ,q_currentFUZ_t);
end
%push calculated node in to the tree
POT_Constraint_t{dep,1}=double(q_current);
POT_FDES_t{dep,1}=double(q_currentFUZ);
%track event causing trasnsition
Event_Tracker{dep,1}=Event_Index;
else
%calculate unobservable reach caused by current event
q_unobser=logical(q_current*sigma_e);
q_unobservFUZ=FzOperation(q_currentFUZ,...
sigma_eFUZ,MaxMinswt);
%cumbine current state with unobservable reach
q_current=q_current | q_unobser;
q_currentFUZ=max(q_currentFUZ,q_unobservFUZ);
for j=1:size(sigma_Observ)
%if IfDefined(q_current,sigma_Observ{j},0)
if any(q_current*sigma_Observ{j})
q_current=logical(q_current*sigma_Observ{j});
q_currentFUZ=FzOperation(q_currentFUZ,...
sigma_Observ_FDES{j},MaxMinswt);
%check for unobservable reach
if istherUOb_events(q_current,sigma_UnObserv)
[q_current_t q_currentFUZ_t]= ...
UnObsevReach_Calc(q_current,...
q_currentFUZ,sigma_UnObserv,...
sigma_FDES,MaxMinswt);
q_current=q_current|q_current_t;
q_currentFUZ=max(q_currentFUZ,...
q_currentFUZ_t);
end
POT_Constraint_t{dep,1}=double(q_current);
POT_FDES_t{dep,1}=double(q_currentFUZ);
Event_Tracker{dep,1}=j;
break;
end
end
%update stored node to reflect unobservable reach
% POT_Constraint_t{dep-1,1}=double(q_current);
% POT_FDES_t{dep-1,1}=double(q_currentFUZ);
if ((dep==depth) && ( size(POT_Constraint_t,1)...
< depth))
%BranchOutObservably(q_current,sigma_Observ)
end
end
else
dep=depth;
skip=1;
continue;
end
end

%end
```

```
 if skip==0
 cnt=cnt+1;
 %Event_MatrixBaseTrj{cnt,1}=Event_Matrix_t;
 POT_Constraint{cnt,1}=POT_Constraint_t;
 POT_FDES{cnt,1}=POT_FDES_t;
 %Q_NodeTree{cnt,1}=Q_Matrix_t;
 Event_TrackerBaseTrj{cnt,1}=Event_Tracker;
 %Q_trj=Event_Matrix;
 end
end
emptyCells = cellfun('isempty', POT_Constraint);
POT_Constraint(emptyCells)=[];
emptyCells = cellfun('isempty', POT_FDES);
POT_FDES(emptyCells)=[];
emptyCells = cellfun('isempty', Event_TrackerBaseTrj);
Event_TrackerBaseTrj(emptyCells)=[];
end
```

Observable events checker code:

```
function rtrn=IsItObserv(segma_e,sigma_Observ)
% this fucnctions checks if a given event is observable or not and
returns
% true or false accordingly
k=size(sigma_Observ,2);
e=segma_e;
rtrn=0;
for i=1:k
 if isequal(sigma_Observ{i},e)
 rtrn=rtrn|1;
 break;
 else
 rtrn=rtrn|0;
 end
end
end
```

Further unobservable events checker code

```
function rtrn = istherUOb_events(q_new,Set_UnObservale_Events)
k= size(Set_UnObservale_Events,2);
rtrn=0;
for i=1:k
 if any(q_new*Set_UnObservale_Events{i})
 rtrn =1;
 break;
 else
 rtrn=0;
 end
end
end
```

Max-Min Operation code:

```
function C = Min_Max(q,e)
%q=[3 6 8 10];e=[1 2 3 1;13 41 12 5; 8 6 7 4; 30 2 1 8];
[raws,cols]=size(q);
N_States=cols;
[m,n]=size(e);
```

```
C=zeros(raws,N_States);
%C=q;
for k=1:raws
 for i=1:N_States
 for j=1:m
 temp=min(q(k,j),e(j,i));
 C(k,i)=max(C(k,i),temp);
 %(k,i)=C(k,i)+q(k,j)*e(j,i);
 end
 end
end
end
```

Event observability checker code:

```
function rtrn=IsItObserv(segma_e,sigma_Observ)
% this fucnctions checks if a given event is observable or not and
returns
% true or false accordingly
k=size(sigma_Observ,2);
e=segma_e;
rtrn=0;
for i=1:k
 if isequal(sigma_Observ{i},e)
 rtrn=rtrn|1;
 break;
 else
 rtrn=rtrn|0;
 end
end
end
```

Unobservable reach calculator code:

```
function [q_new q_newFUZ] =
UnObsevReach_Calc(q,q_FUZ,sigma_UnObserv,sigma_UnObserv_Fuz,MaxMinswt)
q;
k=size(sigma_UnObserv,2);
for i=1:k
 e=sigma_UnObserv{i};
 e_FUZ=sigma_UnObserv_Fuz{i};
 if any(q*e)
 [q_new q_newFUZ]=Recur_Check(q,q_FUZ,e,e_FUZ,MaxMinswt);
 end
end
end
```

Recursive calculator code:

```
function [q_new q_newFUZ]=Recur_Check(q,q_FUZ,e,e_FUZ,MaxMinswt)
q_new= logical(q*e);
q_newFUZ= FzOperation(q_FUZ,e_FUZ,MaxMinswt);
if any(q_new*e)
 [q_new_t q_newFUZ_t]=Recur_Check(q_new,q_newFUZ,e,e_FUZ,MaxMinswt);
 q_new=q_new|q_new_t;
 q_newFUZ=max(q_newFUZ,q_newFUZ_t);
end
end
```

Kleene closure calculator code:

```
function [C ]=Master_Event(q0,Sigma,depth,Min_Max_swt)
 a=[0.7 0.3 0;0.4 0.4 0.2;0.2 0.3 0.8];b=[0.2 0.4 0.7;0.4 0.6 0;0.4 0.3
0.3];c=[0.6 0.2 0.3;0.4 0.4 0.3;0 0.7 0.4];
 Sigma ={a,b,c} ; depth=2;
 Min_Max_swt=0;
q0=[1 0 0];
temp=depth-1;
Cell_Size=0;
%Calculation of number of events by adding number of events of depth 1
to
%number of events of depth 2 +......
for z=1 : depth
 Cell_Size=Cell_Size + power(size(Sigma,2),z);
end
%Intializations of cell
C=cell(1,Cell_Size);
C=Sigma;
temp_cell=Sigma;
offset=size(Sigma,2);
ct=0;
while(temp)
 [temp_cell,ct]=ME_Calc(Sigma,temp_cell,Min_Max_swt)
% for i= (offset+1): (offset+ct)
% C{i}=temp_cell{(i-offset)};
%
% end
C={C{:},temp_cell{:}}
 offset=offset+ct;
 temp=temp-1;
end
end
```

Certainty checker code:

```
function cert=certainity(Q,thr,Depth,D,cnt)
% Q=[0.1 0.9;0 0.2];
% thr=0.1;
% Depth=2;
%[x,y,z]=size(Q);
%cert=char(zeros(x,1,z));
%Q=cell2mat(Q)
Index=ceil(Depth*(1-D)+1);
if Index > Depth
 Index=Depth;
elseif Index < 1
 Index =1;
end
cert_temp=[];
for i=Index:Depth;
 temp = max(Q(i,:))/sum(Q(i,:));
 if temp >=thr
 cert1=1;
 else
 cert1=0;
 end
 cert_temp(i-Index+1,1)=cert1;
```

```
end
%cert_temp
cert=cert_temp;
if all(cert)
 detect_est=1;
elseif any(cert)
 detect_est =2;
else
 detect_est=0;
end
cert=detect_est;
%cell2mat(cert);
% for i=1:z
% cnt=1;
% for j=Index:x
% %for k=1:y
% temp = max(Q(j,:,i)/sum(Q(j,:,i)));
% if temp >= thr
% cert(j,1,i)='T';
% else
% cert(j,1,i)='F';
% end
% cnt=cnt+1;
% %end
% end
% end
% %Q
% A=size(cert)
end
```

Event retriever code:

```
function[rslt_t]= num2char_convertor(temp,Lookup_Matrix)
%temp=6;
temp;
emptyCells = cellfun('isempty', temp);
temp(emptyCells)=[];
size(temp);
for i=1:size(temp)
 rslt_n=temp{i,1};
 %rslt_shift=char(rslt_n+96);
 rslt_shift=Lookup_Matrix{rslt_n};
 rslt=rslt_shift;
 if i==1
 B=rslt;
 else
 B=horzcat(B,rslt);
 end
 rslt_t=B;
end
end
```

BIBLIOGRAPHY

[1] A. O. Mekki, F. Lin, H. Ying, and M. J. Simoff, "Fuzzy detectabilities for fuzzy discrete event systems," in *proc. 2017 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, 2017.

[2] A. Van Zanten, and Robert Bosch Gmbh, "Evolution of Electronic Control Systems for Improving the Vehicle Dynamic Behavior," in *Proc. of the Int. Symp. on Advanced Vehicle Control*, 2002.

[3] C.G. Cassandras and S. Lafortune, "Introduction to Discrete Event Systems", Kluwer, 1999.

[4] C. M. Ozveren and A. S. Willsky, "Observability of discrete event dynamic systems," *IEEE Trans. on Automatic Control*, vol. 35, no. 7, pp. 797–806, Jul. 1990.

[5] Daowen Qiu and Fuchun Liu, "Fuzzy Discrete-Event Systems Under Fuzzy Observability and a Test Algorithm," *IEEE Tran. on Fuzzy Systems*, vol. 17, no. 3, pp. 578–589, Jun. 2009.

[6] D. Qiu, "Supervisory Control of Fuzzy Discrete Event Systems: A Formal Approach," *IEEE Trans. on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 1, pp. 72–88, Feb. 2005.

[7] F. Lin and H. Ying, "modeling and control of fuzzy discrete event system," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 32, no. 4, pp. 408-415, Aug. 2002.

[8] Feng Lin and Hao Ying, "State-Feedback Control of Fuzzy Discrete-Event Systems," *IEEE Tran. on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 3, pp. 951–956, Jun. 2010.

[9] F. Lin and W. M. Wonham, "Decentralized supervisory control of discrete-event systems," *Information Sciences*, vol. 44, no. 3, pp. 199–224, Apr. 1988.

[10] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information Sciences*, vol. 44, no. 3, pp. 173–198, Apr. 1988.

[11] F. Lin, "Diagnosability of discrete event systems and its applications," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 4, no. 2, pp. 197–212, May 1994.

[12] J. G. Thistle and W. M. Wonham, "Control of Infinite Behavior of Finite Automata," *SIAM Journal on Control and Optimization*, vol. 32, no. 4, pp. 1075–1097, Jul. 1994.

[13] M. Heymann and F. Lin, "Discrete-event control of nondeterministic systems," *IEEE Trans. on Automatic Control*, vol. 43, no. 1, pp. 3–17, 1998.

[14] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Trans. on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[15] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzis, "Failure diagnosis using discrete-event models," *IEEE Trans. on Control Systems Technology*, vol. 4, no. 2, pp. 105–124, Mar. 1996.

[16] M. Sampath, S. Lafortune, and D. Teneketzis, "Active diagnosis of discrete-event systems," *IEEE Tran. on Automatic Control*, vol. 43, no. 7, pp. 908–929, Jul. 1998.

[17] M. Acosta, S. Kanarachos, and M. Blundell, "Vehicle agile maneuvering: From rally drivers to a finite state machine approach," in *2016 IEEE Symp. Series on Computational Intelligence (SSCI)*, 2016, pp. 1-8.

[18] N. R. Dixit, "Evaluation of Vehicle Understeer Gradient Definitions," M. S. thesis, Dept. Mech. Eng. , Ohio State Univ., Columbus, OH, 2009.

[19] P.E. Caines, R. Greiner, S. Wang, "Dynamical logic observers for finite automata," in *Proc. of the 27th IEEE CDC*, 1988, 226-233.

[20] P. J. Ramadge and W. M. Wonham, "Supervisory Control of a Class of Discrete Event Processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, Jan. 1987.

[21] P.J. Ramadge and W.M. Wonham, "The control of discrete event systems," *Proc. of the IEEE*, vol. 77, no. 1, pp. 81-98, 1989.

[22] P. Ramadge, "Observability of discrete event systems," in *1986 25th IEEE CDC*, 1986, pp. 1108-1112.

[23] R. Cieslak, C. Desclaux, A. S. Fawaz, and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Trans. on Automatic Control*, vol. 33, no. 3, pp. 249–260, Mar. 1988.

[24] S. Shu and F. Lin, "Detectability of discrete event systems with dynamic event observation," *Systems & Control Letters*, vol. 59, no. 1, pp. 9–17, Jan. 2010.

[25] S. Shu and F. Lin, "Generalized detectability for discrete event systems," *Systems & Control Letters*, vol. 60, no. 5, pp. 310–317, May 2011.

[26] S. Shu and F. Lin, "Delayed Detectability of Discrete Event Systems," *IEEE Trans. on Automatic Control*, vol. 58, no. 4, pp. 862–875, Apr. 2013.

[27] S. Shu and F. Lin, "Enforcing Detectability in Controlled Discrete Event Systems," *IEEE Trans. on Automatic Control*, vol. 58, no. 8, pp. 2125–2130, Aug. 2013.

[28] S. Shu and Feng Lin, "I-Detectability of Discrete-Event Systems," *IEEE Trans. on Automation Science and Engineering*, vol. 10, no. 1, pp. 187–196, Jan. 2013.

[29] S. Shu, F. Lin, and H. Ying, "Detectability of Discrete Event Systems," *IEEE Trans. on Automatic Control*, vol. 52, no. 12, pp. 2356–2359, Dec. 2007.

[30] S. Shu, F. Lin, H. Ying, and X. Chen, "State estimation and detectability of probabilistic discrete event systems," *Automatica*, vol. 44, no. 12, pp. 3054–3060, Dec. 2008.

[31] S. Shu, Z. Huang, and F. Lin, "On-line Detection and Sensor Activation for Discrete Event Systems1," *IFAC Proceedings Volumes*, vol. 43, no. 12, pp. 187–192, 2010.

[32] S.H. Zad, R.H. Kwong, and W.M. Wonham, "Fault diagnosis in discrete-event systems: Framework and model reduction," *IEEE Trans. on Automatic Control*, vol. 48, no. 7, pp. 1199-1212, Jul. 2003.

[33] Tae-Sic Yoo and S. Lafortune, "Polynomial-time verification of diagnosability of partially observed discrete-event systems," *IEEE Trans. on Automatic Control*, vol. 47, no. 9, pp. 1491–1495, Sep. 2002.

[34] W. Wang, S. Lafortune, and F. Lin, "An algorithm for calculating indistinguishable states and clusters in finite-state automata with partially observable transitions," *Systems & Control Letters*, vol. 56, no. 9–10, pp. 656–661, Sep. 2007.

[35] Xinyu Du, Hao Ying, and Feng Lin, "Theory of Extended Fuzzy Discrete-Event Systems for Handling Ranges of Knowledge Uncertainties and Subjectivity," *IEEE Trans. on Fuzzy Systems*, vol. 17, no. 2, pp. 316–328, Apr. 2009.

[36] Yongzhi Cao and Mingsheng Ying, "Observability and decentralized control of fuzzy discrete-event systems," *IEEE Trans. on Fuzzy Systems*, vol. 14, no. 2, pp. 202–216, Apr. 2006.

[37] Jang, J., Sun, C. and Mizutani, E., "Neuro-fuzzy and soft computing," Upper Saddle River, NJ: Prentice Hall, 1997, pp.11-70.

[38] L. Zadeh, "Fuzzy sets," *Inform. and Control*, vol. 8, pp. 338-353, 1965.

[39] B. Schofield, *Vehicle dynamics control for rollover prevention*. Lund: Department of Automatic Control, Lund University, 2006.

[40] Firdaus, "Understanding oversteer and understeer and nailing it right! | Maintenance," CarDekho, 02-Feb-2015. [Online]. Available: https://www.cardekho.com/advisory-stories/maintenance/understanding-oversteer-and-understeer-and-nailing-it-right.htm. [Accessed: 19-Aug-2018].

ABSTRACT

DETECTABILITY OF FUZZY DISCRETE EVENT SYSTEMS

by

AHMED MEKKI

December 2018

Advisor: Dr. Hao Ying

Advisor: Dr. Feng Lin

Major: Electrical Engineering

Degree: Doctor of Philosophy

Dynamic systems that can be modeled in terms of discrete states and a synchronous events are known as discrete event systems (DES). A DES is defined in terms of states, events, transition dynamics, and initial state. Knowing the system's state is crucial in many applications for certain actions (events) to be taken. A DES system is considered a fuzzy discrete event system (FDES) if its states and events are vague in nature; for such systems, the system can be in more than one state at the same time with different degrees of possibility (membership). In this research we introduce a fuzzy discrete event system with constraints (FDESwC) and investigate its detectabilities. This research aims to address the gap in previous studies and extend existing definitions of detectability of DES to include the detectability in systems with substantial vagueness such as FDES. These definitions are first reformulated to introduce *N*-detectability for DES, which are further extended to define four main types of detectabilities for FDES: strong *N*-detectability, (weak) *N*-detectability, strong periodic *N*-detectability, and (weak) periodic *N*-detectability. We first

partition the FDES into trajectories of a length dictated by the depth of the event's string (length of the event sequence); each trajectory consists of a number of nodes, which are further investigated for detectability by examining them against the newly introduced certainty criterion. Matrix computation algorithms and fuzzy logic operations are adopted to calculate the state estimates based on the current state and the occurring events. Vehicle dynamics control example is used to demonstrate the practical aspect of developed theorems in real-world applications.

AUTOBIGRAPHICAL STATEMENT

Ahmed Mekki was born in Elobeid, Sudan. He received a B.Sc. (Hons.) degree in Electrical Engineering from University of Khartoum, Khartoum, Sudan in December 1996. He also finished His M.S.E.E degree in Electrical Engineering from University of Detroit Mercy, Detroit, MI, USA, in May 2002. He was awarded Graduate Professional Scholarship (GPS) from Wayne state University for the academic years: 2009-2010, 2011-2012, and 2013-2014.

He is currently working in the automotive industry for the world's biggest automotive supplier (Robert Bosch LLC) as a SENIOR SOFTWARE ARCHITECT in Detroit, MI, USA.

Ahmed Mekki is an IEEE member. He published Fuzzy logic for determination of crack severity in defense applications paper in 2010 SPIE defense conference held in Orlando FL. He also published Fuzzy detectabilities for fuzzy discrete event systems paper in 2017 IEEE International Conference held in Napoli, Italy.