Property-Based Fault Detection and Diagnosis for Safety-Critical Cyber-Physical Systems

BY

YAO FENG
BSc, Northeastern University, 2008

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Chicago, 2018

Chicago, Illinois

Defense Committee:

Miloš Žefran, Chair and Advisor
A. Prasad Sistla, Computer Science
Jie Yang, Statistics
Piotr Gmytrasiewicz, Computer Science
Xinhua Zhang, Computer Science

Contribution of Authors

Chapter 1 is an introduction of my research motivation and a literature review that places my thesis questions in the context of the larger field and highlights the significance of my work in this thesis. Chapter 2 is an introduction of preliminaries related to the research in this thesis. Chapter 3 represents a published manuscript for which Prof. Sistla and Prof. Žefran, my research advisor, are the primary authors. I implemented the train experiment and analyzed the results to prove the main results of the manuscript in the case study. Chapter 4 represents an unpublished manuscript which is extended from the published manuscript in Chapter 3. Under the guidance of Prof. Žefran, I proposed an improved algorithm for a type of generalized concurrent systems, and verified the efficiency of the algorithm using the train experiment. Chapter 5 represents an unpublished manuscript for which the major drive of the research is my internship in automotive industry. The simulation of an automotive project is used to demonstrate the effectiveness of the proposed method, which is intended to be incorporated into the verification of product design. I contributed to the majority of the writing in this chapter, and it will be submitted as a co-authored manuscript. Chapter 6 is the conclusion of the research presented in this thesis. The future directions of the research are discussed.

TABLE OF CONTENTS

CHAPTER		PAGE
1	**INTRODUCTION**	1
1.1	Related Work	6
1.2	Contributions	10
2	**PRELIMINARIES**	12
2.1	State and State variable	12
2.2	State-Space Model	12
2.3	Bayes Filter	13
2.3.1	Bayes Rule	13
2.3.2	Bayes Rule in Recursive State Estimation	13
2.3.3	Bayes Filter Algorithm	15
2.4	Hypothesis Testing	16
2.4.1	Goodness of Fit Testing	17
3	**PREVIOUS WORK**	19
3.1	Characterization of Monitorabilities	26
3.1.1	Strong Monitorability	26
3.1.1.1	Precise Characterization of Strong Monitorability	27
3.1.2	Monitorability	27
3.1.2.1	Characterization of Monitorability	28
3.1.3	Threshold-Based Monitors	28
3.1.3.1	Monitoring Safety Properties	29
3.1.3.2	Monitoring Liveness Properties	30
4	**HIERARCHICAL PARTICLE FILTERS FOR RUNTIME MONITORING OF CONCURRENT CYBER-PHYSICAL SYSTEMS**	31
4.1	Introduction	31
4.2	Problem Formulation	35
4.2.1	System Automata	35
4.2.1.1	Probabilistic Hybrid Automata with External Variables	36
4.2.1.2	Concurrent Dynamical Systems	38
4.2.1.3	Modeling of Concurrent CPSs	41
4.2.1.3.1	Link Variables and Local Variables	45
4.2.1.4	Concurrency in Subsystem Operation in a cPHA	48
4.2.1.5	Generic Algorithm for Subsystem Decomposition in cPHA	54
4.2.2	Property Automata	54
4.2.2.1	Streett Automata	56

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

4.2.3	Monitors	57
4.2.3.1	Monitorability	58
4.3	Hybrid State Estimation	59
4.3.1	Particle Filters	60
4.3.2	Particle Filtering Modifications	63
4.3.2.1	Risk-Sensitivity	63
4.3.2.2	Rao-Blackwellization	63
4.4	Hierarchical Particle Filter	64
4.4.1	Complexity Analysis	65
4.4.2	HPF Algorithm	66
4.4.2.1	Hierarchical adaptation	68
4.4.2.2	Complexity Reduction	71
4.4.2.3	More discussion	72
4.5	Experiment	78
4.5.1	Performance Evaluation	86
5	**FAULT DETECTION AND DIAGNOSIS FOR SAFETY-CRITICAL CYBER-PHYSICAL SYSTEMS**	89
5.1	Introduction	89
5.2	Background	93
5.2.1	DFMEA and Product Design	94
5.2.2	A Motivating Project	96
5.2.2.1	Auxiliary Transmission Fluid Pump Motor Controller	96
5.2.2.2	System Requirements	96
5.2.2.3	Software Design Diagram	97
5.2.2.4	Problems	99
5.3	Modeling	99
5.3.1	Hybrid Systems	99
5.3.2	Fault Models	101
5.3.2.1	System Level Faults	102
5.3.2.2	Component Level Faults	103
5.4	Structure of Property-Based Fault Detection and Diagnosis	104
5.5	Fault Detection	105
5.5.1	Monitorability	107
5.5.2	Design of property automaton	107
5.6	Fault Diagnosis	108
5.6.1	Importance Weights	108
5.6.2	Detection of Particle Inconsistency	110
5.6.3	Causes for particle inconsistency	114
5.6.3.1	Particle Propagation Depletion	115
5.6.3.2	System Modeling Inconsistency	115
5.6.3.3	Kolmogorov-Smirnov Test	116

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

5.7 Case study . . . 117
5.7.1 Controller Design for a Auxiliary Transmission Fluid Pump Motor . . . 117
5.7.1.0.1 ePump System Automaton with Modeled Faults . . . 118
5.7.1.0.2 ePump Property Automaton . . . 120

6 CONCLUSION AND FUTURE WORK . . . 123
6.1 Conclusion . . . 123
6.2 Future Work . . . 124
6.2.1 Output Decoupling Analysis . . . 124
6.2.1.1 Particle Filter Performance Related to Resampling . . . 126

APPENDICES . . . 128

CITED LITERATURE . . . 136

VITA . . . 146

LIST OF TABLES

TABLE		PAGE
I	ERRORS IN HYPOTHESIS TESTING	17
II	MOTOR STATUS VS. PWM OUTPUT DC	97
III	FMEA WORKSHEET	100

LIST OF FIGURES

FIGURE		PAGE
1	Runtime Monitoring Framework for CPS	32
2	Evolution Dependency of a Coupled cPHA	50
3	Evolution of the discrete state variable and the local variables depending on the link variables	51
4	Fully Locally Parallel Operation	52
5	Locally Parallel Operation	53
6	Velocity subsystem	80
7	Braking subsystem	81
8	Calculated link forces and accelerations for braking patterns of $N = 5$ cars	82
9	Property automaton	84
10	"True" hybrid state trajectory of the train system	87
11	Comparison of Estimated hybrid state trajectory by RBPF and Hierarchical PF	88
12	Stateflow of Software design for ePump Controller	98
13	PB-FDD architecture	106
14	System Automaton with Modeled Faults for ePump Control System	119
15	Property Automaton for ePump Control System	120
16	Sample Run for the ePump Control System	121
17	Unmodeled Fault Detection in ePump Control System	122

SUMMARY

A large variety of modern applications including autonomous cars, energy systems and medical devices are referred to as Cyber-Physical Systems (CPSs). Runtime monitoring of CPSs is an important method to verify if an execution of the system violates a given correctness property defined over the system state. Regarding safety properties of which the violation can cause disastrous consequences, an alarm raised timely allows mitigation or even corrective action to take before the disastrous consequences. A threshold based monitor was proposed in our previous work. It evaluates the rejection probability, which is the probability that the system is in one of the states that violate the property, against a predefined threshold. The evolution of property automata, which model the correctness properties, is driven by the internal state of the Probabilistic Hybrid Automaton (PHA), which is the mathematical dynamical model of the CPS. Therefore, the belief propagation is the core technique in this threshold based monitor. Particle Filter (PF), an Monte Carlo implementation of Bayes Filter, is employed for the estimation/belief propagation of the internal hybrid state.

However, for one class of CPS of our interest which consist of multiple concurrent components, existing particle filtering methods are often ineffective due to the state explosion and lead to particle depletion. Associated with the local and link variables, which are formally defined in concurrent probabilistic hybrid automata (cPHA), the interaction among concurrent subsystems is studied. We focus on a special type of concurrency, locally parallel operation where the state explosion would occur. Then, we propose a new particle filtering approach, hierarchical

particle filter, to reduce the computational complexity in the mode of locally parallel operation by decoupling the local variables in subsystems. A train example demonstrate the effectiveness and performance of the proposed algorithm.

An important application of runtime monitoring is Fault Detection and Diagnosis (FDD). It extends the the faults studied in FDD to a more general form, i.e., the violation of properties. In the proposed property-based fault detection and diagnosis (PB-FDD), hierarchical faults are defined as system level faults and component level faults. Furthermore, the causes for particle inconsistency which would fail the belief propagation are further studied. Besides the particle depletion occurred during particle propagation step in PF, another hypothesis is system model inconsistency, which corresponds to an unknown component level fault. Based on the distribution of the importance weights in PF, two hypothesis tests are formalized to first detect particle inconsistency and further diagnose the causes. The application of PB-FDD on an automotive project shows the practical meaning of the proposed framework in industrial product verification.

CHAPTER 1

INTRODUCTION

(Parts of) this chapter was previously published as Sistla, A. Prasad and Žefran, Miloš and Feng, Yao, "Runtime Monitoring of Stochastic Cyber-Physical Systems with Hybrid State", Runtime Verification (Springer Berlin Heidelberg, 2012), pp. 276--293 and Sistla, A. Prasad and Žefran, Miloš and Feng, Yao, "Monitorability of stochastic dynamical systems", in Proceedings of the 23rd international conference on Computer aided verification (Berlin, Heidelberg: Springer-Verlag, 2011), pp. 720--736.

Autonomous cars, energy systems and medical devices are examples of systems that integrate computation, communication and control; they are commonly called *Cyber-Physical Systems (CPSs)*. CPSs have attracted significant attention in the research community. A variety of applications and subjects are promising to develop the technology to transform our world [1]. According to the National Science Foundation, some important research applications in CPSs are "systems that respond more quickly (e.g., autonomous collision avoidance), are more precise (e.g., robotic surgery and nano-tolerance manufacturing), work in dangerous or inaccessible environments (e.g., autonomous systems for search and rescue, firefighting, and exploration), provide large-scale, distributed coordination (e.g., automated traffic control), are highly efficient (e.g., zero-net energy buildings), augment human capabilities, and enhance societal well-being (e.g., assistive technologies and ubiquitous health care monitoring and delivery)" [2].

One important class of CPSs are *safety-critical* cyber-physical systems. Important applications of the safety-critical CPSs include automobiles, aircraft, medical devices, robots and more, since failure can have catastrophic consequences. They are often complex systems that consist of subsystems/components whose operation is coordinated and integrated.

As a CPS evolves, its high-level behavior often changes in response to the environment or an external signal. Such changes are often considered to correspond to different *modes* of operation. Formally, modes can be associated with integer-valued (discrete) states whose evolution is governed by an automaton (with finite or countable number of states), while in each mode the physical system is described by a set of real-valued state variables whose evolution is governed by a set of differential/difference equations. Hybrid automata [3, 4] can be used to describe such systems. However, it is often necessary to capture the stochastic nature of the CPS evolution in the presence of noise and disturbances. Probabilistic hybrid automata (PHA) [5–7] have been proposed to describe such stochastic CPSs.

For safety-critical CPSs, the specifications that focus on the safety and reliability are *safety specifications*. For such systems, it is therefore necessary to verify that they do not violate desirable safety specifications. Formal verification of safety specifications has been shown to be undecidable for systems modeled as hybrid automata [8]. Testing can increase the confidence in a system design, but cannot guarantee its correctness. Runtime monitoring is an alternative way to testing or verification.

A monitor observes the inputs and outputs of the system and checks whether its behavior is consistent with the expected behavior. One advantage of monitors is the easy implementation

in principle. Also, there are many factors in the system design process and in the system implementation that might lead to faulty behavior. Another important advantage of monitors is that they can be designed independently of both the specifics of the design and the implementation. They are thus often able to detect such faults.

Safety specifications are typically described by *safety properties* and *liveness properties* [9,10]. These properties are expressed as formulas in an appropriate logic formalism such as *Linear Temporal Logic* (LTL) or equivalently, as automata [11] specified on the infinite sequences of the states of the system.

In [12], we presented a runtime monitoring approach for safety-critical CPSs and introduced so-called threshold based monitor. The decision to reject a system execution α with respect to a property was based on the rejection probability – the probability that the system is in one of the states that violate the property. Particle Filters (PFs) were employed to estimate the lower bound on the rejection probability. A PF is the sequential Monte Carlo implementation of a *Bayes filter* [13], a recursive state estimation/belief propagation algorithm. At each time step, like the Bayes filter algorithm, PF algorithm consists of two steps, *prediction* and *correction*. In the prediction step, a set of particles at time t is generated from the set of particles at time $t-1$ using the system evolution model. In the correction step, the observation model is used to make the distribution of the particle set consistent with the measured output.

The performance of a PF is inherently limited by the statistical nature of the representation of the belief distribution. In monitoring, the performance of the monitor critically depends on the performance of the PF. In this thesis, we focus on a phenomenon that may cause the PF

estimation – and thus monitor – to fail, *particle inconsistency*. Particle inconsistency can occur during the prediction step when no particles are generated that can adequately describe the behavior of the system suggested by the measured output. In the case of CPS with several modes, particle inconsistency would occur if none of the particles generated in the prediction step would correspond to the true mode of the system.

There are multiple causes of particle inconsistency in CPSs. One possibility is *particle propagation depletion*, and the other possibility is *system model inconsistency*. In the case of particle propagation depletion, a critical discrete transition in the system model or a possible trajectory of the continuous state is not captured by any particle in the predicted set. It may occur, for example, if the discrete transition has very low probability, or because a limited number of particles can not cover a large state space. In the case of system model inconsistency, the system model does not sufficiently describe the actual system behavior so no particle can correctly keep track of the actual state.

We show that in certain situations we can improve the PF algorithm to prevent particle inconsistency. For example, in monitoring a concurrent CPS with N components where the state space can be exponential in N, particle propagation depletion is likely to happen because the number of particles is inherently limited. By modeling the interaction among components represented as subsystems, we can explore different types of concurrency in the subsystem operation. In turn, we identify the type of concurrency that would result in state space explosion and propose a new algorithm that improves the performance of the particle filter.

Runtime monitoring is a rather general procedure that can be used beyond guaranteeing safety. An application of great industrial relevance that is extensively studied in this thesis is fault detection and diagnosis (FDD). In this case, the violation of the desired properties in monitoring can be mapped to faults in FDD, and the decision to reject a system execution can be interpreted as the detection of a fault. We call this application of monitoring *Property-Based* FDD (PB-FDD).

The faults in PB-FDD are defined at two levels: *system level* and *component level*. Compared with existing fault detection algorithms which rely on explicit modeling of faults described as violations of range constraints, PB-FDD describes the system level faults using a set of properties which are independent of the system model. This allows us to study much more complex system level faults that involve interactions between different parts of the system and timing constraints. The component level faults instead specifically identify possible causes of the system level faults at the component level; they are included in the model of each subsystem. By structuring the faults in this way, the fault can be detected when a system level fault specification is violated. But furthermore, by identifying which component level fault has been violated, the fault can also be diagnosed.

The fault detection algorithm in PB-FDD is based on the threshold based monitoring algorithm that uses PF to evaluate the rejection probability. So, particle inconsistency is still a challenge in PB-FDD. However, one of the possible causes, system model inconsistency, has special meaning in the context of FDD.

Most model-based fault detection techniques assume the system model as well as the fault model are well established with the correct operational parameters and have sufficient knowledge of the environment where the system operates. However, every system may fail unexpectedly, even carefully designed and tested systems [14]. One reason for this is that physical components degrade over time. Another is that the computational process rarely has complete knowledge of the situations that the physical components may encounter [15]. For example, in the automotive industry, Failure Mode and Effect Analysis (FMEA) is performed to guarantee the reliability of the system. FMEA is a systematic analysis of the failure modes, and its effect, cause and control strategy. It provides good knowledge of faults/failures, but it still does not guarantee the completeness of the analysis. Moreover, an unexpected fault may be related to a design defect that requires additional analysis to diagnose.

As described, the diagnosis of an unexpected fault is an important part in FDD of practical significance. However, by its definition, an unexpected fault is not explicitly modeled and is therefore difficult to diagnose. In PB-FDD, the unexpected faults are described as unknown component level faults. The occurrence of an unexpected component level fault would lead to particle inconsistency in PF due to system model inconsistency. Therefore, we propose a formal framework to detect such particle inconsistency and thus unknown component level faults through hypothesis testing.

1.1 Related Work

(Parts of) this chapter was previously published as Sistla, A. Prasad and Žefran, Miloš and Feng, Yao, "Runtime Monitoring of Stochastic Cyber-Physical Systems with Hybrid State",

in Khurshid, Sarfraz and Sen, Koushik, ed., Runtime Verification (Springer Berlin Heidelberg, 2012), pp. 276--293 and Sistla, A. Prasad and Žefran, Miloš and Feng, Yao, "Monitorability of stochastic dynamical systems", in Proceedings of the 23rd international conference on Computer aided verification (Berlin, Heidelberg: Springer-Verlag, 2011), pp. 720--736.

Fault detection and diagnosis is a well established subject (see e.g. textbooks [16,17]) spanning a variety of disciplines ranging from signal processing, to control and Artificial Intelligence (AI). In recent years, a number of researchers have developed approaches for fault detection and diagnosis of hybrid systems. A problem that has been extensively studied is monitoring and diagnosis of hybrid automata [15,18–25]. Basically, it is to detect when a fail state of the automaton is reached. Similar work has been done in the AI community on failure detection and recovery from failures using Hidden Markov models [26]. In most cases, these works employ techniques that depend on the specific possible modes of failure and require the system to be monitored for correct functioning.

There has been much work done in the literature on monitoring violations of safety properties in distributed systems, for example [27]. This work assumes that it can fully observe the system state and it instruments the program with commands to gather its state information and use it for monitoring. In contrast, we assume that the system is not directly observable. The works [28,29] consider monitoring non-hybrid systems whose states are not observable. In these works, the monitored property refers to the output values and not the system state variables, which is required for complex robot systems. A method for monitoring and checking quantitative and probabilistic properties of real-time systems has been given in [30,31]. These works take

specifications in a probabilistic temporal logic (called CSL) and monitor for its satisfaction. The probabilities are deduced from the repeated occurrence of events in a computation. The work presented in [32] considers monitoring interfaces for faults using game-theoretic framework. Run-time monitoring is used to verify that the interface has a winning strategy. *Conservative* run time monitors were proposed in [33, 34]. In this scheme, one identifies a safety property that implies the given property f (in general, f is the intersection/conjunction of a safety and a liveness property). Monitoring of programs modeled as Hidden Markov Models subject to observations has been studied in [35]. None of these works is intended for monitoring of hybrid systems.

A wealth of literature is available for the modeling and analysis of hybrid systems and we refer the reader to the overview articles [36,37] and the books [38,39]. A number of models were proposed for stochastic hybrid system and introduced different types of randomness. Stochastic hybrid system proposed by Hu [40] introduced stochastic differential equations (SDE) in the evolution of continuous state, and the discrete transitions are deterministic specified by a guard function on the continuous state. A piecewise deterministic Markov Process [5] has deterministic continuous state evolution in each discrete state, but assigns a stochastic transition frequency for transitions between discrete states. Switching diffusion processes [41] combine SDE and controlled Markov Chain to capture the randomness both in continuous and discrete states. Several review papers [6, 7, 42] list and compare a variety types of stochastic hybrid system models. In monitoring of such systems, safety requirements are described by a set of system states which are permissible, or equivalently, by a set of system states that are forbidden. A

closely related problem is checking liveness properties, where in general it is required that a set of states is visited infinitely often. Formally, safety and liveness properties can be described using temporal logic [43,44]. Safety and liveness verification thus becomes a verification problem for a hybrid automaton modeling the robotic system [8,45]. It was shown that except for the simplest hybrid automata, this verification problem is undecidable [8,46].

Traditional model based fault diagnosis techniques have been extended for hybrid systems in [47–50]. In these works it is typically assumed that the system is observable, and that different discrete modes can be distinguished (the system is discrete-mode observable [22]), and that switching between different modes is slow enough that the correct mode can be precisely identified. But none of these works address the general problem of monitoring system behaviors against specifications given in an expressive formal system such as the hybrid automata. Furthermore, they do not address the problem of monitoring liveness properties, and they can not deal with systems where the evolution of discrete modes is uncertain.

For stochastic hybrid systems, one important approach used in FDD is hybrid state estimation [15,24,51–53]. In these works, multiple mode estimation approach [51,53,54] applies a bank of filters on each discrete mode, and particle filters [24,55–57] are used to handle more general estimation problems with less restrictions of the system model. However, both methods require large computational resources for complex systems. To improve the performance of estimation, [24] proposed Interacting Multiple Model (IMM) particle filter which combines particle filtering with multiple model approach. Rao-Blackwellised Particle Filter (RBPF) [55,58,59] combines particle filtering with Kalman filter, an optimal filter for finite-dimensional linear sys-

tems to achieve better accuracy . Gaussian Particle Filter (GPF) [56] looks one-step ahead to make the algorithm computationally efficient. Several adaptive particle filters [60–62] are proposed to adapt the number of particle and the proposal distribution at each iteration. These works all tried to improve the performance of particle filters in general. Our work targets on a specific weakness of particle filters which is not completely solved using these proposed approaches. One closely related work is [52]. A concurrent hybrid system, which is also a focus in this thesis, is modeled and analyzed in this work. The interconnection among components in the concurrent system is represented by a causal graph and by partitioning the causal graph, the filter used on the whole causal graph is reduced to multiple filters to individual pieces from the partition.

FDD approaches with unknown faults are of interest in many works for the past several decades and we refer the readers to some surveys papers [63, 64]. It is typically assumed that the unknown disturbances are structured as parameter changes or additive noises on the nominal system model and the residual is generated such that it is not sensitive to the unknown faults. However, unknown faults could be the targeted concern in diagnosis, i.e., the occurrence of unknown faults needs to be detected. There works are mainly on the isolation of the unknown faults rather than diagnosis.

1.2 Contributions

The aim of this work is to propose a fault detection and diagnosis strategy for safety-critical CPSs. Particle filter that is employed as an efficient tool is studied as the core part of the work. The main contributions are as follows:

- A concurrent PHA (cPHA) with subsystems is proposed to model a concurrent CPS, and different types of concurrency in subsystem operation are studied with notions of local and link variables.
- A novel PF, hierarchical PF with reduced complexity, is proposed when the cPHA is running in a specific type of concurrency, locally parallel operation, which introduces exponential complexity using the classical PF algorithm. The effectiveness of the algorithm is shown in a simulated train example.
- The notions of system level faults and component level faults are proposed and modeled according to the scopes of the system requirements and component requirements, respectively. The definition of the system level faults introduces a general way to describe a fault without being referred to a nominal system model.
- Hypothesis testing is formalized to detect particle inconsistency and further system model inconsistency using the importance weights in PF algorithm.

CHAPTER 2

PRELIMINARIES

2.1 State and State variable

The state variables of a dynamical system [65] are a smallest ordered set of variables $\mathbf{x}$ such that given the value of the variables at t_1, the values of the variables at any $t > t_1$ can be determined using a system model. A possible value of the state variables is called a *state*. At each time t, the value of the state variables, that is, the state, is denoted by x_t.

In stochastic dynamical systems, the evolution of the state can be described by stochastic differential/difference equations. The state x_t is a realization of $\mathbf{x}_t$ at time t whose probability distribution function is $p(x_t)$.

2.2 State-Space Model

The state space representation of a discrete-time, time-invariant stochastic dynamical system is given by the following equations [65]:

$$x_{t+1} = f(x_t, u_{t+1}) + w_{x,t} \tag{2.1}$$

$$y_t = g(x_t, u_t) + w_{y,t} \tag{2.2}$$

where x_t is the state, u_t is the input (value of the input variables $\mathbf{u}$) and y_t is the output (value of the output variables $\mathbf{y}$), respectively, at time t; $\{w_{x,t}\}$ and $\{w_{y,t}\}$ are generated by sequences of

independent identically-distributed (i.i.d.) random variables that are independent of the initial condition; the mappings f and h are time-invariant and continuous in x and u.

A dynamical system is *autonomous* [66] if there are no independent input variables $\mathbf{u}$.

2.3 Bayes Filter

2.3.1 Bayes Rule

Considering the random variables $\boldsymbol{a}$ and $\boldsymbol{b}$ defined on $\mathbb{R}$, Bayes rule is stated by the equation [67]:

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)} \tag{2.3}$$

In Bayesian applications, $p(a|b)$ is the posterior distribution of $\boldsymbol{a}$ given $\boldsymbol{b} = b$, and $p(a)$ is the prior distribution of $\boldsymbol{a}$, and $p(b|a)$ is the likelihood function, and $p(b)$ is probability that $\boldsymbol{b} = b$.

Consider a more general form of Bayes rule given an additional variable $\boldsymbol{c}$, we have

$$p(a|b,c) = \frac{p(b|a,c)p(a|c)}{p(b|c)} \tag{2.4}$$

2.3.2 Bayes Rule in Recursive State Estimation

The stochastic dynamical system model has two parts: the evolution model described by (2.1) and the observation model described by (2.2). Equivalently, the equations give the conditional distributions:

$$\begin{cases} p(x_t|x_{t-1}, u_t) \\ \\ p(y_t|x_t, u_t) \end{cases}$$

Both models are assumed *Markovian*, i.e.,

$$p(x_t|X_{t-1}, U_t, Y_{t-1}) = p(x_t|x_{t-1}, u_t) \tag{2.5}$$

$$p(y_t|X_t, U_t, Y_{t-1}) = p(y_t|x_t, u_t) \tag{2.6}$$

where $X_{t-1} = \{x_1, \ldots, x_{t-1}\}$, $U_{t-1} = \{u_1, \ldots, u_{t-1}\}$ and $Y_{t-1} = \{y_1, \ldots, y_{t-1}\}$ are the sequences of states, input and output, respectively. In other word, in the evolution model, given x_{t-1} and u_{t-1}, $\mathbf{x}_t$ is conditionally independent of all the other history states and input; similarly, for the observation model, given x_t and u_t, $\mathbf{y}_t$ is conditionally independent of all the other history states and input.

The state variables of a system are often not observed directly. The knowledge of the state variables is referred to as the *belief*. The belief distribution of the state variables at time t, $\mathbf{x}_t$, is the conditional probability distribution given all the observed data up to time t, $p(x_t|U_t, Y_t)$, denoted as $bel(x_t)$.

By substituting a by x_t, b by y_t and c by (U_t, Y_{t-1}) in Bayes rule (2.4), we get:

$$p(x_t|U_t, Y_t) = \frac{p(y_t|x_t, U_t, Y_{t-1})p(x_t|U_t, Y_{t-1})}{p(y_t|U_t, Y_{t-1})} \tag{2.7}$$

$p(y_t|x_t, U_t, Y_{t-1})$ is simplified to $p(y_t|x_t, u_t)$ according to (2.6), then (2.7) is reduced to

$$bel(x_t) = \frac{p(y_t|x_t, u_t)p(x_t|U_t, Y_{t-1})}{p(y_t|U_t, Y_{t-1})} \tag{2.8}$$

The denominator,

$$p(y_t|U_t, Y_{t-1}) = \sum_{x_t} p(y_t|x_t, u_t)p(x_t|U_t, Y_{t-1})$$

is computed to the normalizing constant $\frac{1}{\eta}$ using the law of total probability [67].

$p(x_t|U_t, Y_{t-1})$ is the belief distribution of $\boldsymbol{x}_t$ without the knowledge of current input and output observations y_t. We denote it as $\overline{bel}(x_t)$. It can be obtained by the law of total probability:

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, U_t, Y_{t-1})p(x_{t-1}|U_{t-1}, Y_{t-1})dx_{t-1}$$

According to (2.5), it can be simplified to be

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t)p(x_{t-1}|U_{t-1}, Y_{t-1})dx_{t-1}$$

where $p(x_{t-1}|U_{t-1}, Y_{t-1})$ is the belief distribution of $\boldsymbol{x}_{t-1}$, $bel(x_{t-1})$.

In brief, the belief distribution is recursively obtained by

$$bel(x_t) = \eta p(y_t|x_t, u_t) \int p(x_t|x_{t-1}, u_t)bel(x_{t-1})dx_{t-1} \tag{2.9}$$

2.3.3 Bayes Filter Algorithm

Recursively, the generic algorithm for *Bayes filter* is depicted as follows [13]:

Algorithm 1 Bayes filter $(bel(x_{t-1}), u_t, y_t)$

for all x_t **do**

Prediction: $\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}$

Correction: $bel(x_t) = \eta p(y_t|x_t, u_t) \overline{bel}(x_t)$

2.4 Hypothesis Testing

Hypothesis Testing is a form of statistical inference. It is a decision process which accept or reject a hypothesis according to the observed data. A general hypothesis test includes the following steps:

1. Null and Alternative Hypotheses formulation
2. Test statistics selection
3. Decision rule making
4. Decision to accept or reject the null hypothesis based on observed data

The null hypothesis is the hypothesis to test, denoted as H_0. Null hypothesis is usually the default statement that we wish to prove false. The alternative hypothesis is the negation of the null hypothesis, denoted as H_1. H_0 is a simple hypothesis [68] if it only leads to one distribution; otherwise, H_0 is a composite hypothesis [68].

A hypothesis testing is associated with two types of errors, *Type I error* and *Type II error*. Basically, a type I error is the incorrect rejection of a true null hypothesis, and a type II error is

	Hypothesis H_o is true	Hypothesis H_o is false
Hypothesis H_o is accepted	Right Decision	Type II error
Hypothesis H_o is rejected	Type I error	Right Decision

TABLE I: ERRORS IN HYPOTHESIS TESTING

an incorrect acceptance of a false hypothesis, as described in Table I. The probability of these two types of error are denoted as γ and β, respectively.

Significance Level [68]. The significance level is the maximum probability of erroneously rejecting a true null hypothesis.

If H_0 is a simple hypothesis, the significance level is the the probability of a type I error, γ.

2.4.1 Goodness of Fit Testing

An important class of hypothesis testing is the goodness of fit testing. It is a non-parametric test used to measure how well the observed data fit the hypothesized probability distribution model. Common tests include Chi-squared test, Kolmogorov–Smirnov test, Anderson–Darling test and Cramér–von Mises test [68–70].

Null and Alternative Hypotheses Formulation

H_0 : The observed data is drawn from the reference distribution $F^*(x)$

H_1 : The observed data is not drawn from the reference distribution $F^*(x)$

Test Statistic Selection

A test statistic is selected differently in different tests, but commonly, the sampling distribution of the test statistic is known under the null hypothesis H_0. For example, the test statistic selected in Chi-squared test follows a chi-squared distribution, and the test statistic selected in Kolmogorov–Smirnov test follows a Kolmogorov distribution.

Decision Rule Making

Critical Region [68]. The critical region is the set of all points in the sample space of the test statistic that result in null hypothesis rejection.

For a given significance level γ, a corresponding critical region is constructed. The decision rule is that H_0 is rejected if the test statistic from observed data is in the critical region, and accepted otherwise.

CHAPTER 3

PREVIOUS WORK

(Parts of) this chapter was previously published as Sistla, A. Prasad and Žefran, Miloš and Feng, Yao, "Runtime Monitoring of Stochastic Cyber-Physical Systems with Hybrid State", Runtime Verification (Springer Berlin Heidelberg, 2012), pp. 276--293 and Sistla, A. Prasad and Žefran, Miloš and Feng, Yao, "Monitorability of stochastic dynamical systems", in Proceedings of the 23rd international conference on Computer aided verification (Berlin, Heidelberg: Springer-Verlag, 2011), pp. 720--736.

This chapter presents our preliminary research on monitoring partially observable systems with respect to a single property given by an automaton or a logical formula using Hidden Markov Chains (HMC) as models, and shows how the formalism can be used for systems specified by probabilistic hybrid automata (PHA).

We start by providing precise definitions of accuracies of monitors and notions of monitorability and strong monitorability. Then, we exactly characterize systems that satisfy these properties. For ease of presentation we assume that the continuous variables are quantized (but not necessarily bounded); as a result, HMCs can be used to model such systems (see [71] for details). However, all these results were also extended to hybrid systems that employ continuous as well as discrete variables, without quantization, using *Extended Hidden Markov systems* [12].

Sequences. Let S be a set. Let $\sigma = s_0, s_1, \ldots$ be a possibly infinite sequence over S. For any $i \geq 0$, $\sigma[0, i]$ denotes the prefix of σ up to s_i. If α_1 is a finite sequence and α_2 is either a

finite or an ω-sequence then $\alpha_1\alpha_2$ denotes the concatenation of the two sequences in that order. We let S^*, S^ω denote the set of finite sequences and the set of infinite sequences over S. If $C \subseteq S^\omega$ and $\alpha \in S^*$ then αC denotes the set $\{\alpha\beta : \beta \in C\}$. We use subsets of S^ω to denote properties of infinite sequences. Given a property $C \subseteq S^\omega$ and $\sigma \in S^\omega$, we say that σ satisfies C iff $\sigma \in C$.

Safety Properties. Safety Properties For any $\sigma \in S^\omega$, let prefixes(σ) denote the set of prefixes of σ and for any $C \subseteq S^\omega$, let prefixes(C) $= \cup_{\sigma \in C}$(prefixes(σ)). We say that $C \subseteq S^\omega$ is a *safety* property if the following condition holds: for any $\sigma \in S^\omega$, if prefixes(σ) $\subseteq$ prefixes(C) then $\sigma \in C$. For any $C \subseteq S^\omega$, let $closure(C)$ be the smallest safety property such that $C \subseteq closure(C)$. We say that a finite sequence $\sigma \in S^*$, violates the safety property C iff $\sigma \notin$ prefixes(C).

Liveness Properties. Properties Let S^* be the set of all finite sequences on S. A property $C \subseteq S^\omega$ is a liveness property [72] if there exist an infinite sequence $\beta \in S^\omega$ such that for every prefix of elements in C, $\alpha \in S^*$, the concatenation sequence $\alpha\beta \in C$. In other word, a property $C \subseteq S^\omega$ is not a liveness property if there exist a finite sequence $\alpha \in S^*$such that for every infinite sequence $\beta \in S^\omega$, the concatenation sequence $\alpha\beta \notin C$.

Automata. We use deterministic Streett automata to specify properties over infinite sequences. Each such automaton $\mathcal{P}$ has an input alphabet Σ and defines a language $L(\mathcal{P}) \subseteq \Sigma^\omega$. These automata can have countable number of states. Throughout Sections 3 and 3.1, an automaton refers to a Streett automaton. We also consider Büchi automata, a subclass of Streett automata, and a subclass of Büchi automata called *safety automata* whose language is a safety

property. Linear Temporal Logic (LTL) is a logical formalism that can used to satisfy safety and liveness properties elegantly. LTL formulas can be easily translated into equivalent Streett Automata. For this reason, we concentrate on using automata for specifying properties.

Markov Chains. We assume that the reader is familiar with basic probability theory, random variables and Markov chains. We consider stochastic systems given as Markov Chains [73] and monitor their computations for satisfaction of a given property specified by an automaton or a temporal formula. A Markov chain $G = (S, R, \phi)$ is a triple satisfying the following: S is a set of countable states; $R \subseteq S \times S$ is a total binary relation (i.e., for every $s \in S$, there exists some $t \in S$ such that $(s,t) \in R$); and $\phi : R \rightarrow (0, 1]$ is a probability function such that for each $s \in S$, $\sum_{(s,t)\in R} \phi((s,t)) = 1$. Note that, for every $(s,t) \in R$, $\phi((s,t))$ is non-zero. Intuitively, if at any time the system is in a state $s \in S$, then in one step, it goes to some state t such that $(s,t) \in R$ with probability $\phi((s,t))$. A finite path p of G is a sequence $s_0, s_1, \ldots, s_n$ of states such that $(s_i, s_{i+1}) \in R$ for $0 \leq i < n$. For any such p, if $n > 0$, then let $\phi(p) = \prod_{0\leq i<n} \phi((s_i, s_{i+1}))$; if $n = 0$ then let $\phi(p) = 1$. An infinite path of G is an infinite sequence of states $s_0, s_1, ...$ such that $\forall i \geq 0$, $(s_i, s_{i+1}) \in R$. We let $Paths(G)$ and $Paths(G, s)$ for any $s \in S$, respectively, denote the set of all infinite paths in G and the set of all infinite paths in G starting from s.

For any Markov chain G, as given above, we define a class $\mathcal{E}_G$ of measurable sets of infinite sequences over S. $\mathcal{E}$ is the σ-algebra [73] generated by sets of sequences of the form pS^ω where $p \in S^*$. Now, for any system state $r \in S$, we define a probability function $\mathcal{F}_{G,r}$ defined on $\mathcal{E}_G$ as follows. Intuitively, for any $C \in \mathcal{E}_G$, $\mathcal{F}_{G,r}(C)$ denotes the probability that a sequence of states generated from the system state r, is in C. $\mathcal{F}_{G,r}$ is the unique probability measure satisfying all

the probability axioms [73], such that for every $p \in S^*$ and $C = pS^\omega$, if p is the empty sequence then $\mathcal{F}_{G,r}(C) = 1$, if p is a finite path starting from state r then $\mathcal{F}_{G,r}(C) = \phi(p)$, otherwise $\mathcal{F}_{G,r}(C) = 0$.

Although, for convenience, we have considered all sequences in S^ω in defining $\mathcal{E}_G$, sequences that are not paths in G do not contribute to the probability of any $C \in \mathcal{E}_G$, as shown below. Since S is a countable set, it is not difficult to see that $Paths(G)$, $Paths(G,r) \in \mathcal{E}_G$. Further more, for any $C \in \mathcal{E}_G$, it can be shown that $\mathcal{F}_{G,r}(C) = \mathcal{F}_{G,r}(C\mathcal{P}Paths(G)) = \mathcal{F}_{G,r}(C\mathcal{P}Paths(G,r))$.

For any $D \in \mathcal{E}_G$, we let $\mathcal{F}_{G,r|D}$ denote the conditional probability function given D; formally, for any $C, D \in \mathcal{E}_G$, $\mathcal{F}_{G,r|D}(C) = \frac{\mathcal{F}_{G,r}(C\mathcal{P}D)}{\mathcal{F}_{G,r}(D)}$. For any $\alpha \in S^*$ and $C = \alpha S^\omega$, we let $\mathcal{F}_{G,r}(\alpha)$ denote the probability $\mathcal{F}_{G,r}(C)$ and $\mathcal{F}_{G,r|\alpha}$ denote the conditional probability function $\mathcal{F}_{G,r|C}$. For a set $C \subseteq S^*$, we let $\mathcal{F}_{G,r}(C)$ denote $\mathcal{F}_{G,r}(CS^\omega)$.

We will use automata to specify properties over sequences of states of a Markov chain G. The input symbols to the automata are states of G, i.e., members of S. It has been shown that, for any automaton $\mathcal{P}$, $L(\mathcal{P})$ is measurable [74]. We will be interested in monitoring sequences of states of a system modeled by G, i.e., computations generated by G, to ensure that it satisfies the property given by an automaton $\mathcal{P}$. However, the monitor can not observe the actual states of the system.

Hidden Markov Chains. A Hidden Markov Chain (HMC) [75] $H = (G, O, r_0)$ is a triple where $G = (S, R, \phi)$ is a Markov chain, $O : S \to \Sigma$ is the output function and $r_0 \in S$ is the initial state. Intuitively, for any $s \in S$, $O(s)$ is the output generated in state s and this output is generated when ever a transition entering state s is taken. The generated symbols

become inputs to the monitor. H is called Hidden Markov chain because one only observes the outputs generated in each state but not the actual state[1]. We extend the output function O to paths of G as follows. For any finite or infinite path $p = s_0, s_1, \ldots, s_i, \ldots$ in G, $O(p) = O(s_0), O(s_1), \ldots, O(s_i), \ldots$. For any finite or infinite sequence α in $\Sigma^* \cup \Sigma^\omega$, we let $O^{-1}(\alpha)$ denote the set of $p \in S^* \cup S^\omega$ such that $O(p) = \alpha$. For any $C' \subseteq \Sigma^* \cup \Sigma^\omega$, we let $O^{-1}(C') = \cup_{\alpha \in C'}(O^{-1}(\alpha))$.

For any HMC H as given above, we define a class $\mathcal{E}_H$ of sets of infinite sequences over Σ and for any $r \in S$, we define a probability measure $\mathcal{F}_{H,r}$ on $\mathcal{E}_H$ as follows. $\mathcal{E}_H$ is the σ-algebra generated by the sets $\alpha\Sigma^\omega$ for $\alpha \in \Sigma^*$. For any system state $r \in S$ and $C' \in \mathcal{E}_H$, $\mathcal{F}_{H,r}(C') = \mathcal{F}_{G,r}(O^{-1}(C'))$. Intuitively, $\mathcal{F}_{H,r}(C')$ denotes the probability that an output sequence generated from the system state r, is in C'.

Quantized Probabilistic Hybrid Automata. Quantized probabilistic hybrid automata (QPHA) are probabilistic hybrid automata [25] whose continuous variables are quantized. Their semantics is given by a HMC, but they provide a convenient formalism for specifying systems. A quantized probabilistic hybrid automaton $\mathcal{A}$ is a tuple $(Q, V, \Delta t, \mathcal{E}, \mathcal{T}, c_0)$ where Q is a finite set of *discrete* states (modes); $V = \{x_q\}_{q \in Q} \cup \{y_q\}_{q \in Q} \cup \{n_q\}_{q \in Q}$ is the finite set of real-valued *continuous*, *output* and *noise* variables, respectively, that will be assumed to be quantized;Δt is the sampling time; $\mathcal{E}$ is a function that with each $q \in Q$ associates a set $\mathcal{E}(q)$ of difference

---

[1]In the traditional definition of HMCs considered in literature, the output of a state can be any symbol in Σ generated with a probability distribution that is specific to the state; since Σ is a countable set, it is not difficult to see that by duplicating each state as many times as there are output symbols, such a HMC can be converted into an equivalent HMC consistent with our model.

equations describing the evolution of the continuous state and the output at time $t + \Delta t$ as a function of the state at t and the noise variables; $\mathcal{T}$ is a function that assigns to each $q \in Q$ a set of *transitions* which are triples of the form (ϕ, p, ψ), where ϕ is the *guard* which is a predicate over the set of continuous and discrete variables, p is a probability distribution over the *discrete states*, and ψ is the *reset relation* which is a set of assignments that update or reset some of the continuous variables; and c_0 denotes the initial discrete and continuous states of the automaton. If no noise variables are present and for each transition triple (ϕ, p, ψ) associated with each discrete state q, the set of discrete states for which p specifies non-zero probability is a singleton, then QPHA is a normal hybrid automaton (QHA) [3]. A property can be specified if an appropriate acceptance condition is defined for a QHA. In fact, in this case the QHA is equivalent to a Streett automaton.

Within each mode q, the evolution of the QPHA is given by the difference equations. For simplicity we will assume that equations describing the continuous evolution of the system have been linearized[1] and discretized [78], resulting in a set of linear difference equations. Since the continuous, noise and output variables are assumed to be quantized, these transitions can be interpreted as an HMC. When a guard ϕ_i becomes satisfied, a transition takes place from q to some target mode q' according to the probability distribution p_{qi}. The overall evolution of the QPHA can be thus interpreted as the evolution of an appropriate HMC. See [25,52] for details.

---

[1]The extensive literature on piecewise-linear approximations [76, 77] provides a justification for such an assumption.

Monitors. A monitor $M : \Sigma^* \to \{0, 1\}$ is a function with the property that, for any $\alpha \in \Sigma^*$, if $M(\alpha) = 0$ then $M(\alpha\beta) = 0$ for every $\beta \in \Sigma^*$. For an $\alpha \in \Sigma^*$, we say that M rejects α, if $M(\alpha) = 0$, otherwise we say M accepts α. Thus if M rejects α then it rejects all its extensions. For an infinite sequence $\sigma \in \Sigma^\omega$, we say that M rejects σ iff there exists a prefix α of σ that is rejected by M; we say M accepts σ if it does not reject it. Let $L(M)$ denote the set of infinite sequences accepted by M. It is not difficult to see that $L(M)$ is a safety property and $O^{-1}(L(M))$ is measurable (it is in $\mathcal{E}_G$).

Note that for a monitor to be implementable, M has to be a computable function as observed in [79] for deterministic systems.

Accuracy Measures. Let $\mathcal{P}$ be an automaton on states of H. The *acceptance accuracy* of M for $\mathcal{P}$ with respect to the HMC H, denoted by $AA(M, H, \mathcal{P})$, is the probability $\mathcal{F}_{G,r_0|L(\mathcal{P})}(O^{-1}(L(M)))$ where r_0 is the initial state of H. Intuitively, it is the conditional probability that a sequence generated by the system is accepted by M, given that it is in $L(\mathcal{P})$. We define the *rejection accuracy* of M for $\mathcal{P}$ with respect to H, denoted by $RA(M, H, \mathcal{P})$, to be the probability that a sequence generated by the system is rejected by M, given that it is not in $L(\mathcal{P})$; formally, it is the probability $\mathcal{F}_{G,r_0|C}(D)$, where C, D are the complements of $L(\mathcal{P})$ and $O^{-1}(L(M))$ respectively.

Monitorability. We say that a system H is *strongly monitorable* with respect to an automaton $\mathcal{P}$ if there exists a monitor M such that $AA(M, H, \mathcal{P}) = RA(M, H, \mathcal{P}) = 1$, i.e., both of its accuracies are 1. We say that a system H is *monitorable* with respect to an automaton $\mathcal{P}$ if for every $x \in [0, 1)$ there exists a monitor M such that $AA(M, H, \mathcal{P}) \geq x$ and

$RA(M, H, \mathcal{P}) \geq x$. Strong monitorability is a property that is difficult to satisfy. In the next section, we give necessary and sufficient conditions for these properties to be satisfied.

It is worth noting that monitorability, while related to the classical notion of observability, is fundamentally different from it. It is not difficult to construct hybrid systems that are not observable or even discrete-state observable but are monitorable.

3.1 Characterization of Monitorabilities

Let $H = (G, O, r_0)$ be a HMC where $G = (S, R, \phi)$ is the associated Markov chain. Let $\mathcal{P}$ be an automaton with input alphabet S. $H, G, \mathcal{P}$ are fixed throughout this section unless otherwise stated.

3.1.1 Strong Monitorability

We define a set of infinite paths $OverlapSeq(H, \mathcal{P})$ that intuitively captures non-trivial overlap, based on the generated outputs, between sets of infinite paths of G that are accepted and those that are rejected by $\mathcal{P}$. We say that a finite path p in G is *good* if it starts from r_0 and the set C of infinite paths, accepted by $\mathcal{P}$, having p as a prefix, has non-zero measure, i.e., $\mathcal{F}_{G,r_0}(C) > 0$ where $C = (pS^{\omega}\mathcal{P}Paths(G, r_0)\mathcal{P}L(\mathcal{P}))$. Let $GoodPaths(H, \mathcal{P})$ be the set of infinite paths in G having only good prefixes. Now we define $OverlapSeq(H, \mathcal{P}) = (Paths(G, r_0) - L(\mathcal{P}))\mathcal{P}\, O^{-1}(O(GoodPaths(H, \mathcal{P})))$. Intuitively, $OverlapSeq(H, \mathcal{P})$ is the set of $p \in Paths(G, r_0)$ such that p is rejected by $\mathcal{P}$ and each of its prefix generates the same output sequence as some good path in G, i.e., it can not be distinguished from a good path based on the outputs.

3.1.1.1 Precise Characterization of Strong Monitorability

The HMC H is strongly monitorable with respect to $\mathcal{P}$ iff $\mathcal{F}_{G,r_0}(OverlapSeq(H,\mathcal{P})) = 0$, i.e., the measure of overlap sequences is zero. Also, given a finite HMC H and a finite state automaton $\mathcal{P}$, the problem of determining if H is strongly monitorable with respect to $\mathcal{P}$ is PSPACE-complete.

If H is strongly monitorable with respect to $\mathcal{P}$, using the techniques employed in the proof of the above results, we can construct a monitor M' both of whose accuracies equal 1. M' simply constructs a deterministic automaton $\mathcal{C}$ and runs it on the output generated by H. It rejects iff $\mathcal{C}$ rejects. M' does not estimate any probabilities.

3.1.2 Monitorability

Consider any $\alpha \in \Sigma^*$. According to our notation $\mathcal{F}_{H,r}(\alpha)$ is the probability that H initially generates the output sequence α, i.e., outputs α during the first n states, where n is the length of α. Let $\alpha \in \Sigma^*$ be such that $\mathcal{F}_{H,r}(\alpha) > 0$. Now, we define a probability measure $GoodProb(\alpha)$ which is the conditional probability that an execution of the system H that initially generated the output sequence α is accepted by $\mathcal{P}$,i.e., the execution is good. Formally, $GoodProb(\alpha) = \mathcal{F}_{H,r_0|C}(L(\mathcal{P}))$ where $C = O^{-1}(\alpha)S^\omega$. Let $BadProb(\alpha) = 1 - GoodProb(\alpha)$. Observe that $BadProb(\alpha)$ is the conditional probability that an execution of the system that initially generated the output sequence α is rejected by $\mathcal{P}$, i.e., is bad.

Recall that for any $\beta \in \Sigma^\omega$ and integer $i \geq 0$, $\beta[0,i]$ denotes the prefix of β of length $i+1$. Now, let $OneSeq(H,\mathcal{P})$ be the set of all $\beta \in \Sigma^\omega$ such that $\lim_{i\to\infty} GoodProb(\beta[0,i])$ exists and its value is 1. Similarly, let $ZeroSeq(H,\mathcal{P})$ be the set of all $\beta \in \Sigma^\omega$ such that the above

limit exists and is equal to 0. Let $ZeroOneSeq(H, \mathcal{P}) = OneSeq(H, \mathcal{P}) \cup ZeroSeq(H, \mathcal{P})$. It has been shown in [71] that the sets $ZeroSeq(H, \mathcal{P})$, $OneSeq(H, \mathcal{P})$ and their union are all measurable.

3.1.2.1 Characterization of Monitorability

The HMC H is monitorable with respect to $\mathcal{P}$ iff $\mathcal{F}_{H,r_0}(ZeroOneSeq(H, \mathcal{P})) = 1$, i.e., almost all of its output sequences are zero- or one-sequences. Also, the problem of deciding if a finite state HMC is monitorable with respect to a finite state automaton is undecidable.

3.1.3 Threshold-Based Monitors

Although, the problem of determining if a HMC is monitorable with respect to an automaton $\mathcal{P}$ for finite state systems, is undecidable, we can give sufficient conditions that ensure monitorability. Intuitively, H is going to be monitorable with respect to an automaton $\mathcal{P}$ if the statistics (i.e., probability distributions) of outputs generated in paths that are accepted by $\mathcal{P}$ is different from those generated in paths that are rejected by $\mathcal{P}$. Many times, this property may be known. For example, consider a system that can fail, i.e., can get into any of a set of failure states and once it gets into these states, it remains in these states. Further more, assume that the probability distributions of outputs generated in failure states is different from that in non-failure states. Such systems are monitorable with respect to properties that hold only on computations without failure states.

Assume that H is monitorable with respect to $\mathcal{P}$. Now, we address the problem of constructing accurate monitors for it. Here we choose some probability threshold value z. After each output symbol generated by H, if α is the output sequence generated thus far, we compute

$BadProb(\alpha)$ and reject (i.e., raise an alarm) if $BadProb(\alpha) \geq z$. It can be shown that by choosing arbitrary high values for z, we can obtain monitors whose accuracies approach 1. This method can be implemented using product construction and state estimation as given below.

3.1.3.1 Monitoring Safety Properties

Assume that the system is modeled using a probabilistic hybrid automaton $\mathcal{A}$ and a property is specified by a safety automaton $\mathcal{P}$. We construct the product of $\mathcal{A}$ and $\mathcal{P}$ to obtain the product automaton $\mathcal{A} \times \mathcal{P}$. As the system executes, after each output, generated by the actual system, we estimate the probability that the system execution is bad. This is estimated to be the probability that the component denoting the state of $\mathcal{P}$ is an error state in the product automaton, given that it has generated the observed output sequence. If this estimated value is $\geq z$ then we reject.

In order to estimate the probability that the property automaton $\mathcal{P}$ enters the bad state we use belief (probability distribution over the states of the product automaton) propagation. At each time step, the belief is propagated from the current state to the next state, given the new observation [26]. A similar approach has been used in [23]. Particle filters were developed as a computationally efficient approximation of the belief propagation [80–82]. They have been successfully applied in the hybrid system community for state estimation [19, 20, 24, 83]. These methods become impractical for realistic systems with high number of states and several improvements have been suggested in recent years that make them practical for monitoring of stochastic cyber-physical systems.

3.1.3.2 Monitoring Liveness Properties

Monitoring of properties specified by liveness automata can be achieved using the methods given in [33, 34]. Let $\mathcal{P}$ be a Büchi automaton (The construction can be easily extended to an arbitrary Streett automaton.). We convert $\mathcal{P}$ into a safety automaton $\mathcal{P}'$ by using timeouts. Let T' be positive time out value. $\mathcal{P}$ is modified so that if an accepting state is not reached within T' units from the start or from the last time an accepting state is reached, then the automaton goes to the error state. It is fairly easy to show that any input sequence that is rejected by $\mathcal{P}$ is also rejected by $\mathcal{P}'$; however $\mathcal{P}'$ rejects more input sequences. Thus, $\mathcal{P}'$ is an approximation of $\mathcal{P}$. Note that we get better approximations by choosing larger values of T'. The above construction can be incorporated by including a *counter* variable in the QHA model. The details are straightforward.

CHAPTER 4

HIERARCHICAL PARTICLE FILTERS FOR RUNTIME MONITORING OF CONCURRENT CYBER-PHYSICAL SYSTEMS

4.1 Introduction

(Parts of) this chapter was previously published as Sistla, A. Prasad and Žefran, Miloš and Feng, Yao, "Runtime Monitoring of Stochastic Cyber-Physical Systems with Hybrid State", Runtime Verification (Springer Berlin Heidelberg, 2012), pp. 276--293 and Sistla, A. Prasad and Žefran, Miloš and Feng, Yao, "Monitorability of stochastic dynamical systems", in Proceedings of the 23rd international conference on Computer aided verification (Berlin, Heidelberg: Springer-Verlag, 2011), pp. 720--736.

In [12], we presented a runtime monitoring technique where a monitor observes the inputs and outputs of the system and checks whether the execution of the system is consistent with the expected behavior (1). The CPS is modeled as a probabilistic hybrid automaton (PHA) $\mathcal{A}$ [25,52], referred to as the *system automaton*. The desired behavior of the system is specified by a Linear Temporal Logic (LTL) formula; the LTL formula can be translated to a deterministic Streett automaton $\mathcal{P}$ [11], referred to as the *property automaton*. In [12], a threshold based monitor is proposed that, given a (finite) sequence of outputs α, evaluates the *rejection probability*, $RejProb(\alpha)$, and rejects the system execution (i.e., raises an alarm) if $RejProb(\alpha) \geq z$, where z is a chosen threshold. The rejection probability $RejProb(\alpha)$ is the conditional probability

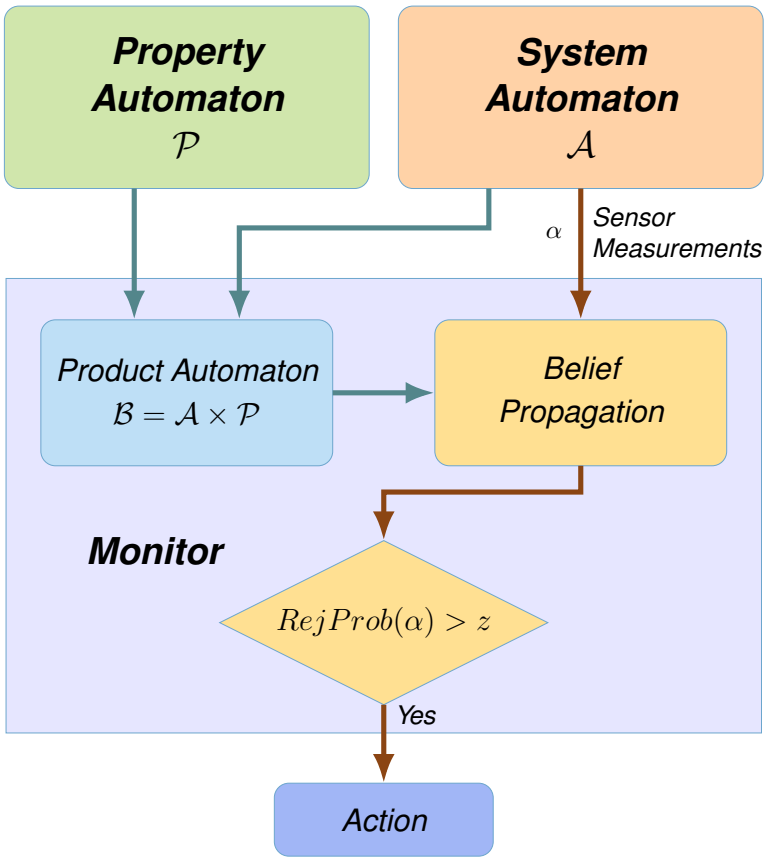

Figure 1: Runtime Monitoring Framework for CPS

that an execution of the system that generated the output sequence α violates the correctness specification. In the case of safety properties [9,10], *fault modes* can be defined as a subset of discrete states of the property automaton where the property is violated. In this case, the lower bound on the $RejProb(\alpha)$ is the probability $P_{fault}(\alpha)$ that a fault mode is reached [71]. Failure to accurately estimate $RejProb(\alpha)$ results in *missed alarms* and *false alarms*; [71] introduces the related concepts of *acceptance accuracy* and *rejection accuracy*.

Formally, the computation of $P_{fault}(\alpha)$ can be formulated as belief propagation on the product automaton $\mathcal{B} = \mathcal{A} \times \mathcal{P}$, which is again a probabilistic hybrid automaton (PHA). Computation of the belief for a PHA is thus the main technical challenge for runtime monitoring of CPSs. In [12], particle filters (PFs) [84] were employed to implement the threshold based monitor. PFs are well suited for this task since they can deal with high dimensional and hybrid state space, nondeterministic state-dependent transitions and nonlinearities.

PFs utilize a set of samples (particles) to approximate the belief – the probability distribution of state variables given a sequence of observed outputs. Thus, an important factor that affects the performance of a PF is the size of the particle set (number of particles). A *hybrid state* in PHA is comprised of a discrete component (state) – an integer that corresponds to different modes of the system, and a continuous component (state) – a vector of reals that describes the physical state. To obtain a good estimate of the state of a PHA, the number of particles should be at least large enough to cover all the possible transitions between the discrete states. As the particle filter recursively propagates the belief, if a critical transition between two modes or a possible trajectory of the continuous state is not captured by any of the particles, *particle propagation depletion [58,85]* occurs. Particle propagation depletion is a serious problem because it can lead to the failure of the particle filter to converge.

Over the past several decades, many researchers have investigated modifications of PF algorithms in order to improve the estimation performance without increasing the number of particles. Interacting Multiple Model (IMM) particle filter employs exact theoretical Bayesian equations for both accuracy and efficiency improvement [24]. Rao-Blackwellised Particle Filter

(RBPF) combines particle filtering with Kalman filter, an optimal filter for finite-dimensional linear systems to achieve better accuracy [55,58,59]. And Gaussian Particle Filter (GPF) looks one-step ahead to make the algorithm computationally efficient [56].

Although the above modifications of PF algorithms were proven to be effective in improving the estimation performance, in certain applications they do not address the particle propagation depletion problem. Risk sensitive particle filters (RSPF) [15,86,87] focus on mitigating particle propagation depletion due to low probability of discrete transitions. However, particle propagation depletion still presents a considerable challenge for hybrid systems, where phenomena other than low transition probability can cause it as discussed later in the thesis.

One important class of CPSs are systems where multiple components are executing *concurrently* and are interacting with each other during the execution. In such *concurrent* CPSs, the state space of the combined hybrid state is the product of the hybrid state spaces of the individual components. Thus, the cardinality of the product state space, in particular the number of discrete states, grows exponentially with the number of components. When computing $P_{fault}(\alpha)$ during the process of the monitoring of such concurrent CPSs, particle propagation depletion is a major concern.

In this chapter, we formally define a *concurrent* cyber-physical system with N components and propose the notion of a *concurrent* probabilistic hybrid automaton (cPHA) to model it. We introduce the concepts of *link variable* and *local variable* that allow us to characterize different types of concurrent operation of system components. An algorithm is provided to identify a special type of concurrent operation called *locally parallel operation*, when the particle filter

needs to operate on the complete set of discrete states – that can be exponential in the number of components. Finally, we propose a novel particle filtering approach, a *Hierarchical Particle Filter*, that exploits different types of concurrency to avoid particle propagation depletion while maintaining high accuracy of belief approximation.

This chapter is organized as follows. Section 4.2 introduces a modeling framework for concurrent CPSs, discusses safety specifications and describes monitoring. Link and local variables are introduced and different types of concurrent operation are defined. The generic PF algorithm and some existing modified algorithms (Risk-Sensitive PF and Rao-Blackwellised PF) are reviewed in Section 4.3.1. In Section 4.4, Hierarchical PF algorithm is proposed; this Section is the main contribution of the thesis. Finally, Section 4.5 demonstrates the effectiveness of the algorithm using the example of a train with electronically controlled brakes.

4.2 Problem Formulation

4.2.1 System Automata

In this work, we focus on stochastic CPS. A number of models were proposed for stochastic hybrid system and introduced different types of randomness. Stochastic hybrid systems proposed by Hu [40] introduced stochastic differential equations (SDE) in the evolution of continuous state, and the discrete transitions are deterministically specified by a guard function on the continuous state. A piecewise deterministic Markov Process [5] has deterministic continuous state evolution in each discrete state, but assigns a stochastic transition frequency for transitions between discrete states. Switching diffusion processes [41] combine SDE and controlled Markov Chain to capture the randomness both in continuous and discrete states. Several review papers [6, 7, 42]

list and compare a variety types of stochastic hybrid system models. In this thesis, based on the model proposed by Hu [40], we define a model of stochastic hybrid systems which include uncertainty/randomness in the continuous state evolution, the timing and destination of discrete mode transitions.

We start by formally defining Probabilistic Hybrid Automata, the modeling framework we will use.

4.2.1.1 Probabilistic Hybrid Automata with External Variables

A probabilistic hybrid automaton with external variables (PHAe) $\mathcal{A}$ is a tuple $(Q, V, \Delta t, \mathcal{E}, \mathcal{T}, c^0)$ where Q is a countable set of *discrete* states (modes) described with Q-valued discrete state variable $\mathbf{q}$; V consists of four disjoint sets of real-valued variables: a set $\mathbf{x} = \left\{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{n_x}\right\}$ of n_x continuous state variables, a set $\mathbf{u} = \left\{\boldsymbol{u}^1, \boldsymbol{u}^2, \ldots, \boldsymbol{u}^{n_u}\right\}$ of n_u external variables, a set $\mathbf{y} = \left\{\boldsymbol{y}^1, \boldsymbol{y}^2, \ldots, \boldsymbol{y}^{n_y}\right\}$ of n_y output variables and a set $\mathbf{w} = \left\{\boldsymbol{w}^1, \ldots, \boldsymbol{w}^{n_x}, \boldsymbol{w}^{n_x+1}, \ldots, \boldsymbol{w}^{n_x+n_y}\right\}$ of $n_x + n_y$ noise variables, where n_x, n_u and n_y are positive integers; Δt is the sampling time; $\mathcal{E}$ is a function that with each $q \in Q$ associates a set $\mathcal{E}(q)$ of discrete-time state equations describing the evolution of the continuous state, i.e., the value of $\mathbf{x}$ at time $t + \Delta t$ and output, i.e., the value of $\mathbf{y}$ at time t as functions of the value of the state variables in $\mathbf{x}$, the value of the external variables in $\mathbf{u}$ and the value of the noise variables in $\mathbf{w}$ at time t; $\mathcal{T}$ is a function that assigns to each $q \in Q$ a set of *transitions* $\mathcal{T}(q) = \{(\phi, p)_{q,\lambda}\}_{\lambda \in J_q}$, where $J_q \subset \mathbb{N}$ is an index set, the *guard* $\phi_{q,\lambda}$ is a measurable predicate over the set of continuous (and possibly discrete) state variables parameterized by the value of $\mathbf{u}$ and $p_{q,\lambda}$ is a probability distribution over Q, again parameterized by the value of $\mathbf{u}$; and $c^0 = (q^0, \mu_{q^0})$ is a pair giving the initial discrete state

q^0 and an initial continuous probability density function μ_{q^0} of the continuous state variables $\mathbf{x}$. We require that for each $q \in Q$, the state equations in $\mathcal{E}(q)$ have noise variables on the right hand side and that the set of guards on the transitions in $\mathcal{T}(q)$ be mutually exclusive (if $\phi_{q,\lambda_0}(x,u) = 1$ for some $\lambda_0 \in J_q$, then $\phi_{q,\lambda}(x,u) = 0$ for all other $\lambda \neq \lambda_0$).

We will assume that all variables in V (continuous state, external, output and noise, respectively) are ordered so that we can consider their values as a vector.

To formally define the semantics of a PHAe $\mathcal{A}$ we would need to introduce the concept of Extended Hidden Markov Chain [12]. We omit the details in the interest of space and refer the reader to [88]. Intuitively, within each mode q, the evolution of the PHAe is given by the difference equations $\mathcal{E}(q)$. When a guard $\phi_{q,\lambda}$ of a transition $(\phi, p)_{q,\lambda} \in \mathcal{T}(q)$ becomes satisfied, a transition takes place from q to target mode q' according to the probability distribution $p_{q,\lambda}$.

A Probabilistic Hybrid Automaton (PHA) is a special class of PHAe without external inputs, $\mathbf{u} = \emptyset$.

While the evolution of a PHAe can be quite general, we will assume that it is governed by the following equations:

$$q_{t+1} = g(q_t, x_t, u_t) \tag{4.1}$$

$$x_{t+1} = f_{q_{t+1}}(x_t, u_t) + w_{x,t} \tag{4.2}$$

$$y_t = h_{q_t}(x_t, u_t) + w_{y,t} \tag{4.3}$$

where $q_t \in Q$, $x_t \in \mathbb{R}^{n_x}$, $u_t \in \mathbb{R}^{n_u}$ and $y_t \in \mathbb{R}^{n_y}$, are discrete state (value of the discrete state variable $\mathbf{q}$), continuous state (vector value of the continuous state variables $\mathbf{x}$), external input (vector value of the external variables $\mathbf{u}$), and output (vector value of the output variables $\mathbf{y}$), respectively, at time t; $w_{x,t} \in \mathbb{R}^{n_x}$ and $w_{y,t} \in \mathbb{R}^{n_y}$ are vector values of the process noise variable $\mathbf{w}_x$ and the measurement noise variable $\mathbf{w}_y$ which partition noise variables $\mathbf{w}$, at time t, where $\{w_{x,t}\}$ and $\{w_{y,t}\}$ are generated by sequences of independent identically-distributed (i.i.d.) random variables that are independent of the initial condition; the mappings f_q and h_q are time-invariant and continuous in x and u for any $q \in Q$; and $g(q_t, x_t, u_t)$ is a random variable over Q whose probability distribution is governed by $p_{q_t,\lambda}$ provided that the predicate $\phi_{q_t,\lambda}$ evaluated for (x_t, u_t) is satisfied, or is an identity on q_t otherwise.

A hybrid state of a PHAe at time t , denoted by s_t, is a pair consisting of the continuous state and the discrete state, $s_t = (q_t, x_t)$.

4.2.1.2 Concurrent Dynamical Systems

Systems that implement complex tasks can often be decomposed into several components during the design phase. This strategy provides flexibility in system verification and failure analysis as well. Each component can be viewed as a *subsystem* implementing a function/procedure for the overall system. The overall system is a concurrent system that is the composition of the interacting subsystems.

Assume we have an autonomous concurrent dynamical system $\mathcal{C}$ which is comprised of N subsystems $\{\mathcal{C}_i, i = 1, \ldots, N\}$. For simplicity, assume each $\mathcal{C}_i$ is described by a linear state space model [89] which we will denote by a subscript i,

$$x_{i,t+1} = A_i x_{i,t} + B_i u_{i,t}$$

$$y_{i,t} = C_i x_{i,t}$$

where $x_{i,t}$ is the vector value of state variables $\mathbf{x}_i = \left\{\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^{n_i}\right\}$, $u_i(t)$ is the vector value of input variables $\mathbf{u}_i = \left\{\boldsymbol{u}_i^1, \boldsymbol{u}_i^2, \ldots, \boldsymbol{u}_i^{p_i}\right\}$, and $y_{i,t}$ is the vector value of output variable $\mathbf{y}_i$, at time t; A_i, B_i and C_i are the state matrix, input matrix and output matrix, respectively.

The interaction among subsystems is specified by

$$u = Sx, \tag{4.4}$$

where $u = (u_1, u_2, \ldots, u_N)$ is the joint input vector, $x = (x_1, x_2, \ldots, x_N)$ is the joint state vector, and S is a real-valued constant matrix of dimension $\sum_{i=1}^{N} n_{p_i} \times \sum_{i=1}^{N} n_{x_i}$.Observe that S has zero matrices of dimension $n_{u_i} \times n_{x_i}$, $i = 1, \ldots, N$ on the diagonal since the system is never connected to itself (possible feedback loops are already part of the subsystem model). The linear interaction model defines how the external input variables of the subsystem $\mathcal{C}_i$ are connected to the state variables of the other subsystems.

The state space model for the resulting concurrent linear dynamical system $\mathcal{C}$ with joint state variables $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ and joint output variable $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$ is the combination of the models of each $\mathcal{C}_i$ and the interaction model described by (4.4). This results in

$$x_{t+1} = Ax_t$$

$$y_t = Cx_t$$

where A is derived by substituting $\{u_{i,t},\ i = 1, \ldots, N\}$ with the linear transformation of the joint state x_t as described by (4.4), and

$$C = \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_N \end{bmatrix}.$$

The input term $B_i u_{i,t}$ of each subsystem is also incorporated into the state matrix A through the interaction model.

The dependency of subsystem evolution determines the type of concurrency among subsystems.

Definition 1 (Input decoupled subsystem) *A subsystem $\mathcal{C}_i$ is input decoupled from the overall system $\mathcal{C}$ if it is an autonomous system, $\mathbf{u}_i = \emptyset$.*

The simplest type of concurrency among subsystems is *parallel operation*. In this case, $\mathbf{u}_i = \emptyset$, $i = 1, \ldots, N$. Equivalently, each subsystem is an input decoupled system and there is no interaction among subsystems.

Then, the state matrix A for the concurrent linear dynamical system $\mathcal{C}$ has the diagonal form

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_N \end{bmatrix}$$

The opposite of the parallel operation is a *fully coupled* operation. In this case, none of the subsystems $\mathcal{C}_i$ can be input decoupled from the overall system $\mathcal{C}$.

If there exist an index set $\Xi \subset \{1, 2, \ldots, N\}$, such that for every $i \in \Xi$, A_{ij} is a zero matrix for all $j \neq i$, then subsystems $\{\mathcal{C}_i, i \in \Xi\}$ are input decoupled from the overall $\mathcal{C}$. Then the concurrent subsystem evolution at each time step can be decomposed into a parallel evolution of subsystems $\{\mathcal{C}_i, i \in \Xi\}$ followed by a coupled evolution of the remaining subsystems of $\mathcal{C}$.

Remark 1 *The types of concurrency in subsystem operation for a linear dynamical system can be extended to a concurrent non-linear dynamical system.*

4.2.1.3 Modeling of Concurrent CPSs

A concurrent CPS can be modeled as the composition of N interacting subsystems. Each subsystem is modeled as a PHAe, where the external variables of a subsystem correspond to the continuous state variables of other subsystems. Thus, we model a concurrent CPS by a

concurrent probabilistic hybrid automaton (cPHA). To distinguish different subsystems, we will use subscript i to indicate the i subsystem. Its model is denoted by $\mathcal{A}_i = (Q_i, V_i, \Delta t_i, \mathcal{E}_i, \mathcal{T}_i, c_i^0)$, where $V_i = \mathbf{x}_i \cup \mathbf{u}_i \cup \mathbf{y}_i \cup \mathbf{w}_i$.

An autonomous concurrent probabilistic hybrid automaton (cPHA) $\mathcal{A}$ is a tuple $\mathcal{A} = (\widehat{\mathcal{A}}, S)$ where $\widehat{\mathcal{A}} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N\}$ denotes the set of PHAe-s representing N subsystems, and S is a real-valued constant matrix of dimension $\sum_{i=1}^{N} n_{u_i} \times \sum_{i=1}^{N} n_{x_i}$ that models how subsystems interact

$$u = Sx \tag{4.5}$$

where $u = (u_1, u_2, \ldots, u_N)$ is the joint input vector (vector value of the joint input variable $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N)$) and $x = (x_1, x_2, \ldots, x_N)$ is the joint state vector (vector value of the joint continuous state variables $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$). The linear interaction model defines how the external variables of the subsystem $\mathcal{A}_i$ are connected to the continuous state variables of the other subsystems.

Similar to a concurrent dynamical system, S has zero matrices of dimension $n_{u_i} \times n_{x_i}$, $i = 1, \ldots, N$ on the diagonal since the system is never connected to itself (possible feedback loops are already part of the subsystem model).

The dependency of the subsystem evolution specified by the interaction model determines the type of concurrency in the subsystem operation. However, in a cPHA, the concurrency of operation is linked to a discrete mode.

We first define a subsystem $\mathcal{A}_i$ without input dependency in $q_i \in Q_i$.

Definition 2 (Input decoupled subsystem in a cPHA) *A subsystem $\mathcal{A}_i$ is input decoupled from the overall cPHA $\mathcal{A}$ if,*

1. $\mathbf{u}_i = \emptyset$, *or*
2. *the evolution of the discrete mode and the continuous state does not depend on u_i.*

Compared with a concurrent dynamical system, condition 2 is an additional case that subsystem $\mathcal{A}_i$ of a cPHA is input decoupled. It refers to a situation in which the evolution of the discrete mode and the continuous state depends on u_i in discrete mode q_i', but does not depend on u_i in another discrete mode q_i''.

If all the subsystems are input decoupled, the evolution for each subsystem is fully decoupled. In other words, the type of concurrency among subsystems is a *parallel operation.*

If none of the subsystems is input decoupled, the subsystem evolution is fully coupled [90].

However, if there exist an index set $\Xi \subset \{1, 2, \ldots, N\}$, such that for every $i \in \Xi$, $\mathcal{A}_i$ is input decoupled, then the concurrent subsystem evolution at each time step can be decomposed into a parallel evolution of subsystems $\{\mathcal{A}_i,\, i \in \Xi\}$ followed by a coupled evolution of the remaining subsystems of $\mathcal{A}$.

By substituting the input variables $\{u_i,\, i = 1, \ldots, N\}$ with the state variables using the interaction model (4.5), a concurrent probabilistic hybrid automaton $\mathcal{A}$ can be viewed as one monolithic PHA.

Specifically, a concurrent probabilistic hybrid automaton (cPHA) $\mathcal{A}$ can be seen as a PHA $(Q, V, \Delta t, \mathcal{E}, \mathcal{T}, c^0)$, where $Q = \prod_{i=1}^{N} Q_i$, the Cartesian product of discrete state space of all subsystems which is associated with the discrete joint variable $\mathbf{q} = (\mathbf{q}_1, \ldots, \mathbf{q}_N)$; V consists of three

disjoint sets of variables: a set $\mathbf{x} = \bigcup_{i=1}^{N} \mathbf{x}_i$ of continuous state variables, a set $\mathbf{y} = \bigcup_{i=1}^{N} \mathbf{y}_i$ of output variables and a set $\mathbf{w} = \bigcup_{i=1}^{N} \mathbf{w}_i$ of noise variables, that when ordered as $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$, $\mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$ and $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N)$, take values in $\mathbb{R}^{\sum_i n_{x,i}}$, $\mathbb{R}^{\sum_i n_{y,i}}$, $\mathbb{R}^{\sum_i n_{x,i} + \sum_i n_{y,i}}$, respectively, where $n_{x,i}$ and $n_{y,i}$ are positive integers representing the cardinality (dimension) of $\mathbf{x}_i$ and $\mathbf{y}_i$ of subsystem $\mathcal{A}_i$; Δt is the sampling time; $\mathcal{E}$ is a function that with each $q \in Q$ associates a set $\mathcal{E}(q)$ of discrete-time state equations describing the evolution of the continuous state, i.e., value of $\mathbf{x}$ and output, i.e., value of $\mathbf{y}$) at time $t+\Delta t$ as functions of the state at t, and the value of the noise variable $\mathbf{w}$; $\mathcal{T}$ is a function that assigns to each $q \in Q$ a set of *transitions* $\mathcal{T}(q) = \{(\phi, p)_{q,\lambda}\}_{\lambda \in J_q}$, where $J_q \subset \mathbb{N}$ is an index set, the *guard* $\phi_{q,\lambda}$ is a measurable predicate over the set of continuous (and possibly discrete) state variables and $p_{q,\lambda}$ is a probability distribution over Q; and $c^0 = (q^0, \mu_{q^0})$ is a pair giving the initial joint discrete state $q^0 = (q_1^0, \ldots, q_N^0)$ and an initial continuous probability density function $\mu_{q^0} = \prod_{i=1}^{N} \mu_{q_i^0}$ of the continuous variable $\mathbf{x}$.

When a cPHA is in discrete state $q = (q_1, \ldots, q_N)$, the discrete-time state equations $\mathcal{E}(q)$ describing the evolution of joint continuous state (value of $\mathbf{x} = \bigcup_{i=1}^{N} \mathbf{x}_i$) are the combination of $\{\mathcal{E}_i(q_i),\ i = 1, \ldots, N\}$ and the model of interaction. Specifically, the state equations $\mathcal{E}(q)$ are derived by substituting $u_{i,t}$ with the corresponding transformation of of x_t described by S. Please note that $\mathcal{E}(q), q \in Q$ is an empty set when q is not reachable.

Transitions $\mathcal{T}(q)$ are extended from $\{\mathcal{T}_i(q_i), i = 1, \ldots, N\}$. Basically, each transition in $\{\mathcal{T}_i(q_i), i = 1, \ldots, N\}$ where $q_i \in Q_i$ is a transition of the overall cPHA. For example, if the system is in state $q' = (q'_1, \ldots, q'_i, \ldots, q'_N)$, and a transition with guard $\phi_{q'_i,\lambda_i}(x_i, u_i)$ in subsystem

$\mathcal{A}_i$ takes place from mode q'_i to q''_i with distribution $p_{q'_i,\lambda_i}$, then it's a transition of the cPHA with guard $\phi_{q',\lambda}(x)$ which is obtained from $\phi_{q'_i,\lambda_i}(x_i,u_i)$ by substituting u_i with the corresponding transformation of of x_t described by S, and the target state is $q'' = (q'_1 \ldots, q''_i, \ldots, q'_N)$ with distribution $p_{q',\lambda=p_{q'_i,\lambda_i}}$ over the joint discrete state space $Q' = \{q'_1\} \times \ldots \times \left\{q'_{i-1}\right\} \times Q_i \times \left\{q'_{i+1}\right\} \times \ldots \times \{q'_N\}$ (i.e., the discrete states of the other subsystems remain the same). If in state q', the guard $\phi_{q'_i,\lambda_i}(x_i,u_i)$ of $\mathcal{A}_i$ and $\phi_{q'_j,\lambda_j}(x_j,u_j)$ of $\mathcal{A}_j$ (assuming $i<j$) are essentially the same after substituting u_i and u_j with the corresponding transformations of of x_t described by S, then the target state is $q'' = (q'_1 \ldots, q''_i, q''_j, \ldots, q'_N)$ with distribution $p_{q',\lambda=p_{q'_i,\lambda_i}} \times p_{q'_j,\lambda_j}$ over the joint discrete state space $Q' = \{q'_1\} \times \ldots \times \left\{q'_{i-1}\right\} \times Q_i \times \left\{q'_{i+1}\right\} \times \ldots \times Q_j \times \ldots \times \{q'_N\}$.

Reachable set of joint discrete states, Q_R can be recursively built from its definition.

Definition 3 (Reachable Set of Joint Discrete States) *Formally, the reachable set of joint discrete states $Q_R \subseteq Q$ is the smallest subset of Q such that*

- $q^0 = (q^0_1, \ldots, q^0_N) \in Q_R$
- *For every $q' \in Q_R$ and every $\lambda \in J_{q'}$, if $p_{q',\lambda}(q'') > 0$, then $q'' \in Q_R$.*

4.2.1.3.1 Link Variables and Local Variables

While the notion of external input variables is merged into the continuous state variables in cPHA, the continuous state variables that are used to characterize the interaction are of interest to further investigate the concurrency in subsystem operation. We refer to these variables as *link variables*. Formally,

Definition 4 (Link variables) *For subsystem $\mathcal{A}_i$, a continuous variable $\boldsymbol{x}_i^k \in \mathbf{x}_i$, $k \in \{1, \ldots, n_{x_i}\}$ is a link variable if its value is shared with other subsystems.*

Equivalently, $\boldsymbol{x}_i^k \in \mathbf{x}_i$, $k \in \{1, \ldots, n_{x_i}\}$is a link variable if the column $(\sum_{j=1}^{i-1} n_{x_j} + k)$ in S has non-zero entry.

The set of link variables in $\mathcal{A}_i$ is denoted as $\hat{\mathbf{x}}_i$, and the set of link variables of overall system is defined as $\mathbf{z} = \bigcup_{i=1}^{N} \hat{\mathbf{x}}_i$.

Definition 5 (Local variables) *For subsystem $\mathcal{A}_i$, a continuous variable $\boldsymbol{x}_i^k \in \mathbf{x}_i$ is a local variable if*

$$\boldsymbol{x}_i^k \notin \mathbf{z}.$$

Equivalently, $\boldsymbol{x}_i^k \in \mathbf{x}_i$, $k \in \{1, \ldots, n_{x_i}\}$is a local variable if the column $(\sum_{j=1}^{i-1} n_{x_j} + k)$ in S is zero.

The set of local variables of subsystem $\mathcal{A}_i$ is denoted as $\mathbf{l}_i$. Intuitively, local variables $\{\mathbf{l}_i,\, i = 1, \ldots, N\}$ are the internal state variables that are not "visible" from other subsystems. In this way, we partition the state variables of the overall system, $\bigcup_{i=1}^{N} \mathbf{x}_i$, into $N+1$ pieces, N sets of local variables $\{\mathbf{l}_i,\, i = 1, \ldots, N\}$ for N subsystems, and a set of link variables $\mathbf{z}$.

By regrouping the continuous state variables in cPHA as link and local variables, the evolution of cPHA can be described by the following equations:

$$q_{t+1} = g(q_t, l_{1,t}, \ldots, l_{i,t}, \ldots, l_{N,t}, z_t) \tag{4.6}$$

$$l_{i,t+1} = \tilde{f}_{q_{t+1},i}(l_{i,t}, z_t) + w_{i,t},\ i = 1, \ldots, N \tag{4.7}$$

$$z_{t+1} = r_{q_{t+1}}(l_{1,t}, \ldots, l_{i,t}, \ldots, l_{N,t}, z_t) + w_{z,t} \tag{4.8}$$

$$y_t = h_{q_t}(l_{1,t}, \ldots, l_{i,t}, \ldots, l_{N,t}, z_t) + w_{y,t} \tag{4.9}$$

where $q_t = (q_{1,t}, \ldots, q_{N,t}) \in \prod_{i=1}^{N} Q_i,\ l_{i,t} \in \mathbb{R}^{n_{l_i}}\ z_t \in \mathbb{R}^{n_z},\ y_t \in \mathbb{R}^{n_y}$ represent joint discrete state, local states, link states, and outputs at time t, with dimensions n_{l_i}, n_z and n_y, respectively; $w_{i,t} \in \mathbb{R}^{n_{l_i}}$, $w_{z,t} \in \mathbb{R}^{n_z}$ $w_{y,t} \in \mathbb{R}^{n_y}$ represent vector values of process noise variables $\mathbf{w}_i$ and $\mathbf{w}_z$ for local variables, link variables and measurement noise variable $\mathbf{w}_y$ which partition noise variable $\mathbf{w}$, at time t, where $\{w_{i,t}\}$, $\{w_{z,t}\}$, and $\{w_{y,t}\}$ are generated by sequences of i.i.d. random variables that are independent of the initial condition; the mappings $\tilde{f}_{q,i}$, r_q, h_q are time-invariant and continuous in l_i and z for any $q \in Q$; and $g(q_t, l_{1,t}, \ldots, l_{i,t}, \ldots, l_{N,t}, z_t)$ is a random variable over Q whose probability distribution is governed by $p_{q_t,\lambda}$ provided that the predicate $\phi_{q_t,\lambda}$ evaluated for $(l_{1,t}, \ldots, l_{i,t}, \ldots, l_{N,t}, z_t)$ is satisfied, or identity on q_t otherwise.

It is not difficult to derive functions $\tilde{f}_{q_{t+1},i}$ and $r_{q_{t+1}}$ from the corresponding $f_{q_{t+1}}$in (4.2) describing the individual subsystems modeled by PHAe-s.

Using link and local variables, a hybrid state of cPHA at time t becomes a tuple $s_t = (q_t, l_{1,t}, \ldots, l_{N,t}, z_t)$ of discrete state, local states and link states.

4.2.1.4 Concurrency in Subsystem Operation in a cPHA

In the context of cPHA, concurrency analysis has to be performed for each discrete mode. Also by using the notions of local variables $\{\mathbf{l}_i,\ i = 1, \ldots, N\}$ and link variables $\mathbf{z}$, we can further exploit the system structure. To start with, it will be useful to take all the link variables and consider them as an additional separate subsystem; this will allow us to possibly identify subsets of subsystem variables that can be input decoupled even though the subsystem themselves are not. In particular, in the joint discrete mode q, for a subsystem $\mathcal{A}_i$ we separately consider the local variables $\mathbf{l}_i$ and determine whether they are input decoupled, the link variables in $\mathbf{x}_i$ are considered separately.

Moreover, the guard of discrete mode switch depends on the continuous state variables. For subsystem $\mathcal{A}_i$ to be considered as input decoupled from the overall cPHA in joint discrete mode $q' = (q'_1, \ldots, q'_N)$, not only the evolution of local state l_i , but also every guard ϕ_{q_i,λ_i}, $\lambda_i \in J_{q_i}$ should not depend on link variables.

Definition 6 (Linking Transition) *For subsystem $\mathcal{A}_i$, a transition $\mathcal{T}_i(q_i) = \{(\phi, p)_{q_i,\lambda_i}\}_{\lambda_i \in J_{q_i}}, q_i \in Q_i$, is a* linking transition *if the transition guard ϕ_{q_i,λ_i} depends on any link variable in $\mathbf{z}$. The corresponding transition in $\mathcal{T}(q)$ is also a linking transition.*

Definition 7 (Locally Input Decoupled subsystem of a cPHA) *For subsystem $\mathcal{A}_i$, if in discrete state $q_i^* \in Q_i$,*

1. *The evolution of the local state does not depend on the link states: $\tilde{f}_{q_i^*,i}(l_{i,t}, z_t) = \tilde{f}_{q_i^*,i}(l_{i,t})$, and*

2. When $J_{q_i^*} \neq \emptyset$, there is no $\lambda_i \in J_{q_i^*}$, such that $\mathcal{T}_i(q_i^*) = \{(\phi, p)_{q_i^*, \lambda_i}\}$ is a global transition, then $\mathcal{A}_i$ is said to be locally input decoupled *from overall system $\mathcal{A}$ in discrete state q_i^*. In turn, discrete state q_i^* is a parallel operated discrete state.*

The concurrency in the operation of the overall system is determined by the dependency of the evolution of the discrete, link and local state. A general case is that the evolution of the discrete, link and states is fully coupled so that none of the subsystem is locally input decoupled, as shown in Figure 2. If the link state does not depend on any local state, and the evolution of local state from some subsystems depends on link state, then the evolution of overall system is decomposed into the evolution of the link state followed by a parallel evolution of discrete mode and local states, as shown in Figure 3.

Figure 4 shows another case. At each time step, if each subsystem is locally input decoupled, and the evolution of link state is dependent on local state, the evolution of the overall system can be decomposed into a parallel evolution of the discrete and local states followed by a coupled evolution of link states. Furthermore, if the evolution of link state does not depend on local states, then the cPHA is in parallel operation. However, if, instead of all subsystems, there exist an index set $\Xi \subset \{1, 2, \ldots, N\}$, such that for every $i \in \Xi$, $\mathcal{A}_i$ is locally input decoupled, then the concurrent subsystem evolution at each time step can be decomposed into a parallel evolution of local states in subsystems $\{\mathcal{A}_i,\, i \in \Xi\}$ followed by a coupled evolution of the local states in the remaining subsystems of $\mathcal{A}$ and the link state, as shown in Figure 5. These two cases are referred to as *fully locally parallel operation* and *locally parallel operation*, respectively. Formally,

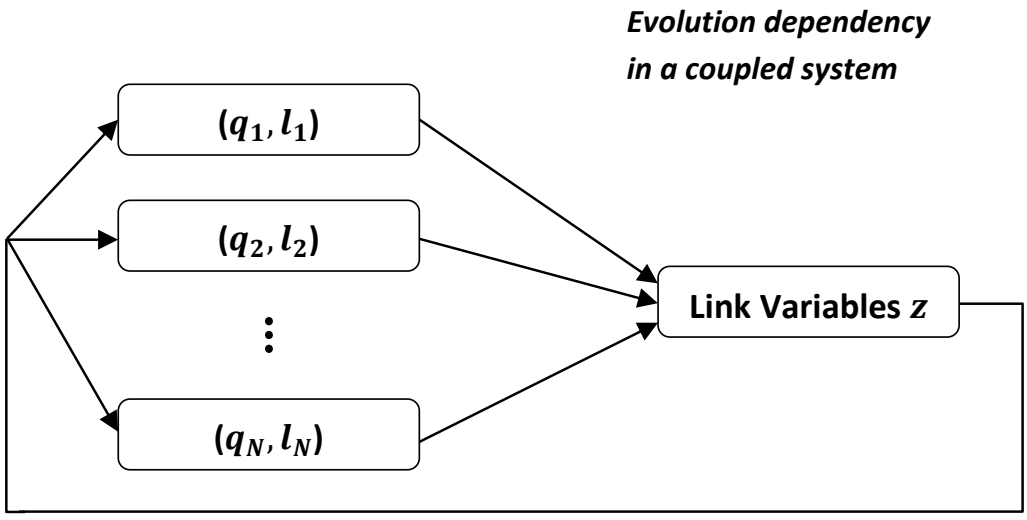

Figure 2: Evolution Dependency of a Coupled cPHA

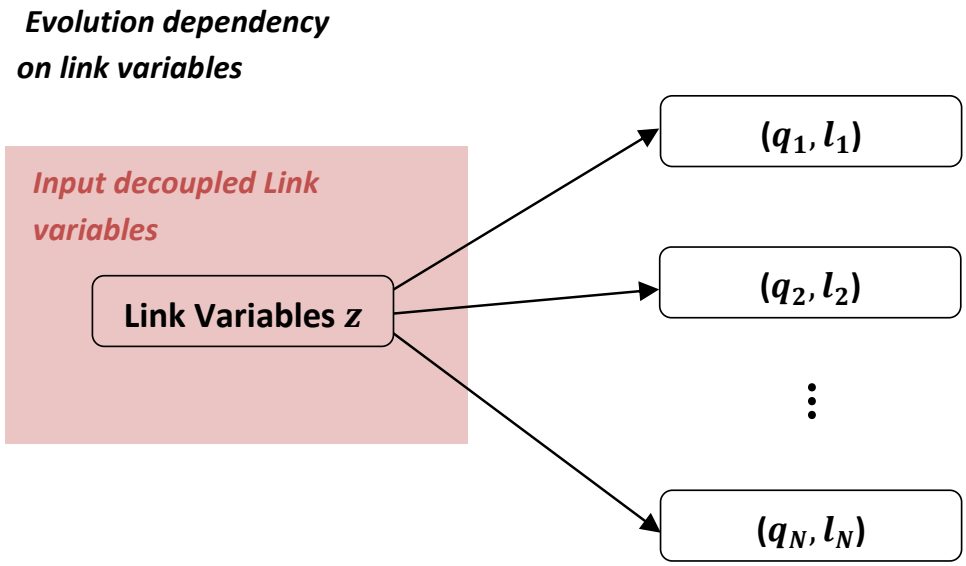

Figure 3: Evolution of the discrete state variable and the local variables depending on the link variables

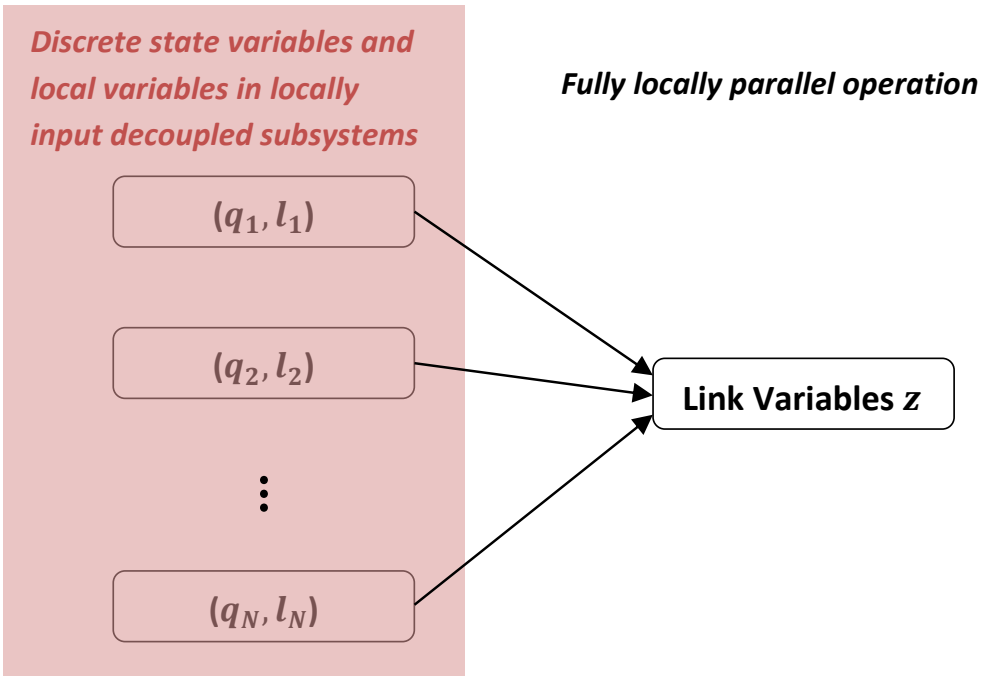

Figure 4: Fully Locally Parallel Operation

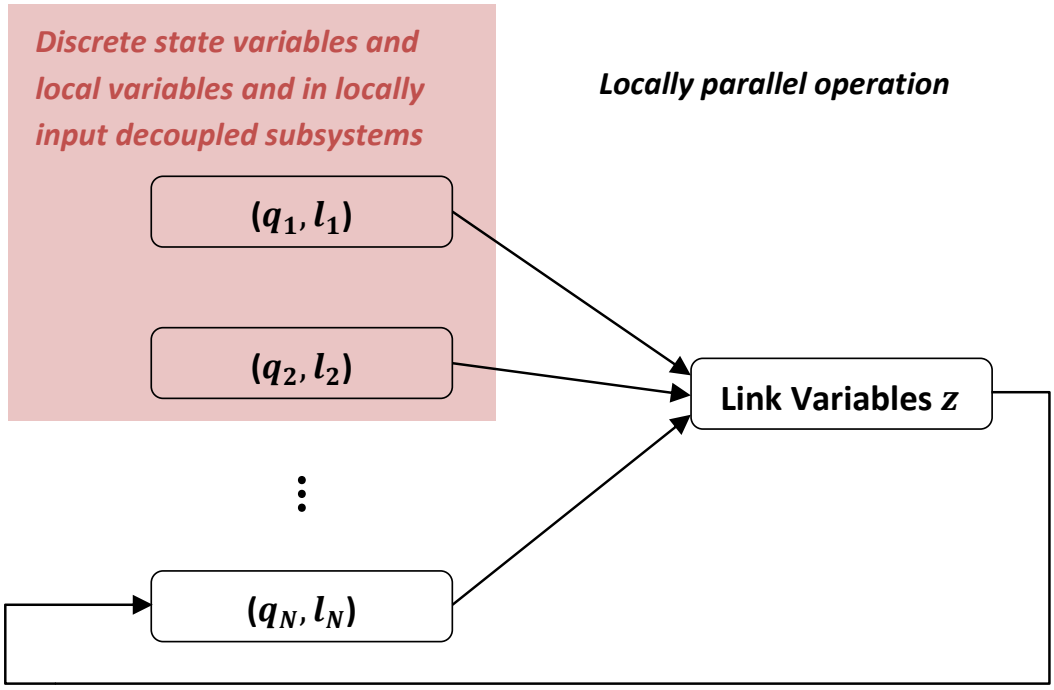

Figure 5: Locally Parallel Operation

Definition 8 (Locally Parallel Operation) *If, in joint discrete state* $q^* = (q_1^*, \ldots, q_N^*)$ *, at least two subsystems are locally input decoupled from overall system* $\mathcal{A}$*, the cPHA is said to be in* locally parallel operation. *A cPHA is said to be in* fully *locally parallel operation if all the subsystems are locally input decoupled from* $\mathcal{A}$*.*

Correspondingly, if the system is not in locally parallel operation, it is in *non-parallel operation.*

Note that by definition, the evolution of the local state l_i of subsystem i as well the local discrete state q_i can not depend on the local state of another subsystem. In addition, when a system is in fully locally parallel operation, the evolution of l_i and q_i is also independent from the link variables. This observation is fundamental to the discussion in Section 4.4.

For cPHA, the notion of concurrency is linked to each discrete mode. Therefore, the type of concurrent operation of subsystems may change when the cPHA transition occurs. By analyzing $\mathcal{E}(q)$ and $\mathcal{T}(q)$, we can determine if a subsystem $\mathcal{A}_i$ is decoupled in mode $q_i \in Q_i$. For each subsystem, $Q_{i,Parallel} \in Q_i$ is the set of discrete modes when subsystem $\mathcal{A}_i$ is locally input decoupled from the overall system $\mathcal{A}$. We give a generic algorithm for the analysis of a cPHA $\mathcal{A}$ which is a composition of N subsystems $\{\mathcal{A}_i, i = 1, \ldots N\}$.

4.2.1.5 Generic Algorithm for Subsystem Decomposition in cPHA

4.2.2 Property Automata

Correctness requirements to be monitored for safety-critical CPSs are specified on system hybrid state $s = (q_S, l_1, \ldots, l_N, z)$ of a system modeled by concurrent PHA $\mathcal{A}$. Specifically, the correctness requirements are described by Linear Temporal (LT) properties over the atomic

Algorithm 2 cPHA Concurrency Analysis $\{\mathcal{A}_i,\, i = 1, \ldots, N\}$

Step 1: Continuous state variables regrouped to $\{\mathbf{l}_1, \ldots, \mathbf{l}_N, \mathbf{z}\}$

for each $\mathcal{A}_i$ **do**

for k in $\{1, \ldots, n_{x_i}\}$ **do**

if The corresponding $(\sum_{j=1}^{i-1} n_{x_j} + k)^{th}$ column of S has any non-zero entry **then**

Identify $\boldsymbol{x}_i^k$ as a *link variable* and add $\boldsymbol{x}_i^k$ in $\mathbf{z}$

else

Identify $\boldsymbol{x}_i^k$ as a *local variable* and add $\boldsymbol{x}_i^k$ in $\mathbf{l}_i$

Step 2: Subset of discrete states in locally parallel operation, $Q_{i,Parallel}$

for each $\mathcal{A}_i$ **do**

if $\mathcal{A}_i$ is locally input decoupled in $q_i \in Q_i$ **then**

Add q_i in $Q_{i,parallel}$

propositions specified on system hybrid state s over time. We can model LT properties by a *deterministic* Streett automaton. $\mathcal{P}$.

4.2.2.1 Streett Automata

Formally, a deterministic Streett automaton $\mathcal{P}$ is a tuple $(Q_P, \Sigma_P, \delta_P, q_P^0, F)$ where Q_P is it set of states associated with discrete state variable $\mathbf{q}_P$, Σ is its input alphabet, $\delta_P : Q_P \times \Sigma_P \rightarrow Q_P$ is the next state function, q_P^0 is the starting state and F is a collection of pairs of subsets of states of the form $(RED, GREEN)$. Runs and accepting runs of the automata on an infinite string of inputs are defined naturally. An infinite input string is accepted if there is an accepting run of the automaton on the input starting from the initial state. We let $L(\mathcal{P})$ denote the set of strings accepted by $\mathcal{P}$. An automaton $\mathcal{P}$, as given above, is called a *safety automaton* [71] if F has a single pair $(\{q_{error}\}, \emptyset)$ where q_{error} is an absorbing state/mode, i.e., all transitions from it go back to itself. The state q_{error} is called the error state/mode and it is easy to see that an infinite input sequence is accepted iff the error state is never reached on this input.

Any LT property can be decomposed to be the conjunction of a safety property and a liveness property [9,10]. Safety property is modeled by a safety automaton, and liveness property is modeled by a liveness automaton. Safety automaton is a deterministic Streett automaton containing a subset of special absorbing states/modes called *error* states/modes with all the other states being the accepting states/modes. The set of sequences accepted by a safety automaton is a safety property. Similarly, a liveness automaton is a property automaton whose accepting set of sequences is a liveness property.

In a safety automaton $\mathcal{P}_s$ modeling a safety property, the *error* mode is reached if the safety property is violated. Considering a property automaton is driven by the hybrid state execution of system automaton $\mathcal{A}$. Thus, we need to construct a product automaton of $\mathcal{A}$ and the $\mathcal{P}_s$.

Liveness property is more complicated because the correctness is considered not to be violated until it is asserted that after some time, the accepting mode is never reachable [9, 10]. Strictly, it can only be verified in infinite horizon which is infeasible in experiment. So, we convert a liveness automaton $\mathcal{P}_l$ into a safety automaton using a timeout T'. If an accepting mode is not reached within T' time units, the property automaton goes to a designed *error* mode, meaning that the specification is violated. It is easy to show that the value of timeout variable T' can be increased to get a better approximation. So, in this way, the liveness property $\mathcal{P}_l$ is approximated by a safety property and can be verified using the same approach introduced in safety property monitoring.

These two types of *error* modes in safety and liveness automaton are referred to as *fault modes*. A discrete state sequence of $\mathcal{P}$ containing fault modes indicates that the hybrid state execution of system automaton $\mathcal{A}$ is a bad run, violating the correctness property.

4.2.3 Monitors

Assuming a partially observable hybrid system is *monitorable* [71] with respect to a given property, we can design a monitor for it. In [71], a threshold based monitor is proposed. For a observation sequence α, the monitor evaluates $RejProb(\alpha)$, and rejects the system execution which generated α if $RejProb(\alpha) > z$, a preset threshold probability. $RejProb(\alpha)$ is defined as the conditional probability that an execution of the system that generated the output sequence

α is rejected by the property. It can be approximated to be the probability $P_{fault}(\alpha)$ that the state run of $\mathcal{P}$ driven by the system execution which generated α reaches a *fault mode*.

Formally, the computation of $P_{fault}(\alpha)$ can be formulated as belief propagation on the product automaton $\mathcal{A} \times \mathcal{P}$, where $\mathcal{A}_S$ is partially observable probabilistic hybrid automaton describing the CPS to monitor, and $\mathcal{P}$ is a deterministic Streett automaton describing correctness requirements to be verified. Thus, the core part of the runtime monitoring of CPS is the hybrid state estimation of a PHA.

4.2.3.1 Monitorability

Monitorability of the system has to be checked before the design of a threshold based monitor. In [71], necessary and sufficient conditions are given for the *monitorability* of a system with respect to a given property. Basically, it states that if a system $\mathcal{A}$ is *monitorable*, with probability 1, given any infinite output generated by $\mathcal{A}$, the state execution can be classified as *good* or *bad* with respect to a property automaton $\mathcal{P}$.

If the system is monitorable, it is possible to find a monitor that achieves arbitrarily high levels of both acceptance accuracy and rejection accuracy. Furthermore, it has been shown in [71] that a specific monitoring algorithm, called threshold-based monitor, can be used to do so. A threshold-based monitor uses hybrid state estimation to compute $P_{fault}(\alpha)$. Monitorability is related to, but fundamentally different from observability. To start with, monitorability is defined with respect to a given property which is independent of the system model. Instead, observability is an intrinsic property of the system itself. However, at least intuitively, if the system is observable then it is monitorable. This is because the observability implies complete

knowledge of the state trajectories; these trajectories can in turn be used to evaluate the evolution of the property automaton. On the other hand, the system can be monitorable even when it is not observable. For example, if the property only depends on the observable part of the state, the system will be monitorable.

Note also that even if the system is observable, monitoring typically does not require the knowledge of the complete state so a full state estimator is not needed. This can significantly reduce the computational cost, especially when the system is complex.

4.3 Hybrid State Estimation

In the estimation problem of the product automaton $\mathcal{B} = \mathcal{A} \times \mathcal{P}$, the high dimensional hybrid state space, nonlinearity and nondeterministic state-dependent transitions, are intractable using the conventional estimation techniques. Particle Filtering (PF) is considered as an efficient tool for such a computational problem, and has been applied to many complicated, nonlinear systems in a variety of areas, including navigation [91], fault detection of complex systems [15] and visual tracking [92].

This section provides the generic particle filtering algorithm and two important modifications: Risk-Sensitive particle filtering and Rao-Blackwellized particle filtering.

4.3.1 Particle Filters

In general, a probabilistic hybrid automaton is described by the following difference equations, on $[0, T]$:

$$q_{t+1} = g(q_t, x_t)$$

$$x_{t+1} = f_{q_{t+1}}(x_t) + w_{x,t}$$

$$y_t = h_{q_t}(x_t) + w_{y,t}$$

where $q_t \in Q$, $x_t \in \mathbb{R}^{n_x}$, and $y_t \in \mathbb{R}^{n_y}$, represent discrete state (value of the discrete state variable $\mathbf{q}$), continuous state (vector value of the continuous state variables $\mathbf{x}$), and output (vector value of the output variables $\mathbf{y}$), respectively, at time t; $w_{x,t} \in \mathbb{R}^{n_x}$, $w_{y,t} \in \mathbb{R}^{n_y}$ represent the vector value of the process noise variable $\mathbf{w}_x$ and the measurement noise variable $\mathbf{w}_y$ which partition noise variable $\mathbf{w}$, at time t, where $\{w_{x,t}\}$ and $\{w_{y,t}\}$ are generated by sequences of i.i.d. random variables that are independent of the initial condition; the mappings f_q and h_q are continuous in x for every $q \in Q$; and $g(q_t, x_t)$ is a random variable over Q whose probability distribution is governed by $p_{q_t,\lambda}$ provided that the predicate $\phi_{q_t,\lambda}$ evaluated for x_t is satisfied, or identity on q_t otherwise.

This state space model can be extended to represent the product automaton, $\mathcal{B} = \mathcal{A} \times \mathcal{P}$. Thus, x_t and q_t are used to represent continuous state and discrete state of the product automaton, at time t, respectively.

The task of state estimation is to obtain the *posterior distribution* $bel(s_t) = p(s_t|Y_t) = p(x_t, q_t|Y_t)$, where $Y_t = \{y_i,\, i = 1, \ldots, t\}$ denoting the observations history up to time t. We use notation $\overline{bel}(s_t) = p(x_t, q_t|Y_{t-1})$ to described the *prior distribution*, belief of s_t without the knowledge of the current observation at time t.

In brief, particle filter is a sequential Monte Carlo implementation of Bayes filter. Particle Filter recursively follows the two essential steps in Bayes filter (1), *prediction* and *correction*.

In the Monte Carlo implementation, a set of M random samples (particles) are drawn from the distribution $bel(s_{t-1})$, $\left\{s_{t-1}^{(i)} = (x_{t-1}^{(i)}, q_{t-1}^{(i)}),\, i = 1, \ldots, M\right\}$, and the distribution can be approximated by

$$bel(s_{t-1}) = p(x_{t-1}, q_{t-1}|Y_{t-1}) \tag{4.10}$$

$$\approx \sum_{i=1}^{M} \delta_{(x_{t-1}^{(i)}, q_{t-1}^{(i)})}(dx_{t-1}, dq_{t-1}) \tag{4.11}$$

The intuition behind this particle presentation is that the more likely the states in a region, the more particles are drawn from the region. So, the number of samples is a key factor to characterize a distribution. With enough particles, this non-parametric representation can describe any type of distribution, as opposed to parametric probability distributions, for example, Gaussian.

Assuming that the posterior distribution at time $t-1$, $bel(s_{t-1})$ is approximated by $\mathcal{S}_{t-1} = \left\{s_{t-1}^{(i)},\, i = 1, \ldots, M\right\}$, the predicted $\overline{bel}(s_t)$ in the prediction step can be obtained by *Sequential*

Importance Sampling (SIS) [93]. In this context, it propagates the particles in $\mathcal{S}_t$ through the probabilistic hybrid model of the system to generate the possible future state at time t. The resulting set $\overline{\mathcal{S}}_t = \left\{ s_t^{(i)},\, i = 1, \ldots, M \right\}$ is the Monte Carlo approximation of $\overline{bel}(s_t)$.

Then, based on Bayes rule, the difference between the *proposal distributions* $\overline{bel}(s_t)$ and *target distribution* $bel(s_t)$ is described by the likelihood $p(y_t|s_t)$. To obtain the Monte Carlo approximation of $bel(s_{t-1})$, the correction step is implemented by *Sequential Importance Resampling* (SIR) [93], in short *Resampling*. For each particle $s_t^{(i)}$ in set $\overline{\mathcal{S}}_t$, the likelihood of the observation y_t which describes the distance away from the actual observation is referred to as *importance weight,*

$$\mathrm{w}_t^{(i)} = p(y_t|s_t^{(i)}) \propto \tag{4.12}$$

Then, according to the importance weights $\left\{ \mathrm{w}_t^{(i)},\, i = 1, \ldots, M \right\}$, particles are resampled with replacement from particle set $\overline{\mathcal{S}}_t$. The set of resampled particles $\mathcal{S}_t$ is the estimated approximation of $bel(s_t)$. Recursively, $\mathcal{S}_t$ is used as prior knowledge towards the next time step. Note that the particles in $\mathcal{S}_t$ are equally weighted.

Next, we will introduce two important modifications that improve the estimation performance.

4.3.2 Particle Filtering Modifications

4.3.2.1 Risk-Sensitivity

In fault detection and identification applications, faults are low-probability but high-risk events. In this case, one drawback of the algorithm is that it requires a very large number of particles to keep track of these faults effectively and thus, is computationally expensive.

Risk-Sensitive PF (RSPF) incorporates some cost function into particle filtering to improve the tracking of these critical fault modes using a smaller amount of particles [15,86,87]. To be specific, an intentional probability increase of the occurrence of the faults is used to propagate particles, thus there are more particles to represent the critical fault modes to during SIS step.

4.3.2.2 Rao-Blackwellization

Another drawback of the generic PF is that in a hybrid system with high-dimensional state space, sampling can be inefficient. However, in some cases, the dimension of the state space to be sampled can be reduced by marginalizing out some variables. This technique is called *Rao-Blackwellization*, and the PF incorporated with this technique is called *Rao-Blackwellized* PF (RBPF) [55].

A hybrid state s_t at time t can be divided into the continuous state x_t, and the discrete mode q_t,$s_t = (x_t, q_t)$.

Then using chain rule, the posterior distribution can be separated as follows:

$$p(s_t|Y_t) = p(x_t, q_t|Y_t) = p(q_t|Y_t)p(x_t|q_t, Y_t)$$

By marginalizing out x_t, particle filtering is only used to obtain $p(q_t|Y_t)$ in the state space with a reduced dimension, which makes resampling more efficient, compared with the generic PF which is used on full-dimensional s_t. The marginalization of x_t can be implemented by Kalman Filter if the noise in state space equations is Gaussian, or some other existing optimal algorithm. The detailed algorithm can be found in [55].

Risk-Sensitivity and Rao-Blackwellization are two important improvements for particle filtering aimed at state space coverage by particle set and efficiency of resampling, respectively.

4.4 Hierarchical Particle Filter

In threshold-based monitors, particle filter is used to evaluate $P_{fault}(\alpha)$. However, in monitoring of concurrent cyber-physical systems modelled by cPHAs, *particle propagation depletion* becomes a challenge, especially when the cPHA is in locally parallel operation.

At each time t, particle filter recursively calculates the Monte Carlo approximations of $\overline{bel}(s_t)$ and $bel(s_t)$, namely prior belief distribution and posterior belief distribution. As described in the *correction* step in the particle filter algorithm in Section 4.3.1, the set of particles $\overline{\mathcal{S}}_t = \left\{ s_t^{(i)},\, i = 1, \ldots, M \right\}$ which represents the prior distribution $\overline{bel}(s_t)$ is resampled according to importance weights $\left\{ \mathrm{w}_t^{(i)},\, i = 1, \ldots, M \right\}$ to obtain the posterior distribution $bel(s_t)$. Resampling algorithm draws M particles with replacement from $\overline{\mathcal{S}}_t$ to obtain $\mathcal{S}_t$ and doesn't generate new particles. Therefore, if a critical transition is not captured in the *prediction* step, the true state is missing in $\overline{\mathcal{S}}_t$ and as a consequence the posterior distribution $bel(s_t)$ will be inaccurate – what is called particle propagation depletion. Moreover, if the missing/depleted states are re-

lated to the computation of $P_{fault}(\alpha)$, particle propagation depletion would result in monitoring failure.

In a risk-sensitive PF, for the risk-critical but low-probable discrete modes, the particles propagated in SIS steps take a cost function into consideration so that a increased number of particles would be in the risk-critical modes [86,87]. But RSPF doesn't prevent the depletion problems in other cases, for example, a cPHA running in locally parallel operation. The local variables and the discrete state variables in locally input decoupled subsystems are mutually independent in locally parallel operation, so the hybrid state space would increase exponentially. In this case, the number of particles M has to be large to cover the state space which dramatically increase the complexity of the algorithm; otherwise, particle propagation depletion in $\overline{\mathcal{S}}_t$ in the prediction step is likely to occur and deteriorate the performance of the threshold-based monitors.

4.4.1 Complexity Analysis

In a cPHA, the hybrid state space of $\mathcal{B} = \mathcal{A} \times \mathcal{P}$ is the reachable set of the Cartesian product of hybrid state space of system automaton and state space of property automaton, $\prod_{i=1}^{N}(\mathbb{R}^{|\mathbf{l}_i|} \times Q_i) \times \mathbb{R}^{|\mathbf{z}|} \times Q_P$. To avoid particle propagation depletion, the number of particles required, M would be sufficient to cover the reachable set of hybrid state space $\prod_{i=1}^{N}(\mathbb{R}^{|\mathbf{l}_i|} \times Q_i) \times \mathbb{R}^{|\mathbf{z}|}$ (Since property automaton $\mathcal{P}$ is deterministically driven by the system automaton, Q_P doesn't affect M). Considering the probability distribution of system model $p(s_t|s_{t-1})$, M can be even larger. When the overall system is running in fully locally parallel operation where $\{\mathbf{l}_i,\, i = 1, \ldots, N\}$ and the discrete state variables are independent, the reachable state space

is exponentially increasing with the number of components, N. If $N = 10$, and assuming the smallest size of the discrete state space is of cardinality 3, the size of the state space will be $3^{10} = 59049$. Potentially, this means that the number of particles should be $3^{10} = 59049$ to guarantee correct state estimation.

In the case of cPHA, different types of concurrency may require dramatically different numbers of particles for adequate estimation performance. We start with the lemma that directly follows from the discussion above.

Lemma 1 *When a concurrent probabilistic hybrid automaton with N subsystems is in fully locally parallel operation, RBPF algorithm has the worst case complexity of $O(c^N)$, $c > 1$. When it is in locally parallel operation where $N'(N' < N)$ subsystems are locally input decoupled, RBPF algorithm has the worst case complexity of $O(c^{N'})$, $c > 1$*

It turns out that by taking advantage of the interconnection structure, it is possible to avoid high computation load and obtain a good estimate when a cPHA is in locally parallel operation. We thus propose a novel particle filtering approach, a *Hierarchical Particle Filter* (HPF).

4.4.2 HPF Algorithm

M is the number of particles used in hierarchical particle filtering algorithm. Recursively, the hierarchical particle filtering algorithm for a concurrent PHA is given as follows:

Algorithm 3 Hierarchical Particle filter $\mathcal{S}_{t-1} = \left\{s_{t-1}^{(j)}\right\} = \left\{(q_{t-1}^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)}), j = 1, \ldots, M\right\}$, y_t

$\overline{\mathcal{S}}_t = \widehat{\mathcal{S}}_t = \mathcal{S}_t = \emptyset$

Step 1: Prediction

for $j = 1$ to M **do**

sample $q_t^{(j)} \sim p(q_t|s_{t-1}^{(j)})$

sample $(l_{1,t}^{(j)}, l_{2,t}^{(j)}, \ldots, l_{N,t}^{(j)}, z_t^{(j)}) \sim p(l_{1,t}, l_{2,t}, \ldots, l_{N,t}, z_t|q_{,t}^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)})$

$\mathrm{w}_t^{(j)} = p(y_t|s_t^{(j)})$

$\overline{\mathcal{S}}_t = \overline{\mathcal{S}}_t + s_t^{(j)}$

Step 2: Hierarchical adaptation of $\overline{\mathcal{S}}_t$

for $i = 1$ to N **do**

Identify the index set $\mathbb{K}_i$ of indeces j where $q_{i,t-1}^{(j)} = q_i^* \in Q_{i,Parallel}$

if $size(\mathbb{K}_i) > P\% * M$ **then**

A: Identify the set of discrete modes Q_i^*that are successors of $q_{i,t-1}^{(j)},\ j \in \mathbb{K}_i$

Normalize $\mathrm{w}_t^{(j)} \leftarrow \mathrm{w}_t^{(j)} / \sum \mathrm{w}_t^{(j)},\ j \in \mathbb{K}_i$

for $j = 1$ to $size(\mathbb{K}_i)$ **do**

Draw m with probability $\propto \mathrm{w}_t^{(\mathbb{K}_i(j))}$

$\widehat{\mathcal{S}}_t = \widehat{\mathcal{S}}_t + s_t^{(\mathbb{K}_i(m))}$

Marginalize the distribution of the discrete mode $q_{i,t}$ from $\widehat{\mathcal{S}}_t$:

$$\overline{\mathrm{w}}_{i,t}(q) = \frac{1}{size(\mathbb{K}_i)} \sum_{s_t^{(j)} \in \widehat{\mathcal{S}}_t} \delta(q_{i,t}^{(j)} = q),\ \forall q \in Q_i^*$$

HPF is developed based on RBPF. The belief distribution of the discrete state variables $p(q_t|Y_t)$ is obtained by particle filtering method, and the belief of continuous state variables $\mathbf{x}_t$ given q_t is obtained by an optimal algorithm. Step 1 and step 3 are not different from a RBPF. The main improvement is with step 2 where the prior/proposal distribution of q_t is adapted. Proposal distribution obtained from SIS step is a prediction of s_t without knowledge from the current observation y_t. When particle propagation depletion occurs in the proposal distribution, the adaptation of the proposal distribution is performed in step 2 with the current observation

The proposal distribution of the discrete state variables of locally input decoupled subsystems are incorporated to made the "distance" between the proposal and target distributions closer.

4.4.2.1 Hierarchical adaptation

In locally parallel operation, the local states and the discrete modes from locally input decoupled subsystems are evolving independently, therefore the adaptation of the proposal distribution of local states and corresponding discrete mode from one locally input decoupled subsystem would not affect another locally input decoupled subsystem. With Rao-Blackwellization, the distribution of the discrete mode of a locally input decoupled subsystem is first adapted. Then, the corresponding local states are adapted given the discrete mode.

At time $t-1$, if subsystem $\mathcal{A}_i$ is believed to be locally input decoupled (more than $P\%$ of the particles have discrete mode $q_{i,t-1} = q_i^*$ where $q_i^* \in Q_{i,Parallel}$ and P is a parameter in percentage), we label the particles that have a discrete mode that is a parallel operated discrete state by index set $\mathbb{K}_i$. From repeated experiments, we choose P to be 70 as a good number to be adequate to represent the majority as well as maintain the variety of the particles.

Algorithm 3 Hierarchical Particle filter $\mathcal{S}_{t-1} = \left\{s_{t-1}^{(j)}\right\} = \left\{(q_{t-1}^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)}), j = 1, \ldots, M\right\}$, y_t (continued)

B: Initialize the discrete mode $\tilde{q}_{i,t}^{(j)}, j = 1, \ldots, size(\mathbb{K}_i)$ to be uniform distributed over Q_i^*, and initialize $\tilde{l}_{i,t}^{(j)}, j = 1, \ldots, size(\mathbb{K}_i)$ to be uniformly distributed over the admissible portions of the state space.

$\tilde{\mathrm{w}}_{i,t}^{(j)} = \overline{\mathrm{w}}_{i,t}(\tilde{q}_{i,t}^{(j)})$, $j = 1, \ldots, size(\mathbb{K}_i)$

Normalize $\tilde{\mathrm{w}}_{i,t}^{(j)} \leftarrow \tilde{\mathrm{w}}_{i,t}^{(j)} / \sum_{j \in \mathbb{K}_i} \tilde{\mathrm{w}}_{i,t}^{(j)}$, $j = 1, \ldots, size(\mathbb{K}_i)$

for $j = 1$ to $size(\mathbb{K}_i)$ **do**

Draw m with probability $\propto \tilde{\mathrm{w}}_{i,t}^{(j)}$

Replace $(q_{i,t}^{(\mathbb{K}_i(j))}, l_{i,t}^{(\mathbb{K}_i(j))})$ in $\overline{\mathcal{S}}_t$ by $(\tilde{q}_{i,t}^{(m)}, \tilde{l}_{i,t}^{(m)})$

Update $z_t^{(j)}, j = 1, \ldots, M$ corresponding to the adapted $(q_{i,t}^{(j)}, l_{i,t-1}^{(j)})$ for each locally decoupled subsystem in $\overline{\mathcal{S}}_t$

C: Evaluate importance weights for particles in the adapted set $\overline{\mathcal{S}}_t$, $\mathrm{w}_t^{(j)} = p(y_t | s_t^{(j)})$, $j = 1, \ldots, M$

Step 3: Correction

Normalize $\mathrm{w}_t^{(j)} \leftarrow \mathrm{w}_t^{(j)} / \sum \mathrm{w}_t^{(j)}$, $j = 1, \ldots, M$

for $j = 1$ to M **do**

draw m with probability $\propto \mathrm{w}_t^{(j)}$

Apply optimal filter to obtain $(l_{1,t}^{(m)}, l_{2,t}^{(m)}, \ldots, l_{N,t}^{(m)}, z_t^{(m)}) \sim Optimal(q_t^{(m)}, s_{t-1}^{(m)}, y_t)$

$\mathcal{S}_t = \mathcal{S}_t + s_t^{(m)}$

Then, at time t, instead of propagating all the particles from time $t-1$ using the system model to obtain $\overline{\mathcal{S}}_t$, we could use the current observation y_t to adapt the discrete mode and the local state of subsystem $\mathcal{A}_i$ in the particles indexed by $\mathbb{K}_i$. Note that the observation is often not made directly on local states of individual subsystems.

Of the particles in $\overline{\mathcal{S}}_t$ indexed by $\mathbb{K}_i$, we first resample according to their importance weights obtained from step 1 which is essentially the likelihood of the particles. The resulting set of particles $\widehat{\mathcal{S}}_t$ has the joint posterior distribution associated with the discrete mode of $\mathcal{A}_i$, though particle propagation depletion may have already occurred in the prior distribution represented by $\overline{\mathcal{S}}_t$. Then, the desired distribution of the discrete mode of subsystem $\mathcal{A}_i$ for adaptation can be marginalized from the joint posterior distribution.

$$\mathrm{w}_{i,t}(q) = \frac{1}{size(\mathbb{K}_i)} \sum_{s_t^{(j)} \in \widehat{\mathcal{S}}_t} \mathbf{1}(q_{i,t}^{(j)} = q),\ \forall q \in Q_i^*$$

We assume that the local states from the locally input decoupled subsystems are conditionally independent given the obseration y_t, then by naive bayes model [94], the joint distribution of all the independent local states is the product of their marginal distributions. The conditional independence assumption is not valid in this case, however, heuristically, the marginal distributions gives the most likely region of the local state, and the independence of the evolution of the local states from locally input decoupled subsystems makes the adaptation of local states from locally independent subsystems not affect each other. So, the combination of marginal distributions reflects the most likely region of the joint state.

To obtain the adapted set of particles whose discrete mode of $\mathcal{A}_i$ follows the marginal distribution $\mathrm{w}_{i,t}(q)$, we initialize the discrete mode $q_{i,t}$ in the set of the particles indexed by $\mathbb{K}_i$ to be uniformly distributed over Q_i^*, the set of successors of q_i^*, and resample to obtain $\tilde{q}_{i,t}^{(j)}$, $j \in \mathbb{K}_i$ (the tilda mark is used to indicate the adapted parts of proposal distribution $\overline{\mathcal{S}}_t$) using the marginal distribution as the importance weight function for each particle. The local states $\tilde{l}_{i,t}^{(j)}$, $j \in \mathbb{K}_i$ are updated accordingly given $\tilde{q}_{i,t}^{(j)}$. Then, $\left\{(\tilde{q}_{i,t}^{(j)}, \tilde{l}_{i,t}^{(j)}),\, j \in \mathbb{K}_i\right\}$ replace $\left\{(q_{i,t}^{(j)}, l_{i,t}^{(j)}),\, j \in \mathbb{K}_i\right\}$ in set $\overline{\mathcal{S}}_t$. Essentially, other than taking *blind* propagated discrete modes and local states for the locally input decoupled subsystem in the proposal distribution, the adapted proposal distribution of $(q_{i,t}, l_{i,t})$ for subsystem $\mathcal{A}_i$ would pick the most probables discrete modes and local states considering the current observation.

The procedure for adaptation is implemented for every locally input decoupled subsystem, and then the importance weights of the adapted particles are evaluated for resampling in step 3.

4.4.2.2 Complexity Reduction

In locally parallel operation, without using an exponentially increasing number of particles that would cause exponentially increasing complexity in an existing particle filtering algorithm, HPF divides each particle based on the concurrency in subsystem operation, and applies adaptation for the corresponding divided part of the particles in the proposal distribution for each individual locally input decoupled subsystems. Therefore, the number of particles, M is only required to be sufficient to cover the state space of each individual locally input decoupled subsystem.

Lemma 2 *When a concurrent probabilistic hybrid automaton with N components is running in the mode of fully locally parallel operation when all the subsystems are fully decoupled, hierarchical particle filtering algorithm has the space complexity of $O(c \cdot N)$.*

4.4.2.3 More discussion

Algorithm 3 assumes the general case that the observation model is on both link and local states, $y_t = h_{q_t}(l_{1,t}, \ldots, l_{i,t}, \ldots, l_{N,t}, z_t) + w_{y,t}$, and the current observation y_t is be used to obtain the marginal distribution of the discrete mode q_i of the locally input decoupled subsystems for the adaptation in proposal distribution at time t.

Next we take a look at other cases about the observation model and discuss the necessary modification of HPF accordingly.

- Case 1: The observation only depends on link variables.

$$y_t = h_{q_t}(l_{1,t}, \ldots, l_{i,t}, \ldots, l_{N,t}) + w_{y,t}$$

When system is in *fully* locally parallel operation,

$$bel(s_t) = p(q_t, l_{1,t}, \ldots, l_{N,t}, z_t | Y_t) = p(q_t, l_{1,t}, \ldots, l_{N,t} | Y_t) p(z_t | q_t, l_{1,t}, \ldots, l_{N,t}, Y_{t-1})$$

The second part is simply the conditional prediction of z_t. The first part is a reduced dimensional filtering/estimation problem on $\{\mathbf{l}_i, i = 1, \ldots N\}$ and q_t which follows the HPF algorithm introduced above.

If the system is in locally parallel operation, but not in fully locally parallel operation, then there is at least one subsystem which is not decoupled. Assume $\mathcal{A}_i$ is a non-decoupled subsystem, and the evolution of $l_{i,t}$ or $q_{i,t}$ is dependent on z_{t-1}. In resampling with weights based on $l_{i,t}$ in step 3, the propagated z_t is not correlated to the current observation y_t in time. Therefore, as of resampling in step 3, the posterior belief should be for $(z_{t-1}, q_t, l_{i,t})$ in every time step. So, the modification is in step 1, prediction from $(l_{i,t-2}, z_{t-2}, q_{t-1}, l_{i,t-1})$ to $(z_{t-1}, q_t, l_{i,t})$.

Algorithm 4 Hierarchical Particle filter – Case 1 $\mathcal{S}_{t-1} = \left\{(q_{t-1}^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-2}^{(j)}), j = 1, \ldots, M\right\}$, y_t

Step 1: Prediction

for $j = 1$ to M **do**

sample $z_{t-1}^{(j)} \sim p(z_{t-1}|q_{t-1}^{(j)}, l_{t-2}^{(j)}, z_{t-2}^{(j)})$

sample $q_t^{(j)} \sim p(q_t|q_{t-1}^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)})$

sample $(l_{1,t}^{(j)}, l_{2,t}^{(j)}, \ldots, l_{N,t}^{(j)}) \sim p(l_{1,t}, l_{2,t}, \ldots, l_{N,t}|q_t^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)})$

$\mathrm{w}_t^{(j)} = p(y_t|l_{1,t}^{(j)}, l_{2,t}^{(j)}, \ldots, l_{N,t}^{(j)})$

$\overline{\mathcal{S}}_t = \overline{\mathcal{S}}_t + s_t^{(j)}$

- Case 2: The observation only depends on the link variables.

$$y_t = h_{q_t}(z_t) + w_{y,t}$$

At time t, local states $(l_{1,t}, \ldots, l_{N,t})$ is not observed. The evolution of link states at time t is dependent on the local states at time $t-1$, $(l_{1,t-1}, \ldots, l_{N,t-1})$. So, in this case, the filtering of $(l_{1,t-1}, \ldots, l_{N,t-1})$ is actually smoothing, which requires the future observation y_t. So, the posterior belief should be made for $(l_{i,t-1}, q_t, z_t)$ recursively. Similarly, in step 1, prediction is modified to be from $(l_{i,t-2}, z_{t-2}, q_{t-1}, z_{t-1})$ to $(l_{i,t-1}, q_t, z_t)$.

If the system is in locally parallel operation, then the proposal distribution adaptation in step 2 is towards independent $(l_{i,t-1}, q_{i,t})$ for the decoupled subsystems. Besides, z_t should also be updated reflecting the adaptation of the local states $l_{i,t-1}$ and the discrete modes $q_{i,t}$ for the decoupled subsystems.

- Case 3: The observations can be decoupled into cases 1 and 2.

$$y_t^a = h_{q_t}^a(l_{1,t}, \ldots, l_{i,t}, \ldots, l_{N,t}) + w_{y,t}^a,\ y_t^b = h_{q_t}^b(z_t) + w_{y,t}^b$$

This is a special case where the observation can be divided into two group, (y_t^a, y_t^b) such that y^a is a function of local states, and y^b is a function of link states. The weight of a particle in the proposal distribution is

$$\begin{aligned} \mathrm{w}_t^{(j)} &= p(y_t|s_t^{(j)}) = p(y_t^a, y_t^b|s_t^{(j)}) \\ &= p(y_t^a|l_{1,t}^{(j)}, \ldots, l_{N,t}^{(j)}) p(y_t^b|z_t^{(j)}) \\ &= \mathrm{w}_t^{a\,(j)} \mathrm{w}_t^{b\,(j)} \end{aligned}$$

Algorithm 5 Hierarchical Particle filter – Case 2 $\mathcal{S}_{t-1} = \left\{(q_{t-1}^{(j)}, l_{1,t-2}^{(j)}, l_{2,t-2}^{(j)}, \ldots, l_{N,t-2}^{(j)}, z_{t-1}^{(j)}), j = 1, \ldots, M\right\}$, y_t

Step 1: Prediction

for $j = 1$ to M **do**

sample $(l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}) \sim p(l_{1,t-1}, l_{2,t-1}, \ldots, l_{N,t-1} | q_{t-1}^{(j)}, l_{t-2}^{(j)}, z_{t-2}^{(j)})$

sample $q_t^{(j)} \sim p(q_t | q_{t-1}^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)})$

sample $z_t^{(j)} \sim p(z_t | q_t^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)})$

$\mathrm{w}_t^{(j)} = p(y_t | z_t^{(j)})$

$\overline{\mathcal{S}}_t = \overline{\mathcal{S}}_t + s_t^{(j)}$

Step 2: Hierarchical adaptation of $\overline{\mathcal{S}}_t$

for $i = 1$ to N **do**

Identify the index set $\mathbb{K}_i$ of j where $q_{i,t-1}^{(j)} = q_i^* \in Q_{i,Parallel}$

if $size(\mathbb{K}_i) > P\% * M$ **then**

$\vdots$

B: Initialize the discrete mode $\tilde{l}_{i,t-1}^{(j)}, j = 1, \ldots, size(\mathbb{K}_i)$ to be uniform distributed, and update $\tilde{q}_{i,t}^{(j)}, j = 1, \ldots, size(\mathbb{K}_i)$ over Q_i^* correspondingly

$\vdots$

Update $z_t^{(j)}, j = 1, \ldots, M$ corresponding to the adapted $(q_{i,t}^{(j)}, l_{i,t-1}^{(j)})$ for each locally decoupled subsystem in $\overline{\mathcal{S}}_t$

$\vdots$

To improve the efficiency of resampling, similarly as in case 2, the prediction is for $(l_{i,t-1}, q_t, z_t)$, so after the adaptation of $(l_{i,t-1}, q_{i,t})$ for decoupled subsystems, z_t is also updated for adapted proposal distribution. In this case, to obtain $\widehat{\mathcal{S}}_t$ in part A of step 2, which is used to calculate the marginal distribution of $\mathrm{w}_{i,t}(q)$ for adaptation of $(l_{i,t-1}, q_{i,t})$, we take the set of particles indexed by $\mathbb{K}_i$ resampled according to $\mathrm{w}^a_{t-1} = p(y^a_{t-1}|l_{1,t-1}, \ldots, l_{N,t-1})$. Then, for resampling done in step 3, we would only use $\mathrm{w}_t^{b\,(j)} = p(y^b_t|z^{(j)}_t)$ as the importance weights.

Algorithm 6 Hierarchical Particle filter – Case 3 $\mathcal{S}_{t-1} = \left\{(q_{t-1}^{(j)}, l_{1,t-2}^{(j)}, l_{2,t-2}^{(j)}, \ldots, l_{N,t-2}^{(j)}, z_{t-1}^{(j)}), j = 1, \ldots, M\right\}, y_t$

Step 1: Prediction

for $j = 1$ to M **do**

sample $(l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}) \sim p(l_{1,t-1}, l_{2,t-1}, \ldots, l_{N,t-1} | q_{t-1}^{(j)}, l_{t-2}^{(j)}, z_{t-2}^{(j)})$

sample $q_t^{(j)} \sim p(q_t | q_{t-1}^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)})$

sample $z_t^{(j)} \sim p(z_t | q_t^{(j)}, l_{1,t-1}^{(j)}, l_{2,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)}, z_{t-1}^{(j)})$

$\mathrm{w}_{t-1}^{a\,(j)} = p(y_{t-1}^a | l_{1,t-1}^{(j)}, \ldots, l_{N,t-1}^{(j)})$

$\mathrm{w}_t^{b\,(j)} = p(y_t^b | z_t^{(j)})$

$\overline{\mathcal{S}}_t = \overline{\mathcal{S}}_t + s_t^{(j)}$

Step 2: Hierarchical adaptation of $\overline{\mathcal{S}}_t$

for $i = 1$ to N **do**

Identify the index set $\mathbb{K}_i$ of j where $q_{i,t-1}^{(j)} = q_i^* \in Q_{i,Parallel}$

if $size(\mathbb{K}_i) > P\% * M$ **then**

A: Identify the set of discrete modes $Q_i^*$of successors of q_i^*

Normalize $\mathrm{w}_{t-1}^{a\,(j)} \leftarrow \mathrm{w}_{t-1}^{a\,(j)} / \sum \mathrm{w}_{t-1}^{a\,(j)}$, $j \in \mathbb{K}_i$

for $j = 1$ to $size(\mathbb{K}_i)$ **do**

Draw m with probability $\propto \mathrm{w}_{t-1}^{a\,(\mathbb{K}_i(j))}$

$\widehat{\mathcal{S}}_t = \widehat{\mathcal{S}}_t + s_t^{(\mathbb{K}_i(m))}$

Marginalize the distribution of the discrete mode $q_{i,t}$ from $\widehat{\mathcal{S}}_t$:

$$\overline{\mathrm{w}}_{i,t}(q) = \frac{1}{size(\mathbb{K}_i)} \sum_{s_t^{(j)} \in \widehat{\mathcal{S}}_t} \mathbf{1}(q_{i,t}^{(j)} = q),\ \forall q \in Q_i^*$$

Algorithm 6 Hierarchical Particle filter – Case 3 $\mathcal{S}_{t-1} = \left\{(q_{t-1}^{(j)}, l_{1,t-2}^{(j)}, l_{2,t-2}^{(j)}, \ldots, l_{N,t-2}^{(j)}, z_{t-1}^{(j)}), j = 1, \ldots, M\right\}$, y_t (continued)

B: Initialize the discrete mode $\tilde{l}_{i,t-1}^{(j)}, j = 1, \ldots, size(\mathbb{K}_i)$ to be uniform distributed, and update $\tilde{q}_{i,t}^{(j)}, j = 1, \ldots, size(\mathbb{K}_i)$ over Q_i^*correspondingly

$\vdots$

Update $z_t^{(j)}, j = 1, \ldots, M$ corresponding to the adapted $(q_{i,t}^{(j)}, l_{i,t-1}^{(j)})$ for each locally decoupled subsystem in $\overline{\mathcal{S}}_t$

C: Evaluate importance weights $\mathrm{w}_t^{b\,(j)}$ for the particles in the adapted set $\overline{\mathcal{S}}_t$, $\mathrm{w}_t^{b\,(j)} = p(y_t^b | z_t^{(j)}), j = 1, \ldots, M$

$\vdots$

In the next section, we demonstrate a monitoring example on a multi-car train system, and show the experimental results of hybrid state estimation using the Hierarchical Particle filter.

4.5 Experiment

A train with electronically-controlled pneumatic (ECP) brakes [71] is used as an example. The train has N cars in a sequence connected by rigid links between every two neighbor cars. Each car has its own braking apparatus, and each brake works and fails independently. When

the train runs normally, the velocity is controlled by the engine operation. Due to disturbances or noises, when it's running too fast, a braking signal is sent to all the cars simultaneously, then each car starts engaging their own brakes independently to help bring the velocity down to its normal range. After some random dwell time, The brake may fail with a small probability $p_o = 0.1$. As long as at least one brake works, the train slows down. When the velocity of the train falls back to the preset point, all the brakes are released simultaneously and all the cars are controlled by the engine again. During the whole process, a speed sensor measures the velocity of the whole train, and two force sensors measure the forces only on the first and last rigid links.

The desired behavior of the train is given by the following specifications. First, the train itself shall adjust the velocity v when the it is beyond the desired range due to noises or some other disturbances, which is a liveness property. Second, the forces exerted on the links between each neighboring car cannot exceed the limit force it can afford. A disaster such as the rupture of the whole train would occur if this is not prevented, which is a safety property.

A *concurrent* probabilistic hybrid automaton $\mathcal{A}$ is used to model the train. Formally, it is comprised of two types of subsystems: velocity subsystem $\mathcal{A}_v$ and braking subsystem $\mathcal{A}_b$.

The velocity subsystem $\mathcal{A}_v$ (6) gives how the velocity (v) evolves. Three modes (three states of discrete variable q_v) are included in the velocity subsystem. The train starts in the discrete mode $q_v = 1$ and remains in that mode until the velocity exceeds the threshold $V_U = 28.5$, then it switches to the mode $q_v = 2$. The train remains in $q_v = 2$ until one of the brakes engages and it switches to state $q_v = 3$. $\sum_{i=1}^{N} b_i$ in the guard condition of transition from $q_v = 2$ to $q_v = 3$

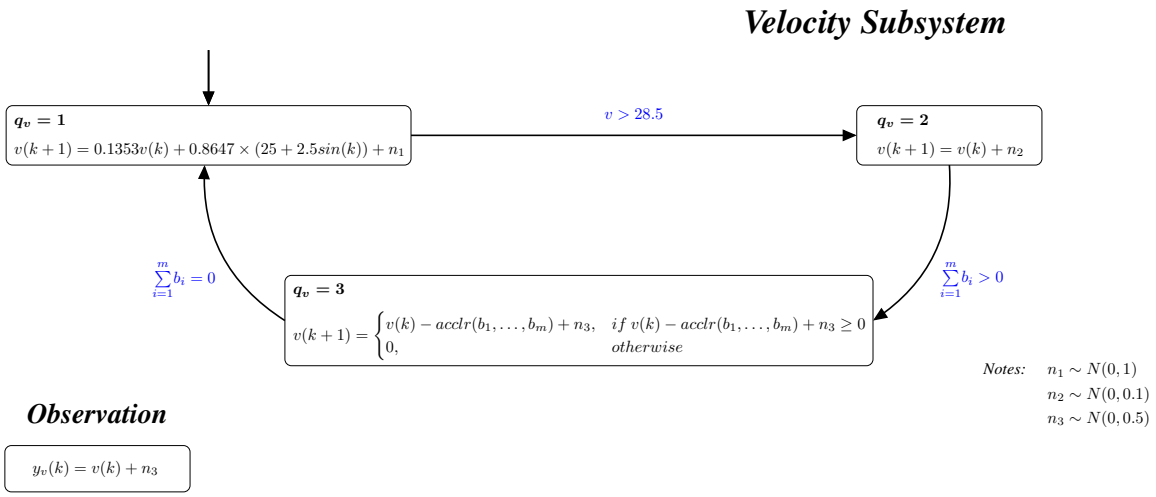

Figure 6: Velocity subsystem

represents the number of brakes that are engaged in the braking subsystems. When all the brakes disengage, the velocity subsystem switches back to the state $q_v = 1$. When $q_v = 1$, the velocity is controlled to oscillate around 25 with amplitude 2.5, which lies in range $[22.5, 27.5]$. When $q_v = 2$, the velocity keeps constant, and decreases with a deceleration which depends on the braking pattern when $q_v = 3$.

The individual braking system $\mathcal{A}_b$ is described in Figure 7.

The braking subsystem starts in the discrete mode $q_b = 1$ and remains in that mode until the velocity exceeds a threshold $V_U = 28.5$, when it switches to the mode $q_b = 2$. The braking subsystem remains in $q_b = 2$ until the timer c_1 reaches T_1 (modeling delay in actuation and computational delays). Note that the initial value of the timer c_1 in the state $q_b = 2$ is not deterministic, so the duration of time the system remains in $q_b = 2$ is a random variable. After

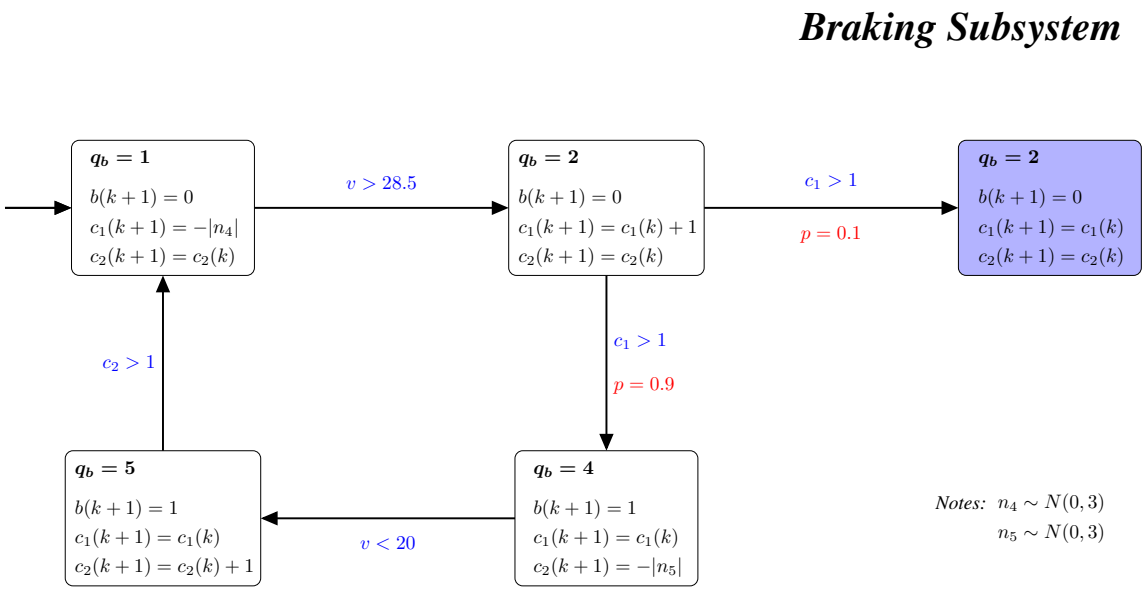

Figure 7: Braking subsystem

the timer reaches T_1, the braking system can fail with a probability $p_o = 0.1$ and permanently switch to $q_b = 3$, or switch to $q_b = 4$ with probability $1 - p_o = 0.9$ and engages in the brake. When the brake engages, the variable b is set to be 1, thereby affecting the velocity of the train as described above. When the velocity falls below $V_L = 20$, the brake disengages after a random amount of time (modeled by the timer c_2 in mode $q_b = 5$, another modeled random delay in actuation and computation), when it switches to the state $q_b = 1$.

The braking subsystem $\mathcal{A}_b$ described above (7) is for a single car. However, the braking parameters are not identical for all the N cars, so the computation of acceleration is dependent on the braking pattern. With each of the $N = 5$ brakes engaged or not, there are $2^5 = 32$

braking patterns. Specifically, the 5 cars have exactly the same mass $m = 12$, and the braking power of each car in unit N is configured to be $F(n) = 11.7+9.6\times1.2^n$, $n = 1, \ldots, N$. Therefore, by Newton's Law, for each of the 32 braking pattern, the acceleration of the whole train and the forces on links between neighboring cars can be calculated, and listed in Figure 8. The patterns that would cause the violation of link force property.

Braking pattern	Car 1	4	0	4	4	4	4	0	4	4	4	0	4	4	0	4
	Car 2	4	4	0	4	4	4	0	0	4	4	4	0	4	4	0
	Car 3	4	4	4	0	4	4	4	0	0	4	0	4	0	4	4
4: brake engaged;	Car 4	4	4	4	4	0	4	4	4	0	0	4	0	4	0	4
0: brake not engaged	Car 5	4	4	4	4	4	0	4	4	4	0	4	4	0	4	0
Link forces (N)	Link 1	-5.625	-24.2	-0.521	0.032	0.696	1.492	-19.1	5.137	6.354	7.813	-18.54	5.801	7.15	-17.88	6.597
	Link 2	-8.95	-22.88	-24.26	2.37	3.696	5.288	-38.19	-12.95	15.01	17.93	-11.56	-11.62	16.6	-10.24	-10.03
	Link 3	-9.50	-18.79	-19.71	-20.82	9.46	11.85	-29	-31.03	-1.855	30.81	-30.11	-0.749	0.534	0.172	1.64
	Link 4	-6.742	-11.39	-11.85	-12.4	-13.06	21.73	-16.49	-17.5	-18.72	15.41	-17.04	-18.17	16.07	-17.71	16.62
Acclr (m/s^2)		2.40	2.02	1.98	1.93	1.877	1.811	1.591	1.507	1.406	1.284	1.545	1.452	1.339	1.49	1.385

Braking pattern	Car 1	0	0	4	4	0	0	4	0	0	4	0	4	0	0	0	0
	Car 2	4	0	0	4	4	4	0	0	0	0	4	0	4	0	0	0
	Car 3	4	0	0	0	0	4	4	4	4	0	0	0	0	4	0	0
4: brake engaged;	Car 4	4	4	0	0	0	0	0	0	4	4	4	0	0	0	4	0
0: brake not engaged	Car 5	0	4	4	0	4	0	0	4	0	0	0	0	0	0	0	4
Link forces (N)	Link 1	-17.08	-13.44	11.46	13.47	-12.22	-10.76	12.92	-12.78	-11.98	12.25	-11.43	18.58	-5.105	-5.658	-6.321	-7.118
	Link 2	-8.644	-26.88	-0.303	29.25	1.079	3.999	2.62	-25.55	-23.96	1.29	2.67	13.93	15.31	-11.32	-12.64	-14.24
	Link 3	2.561	-40.32	-12.06	19.5	-11.14	21.53	20.60	-10.04	-7.65	-9.68	-8.75	9.29	10.21	11.32	-18.96	-21.35
	Link 4	17.08	-22.15	-23.83	9.749	-23.37	10.76	10.3	-22.81	11.98	10.97	11.43	4.644	5.105	5.658	6.321	-28.47
Acclr (m/s^2)		1.424	1.12	0.98	1.625	1.019	0.897	0.86	1.06	1.00	0.91	0.95	0.39	0.43	0.47	0.53	0.59

Figure 8: Calculated link forces and accelerations for braking patterns of $N = 5$ cars

The output of the system is the measured velocity

$$y_v = v + n_3$$

where $n_3 \sim N(0, \frac{1}{2})$ and the measured link forces on the first and last links, f_f and f_l,

$$\begin{cases} y_{f_f} = f_f + n_6 \\ y_{f_l} = f_l + n_7 \end{cases}$$

where $n_6 \sim N(0, \frac{1}{3})$ and $n_7 \sim N(0, \frac{1}{6})$.

According to the specifications, the property automaton $\mathcal{P}$ describing the specifications are given in Figure 9:

The specifications contain a liveness property regarding v and safety property regarding link forces. The liveness automaton for the liveness property is converted to a safety automaton with a timer $counter_1$. The starting and accepting mode is $q_1 = 1$, when the $v > V_U$ for more than $counter_1$ time units continuously, the specification is considered to be violated, and it permanently comes to *fault mode* $q_1 = 3$. The second safety automaton specifies the link force safety property. The starting mode is $q_2 = 1$, and the accepting modes are $q_2 = 1$ and $q_2 = 2$. $q_2 = 3$ is the *fault mode* when the maximum force on all the 4 links $f_{max} > 30$ for more than a continuous time interval of $counter_2$ time units.

Upon analyzing the cPHA $\mathcal{A}$, which is the composition of the velocity subsystem $\mathcal{A}_v$ and N copies of the braking subsystems $\mathcal{A}_b$, the link variables are velocity v, and variables $\{b_i, i = 1, \ldots, N\}$. For the velocity subsystem $\mathcal{A}_v$, there's no local variable, so $\mathcal{A}_v$ is never decoupled; for braking subsystems $\mathcal{A}_b$, 2 timers c_1 and c_2 are local variables, and $Q_{b,parallel} = \{2, 5\}$ where $\mathcal{A}_b$ is decoupled.

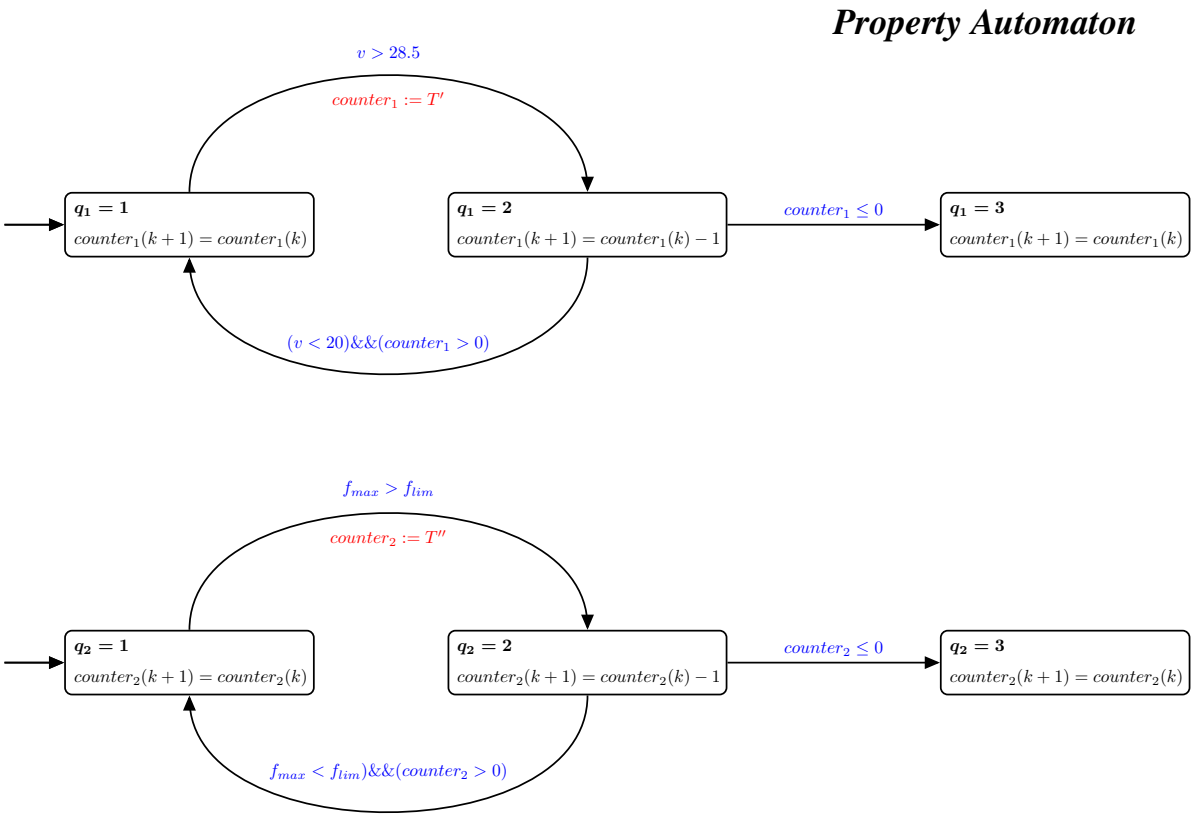

Figure 9: Property automaton

When $q_b = 2$ at time t for the i_{th} braking subsystem, then depending on the local state c_1, q_b can be propagated to 3 modes at time $t+1$, i.e., $Q_b^*(q_b^* = 2) = \{2, 3, 4\}$ as the set of the successors of $q_b^* = 2$. When $q_b = 5$ at time t for the i_{th} braking subsystem, depending on the local state c_2, q_b can be propagated to 2 modes at time $t+1$, i.e., $Q_b^*(q_b^* = 5) = \{1, 5\}$ as the set of the successors of $q_b^* = 5$.

A sample of discrete trajectory of system automaton would be

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \end{pmatrix} \rightarrow \begin{pmatrix} 2 \\ 2 \\ 3 \\ 2 \\ 2 \\ 2 \end{pmatrix} \rightarrow \begin{pmatrix} 3 \\ 2 \\ 3 \\ 4 \\ 2 \\ 4 \end{pmatrix} \rightarrow \begin{pmatrix} 3 \\ 4 \\ 3 \\ 4 \\ 2 \\ 4 \end{pmatrix} \rightarrow \begin{pmatrix} 3 \\ 4 \\ 3 \\ 4 \\ 4 \\ 4 \end{pmatrix} \rightarrow \begin{pmatrix} 3 \\ 5 \\ 3 \\ 5 \\ 5 \\ 5 \end{pmatrix} \rightarrow \begin{pmatrix} 3 \\ 1 \\ 3 \\ 5 \\ 5 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 \\ 1 \\ 3 \\ 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \dots$$

The execution will be repeated every time the velocity subsystem switches back to $q_v = 1$, and we call such a repetition an operation cycle. During an operation cycle, when the guard condition $v > 28.5$ is satisfied in joint discrete mode $q = (1, 1, \dots 1)$, the velocity subsystem and all the braking subsystems are switched to $q_b^* = 2$ at the same time. Then, with $N = 5$, the size of the discrete state space explodes to $3^N = 243$ resulting 32 braking patterns related to the link force safety property. So, the estimation failure may happen when the stochastic failure of

each braking subsystem occurs after the braking signal is sent (e.g., $v > 28.5$). So, hierarchical PF is used in this situation until less than two braking subsystems stays in $q_b^* = 2$.

An estimation failure is a divergent estimate of hybrid state which leads to a misalarm or false alarm based on the property automata. Therefore, with each individual timer c_1 expired randomly, $\mathcal{A}_b$ starts braking in $q_b = 4$ with probability $1 - p_o$. When $v < 20$, $\mathcal{A}_b$ switches to another $q_b^* = 5$ with $Q_b^*(q_b^* = 5) = \{1, 5\}$, and the discrete state space explodes to $2^5 = 32$. However, since no failure would occur after timer c_2 in $q_b = 5$ expires, then the explosion of the discrete state space is temporary and unrelated to the two properties to verify. Therefore we don't take hierarchical adaptation for particles in proposal distribution in this situation.

4.5.1 Performance Evaluation

From simulation, the Figure 10 gives an example execution of the train system. The blue and green lines are the trajectories of velocity v of the velocity subsystem, and the discrete state q_1 of the liveness automaton regarding the velocity. The red and black lines are the calculated maximum link force f_{max} and the discrete state q_2 of the safety automaton regarding the link force. It is observed that at about time step 400, the force property is violated (the maximum link force $f_{max} > 30$, and then $q_2 = 3$).

Figure 11 shows estimation results from RBPF and HPF, respectively. With $M = 400$ particles, the RBPF estimate fails when the train system is in locally parallel operation in all the four operation cycles. On the other side, hierarchical PF shows a significant improvement in performance. This is the situation of case 3 where two observations are made on local states and link states, respectively.

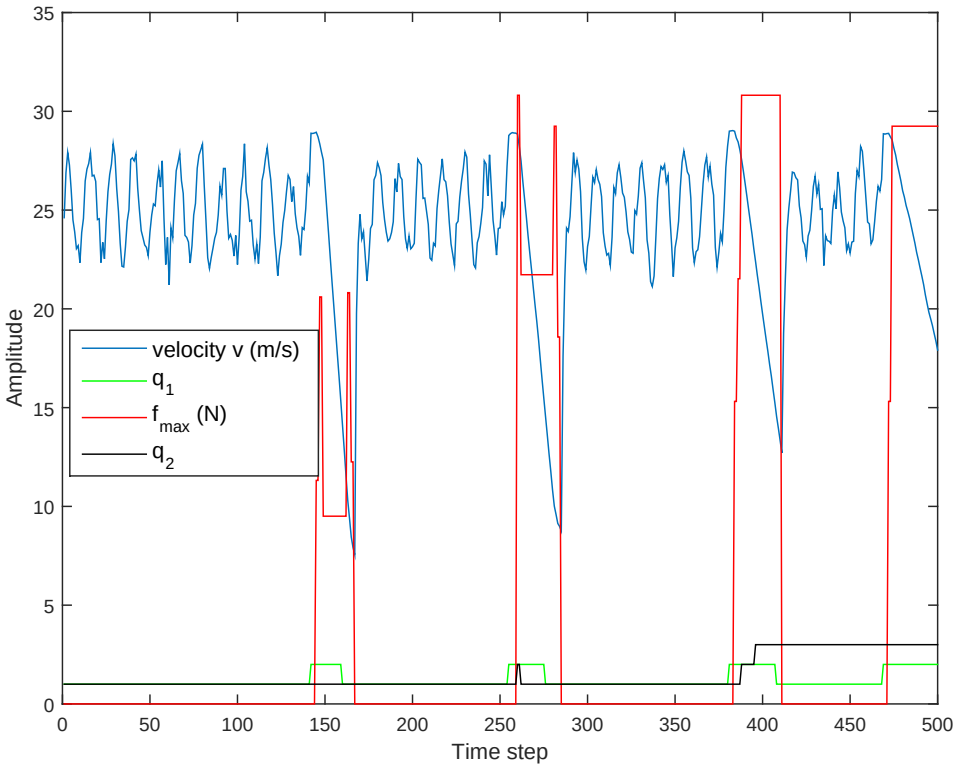

Figure 10: "True" hybrid state trajectory of the train system

In the train example, a single run can have multiple operation cycles that require the brakes to be engaged and that could violate the link force property. By running a simulation 1000 times and counting each operation cycle, the rate of estimation failure is less than 3%, which dramatically improves the estimation performance.

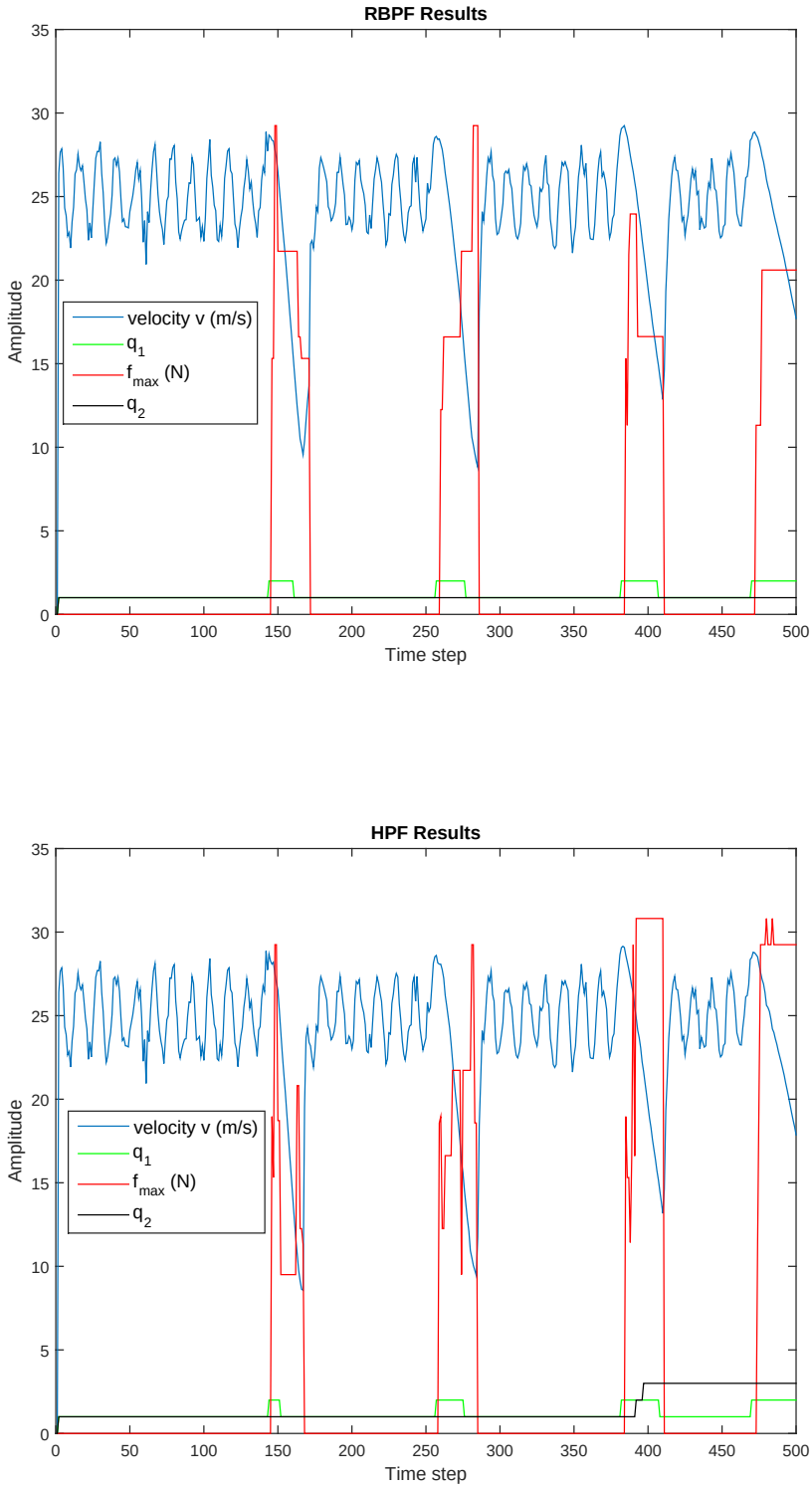

Figure 11: Comparison of Estimated hybrid state trajectory by RBPF and Hierarchical PF

CHAPTER 5

FAULT DETECTION AND DIAGNOSIS FOR SAFETY-CRITICAL CYBER-PHYSICAL SYSTEMS

5.1 Introduction

For safety-critical systems, there is a growing need to guarantee system safety on-the-fly. Runtime Fault Detection and Diagnosis (FDD) has been developed and applied to achieve safety and reliability of systems. *Fault detection* entails recognizing that a fault happened, while *fault diagnosis* requires finding the cause and location of the fault. An extensive range of research has been done to develop effective FDD methods, especially on *model-based* FDD [95]. Many solutions were developed and investigated since 1980s in the control community. Some important developments include parity equations, state observer/estimation, and parametric-based approaches [17, 63, 95, 96]. Since 1990s, model-based FDD publications focused more on specific aspects such as robustness and sensitivity, diagnosis-oriented modeling, or robust isolation. Today, the model-based FDD is considered as a mature field of research.

Researchers have also extended the existing model-based fault detection and diagnosis methods to hybrid systems. Hybrid state estimation techniques for both discrete mode and continuous state were developed in [24, 52, 97–99]; Structural partity residual was used in Fault Detection and Diagosis for hybrid systems in [50]; In [100], based on the model of hybrid bond graphs, an online hybrid observer is designed for parameter fault detection and used qualitative and

quantitative methods for fault isolation; [101] used analytic redundancy method for mode detection, and extended minimal structurally overdetermined (MSO) approach for to hybrid MSO (HMSO) for fault diagnosis in hybrid systems.

In modern safety-critical cyber-physical systems, e.g. automobiles, with ever more functions integrated in a more compact system, it is expensive to locate the root cause of a fault/failure. Failure Mode and Effect Analysis (FMEA) is widely used in automotive industry as a tool to analyze the root causes of detected failures [102]. Design FMEA (DFMEA) [103] is performed during product development to achieve robust system design. Process FMEA (PFMEA) [103] is applied to evaluate potential process induced failures before production phase begins. Society for Automotive Engineers (SAE) in 1994 first published related standard J1739 which standardizes the process and provides the worksheet which guides the FMEA process.

In the process of FMEA, it is important to identify all the potential *effects and causes* of a potential failure mode thoroughly. For each of the potential effects/causes, three rankings are assigned: Severity, Occurrence, and Detection. Action taken to eliminate or reduce high-risk failure mode is based on Risk Priority Number (RPN), which is calculated as the product of the 3 rankings. Note that each potential failure mode can have more than one potential effect, and each potential effect also can have multiple potential causes. The severity ranking is evaluated on a potential effect, and the occurrence and detection rankings are evaluated on the potential causes of the failure mode.

A person with expertise for FMEA (for example, the design engineer for a DFMEA and process or process engineer for a PFMEA process) can bring tremendous insight to the whole

team, specifically, the detailed knowledge of the potential effects and causes of each potential failure mode. However, the existing process cannot guarantee a systematic analysis on the possible causes of each potential failure mode. Currently, brainstorming to list all the potential failure modes as well as the corresponding effects and causes is the main method used in automotive industry. Therefore, if a potential cause of a specific failure mode is not listed, then the remaining processes, e.g., prioritizing the failure modes and actions to take to eliminate or reduce failure modes, will not meet the expected outcome.

Motivated by this industrial process and our previous work on monitoring on safety-critical hybrid systems, we propose a new model-based FDD framework for hybrid systems. The proposed framework attempts to address shortcomings of existing model-based FDD approaches and provide a better fit for applications. In particular:

1. The fault models in modern hybrid systems can be complex, e.g. not just defined on quantitative deviation (constant or adaptive thresholds) of parameter, internal states or output, which is typical to see in existing model-based FDD. In modern hybrid systems, *time* is often a critical aspect of a fault. For example, a fault-tolerant control system should be able to perform adaptive control for self-recovery within a certain amount of time t_0; a fault thus occurs when the system deviates from the expected behavior continuously for a period of time $t > t_o$.

2. Performing fault diagnosis usually requires prior knowledge of the faults, including the modeling of faults. In existing model-based FDD, the faults are usually classified into two types: *additive faults* representing faults of actuators and sensors and *multiplicative faults*

representing faults leading to a parameter change. Modern hybrid systems need a more general way to represent complex faults, including random faults and temporal patterns of safety violations.

3. Model-based approaches for FDD rely on the analytic model of the system and the faults. However, the analysis of faults and their effects may not be accurate or even complete. Although the knowledge about certain faults may be incomplete, we still need to identify their occurrences and provide as much information as possible to correct them.

4. In existing model-based FDD of hybrid systems, diagnosis is mostly done offline using generated residual signals and the analysis/knowledge of system faults. These methods often include learning and other soft computing techniques. There is a growing need for runtime robust diagnosis of system faults which does not need large existing data sets.

5. Most existing algorithms are confined to single fault diagnosis.

All these call for an alternative framework for model-based FDD. In this work, we propose such novel framework, *Property-Based* FDD (PB-FDD) with a focus on safety-critical hybrid systems. To distinguish Property-Based FDD from the existing model-based FDD, we name the latter *classical* model-based FDD. The major improvements made in property-based FDD include the following:

1. We use Linear Temporal Logic (LTL) to define faults. This enables us to specify complex time-dependent faults.

2. Traditionally, faults are defined based on the nominal model of the system or a component. As a result, existing methods are mainly confined to FDD at the component level, where models are available. In contrast, in our framework, fault is defined more generally as a violation of a system specification. In particular, the specification can often be given without actually knowing the model of the system. In addition, hybrid automata formalism also allows us to model the faults as a discrete transition to a faulty mode that is explicitly included in the model of the component; this approach mirrors the traditional way of defining a fault, but without the need to identify it as an *additive* or a *multiplicative* fault.

3. Our methodology uses particle filtering algorithm to generate a fault signature; particle distribution is used to detect the fault. However, this approach also allows us to use the information provided by the importance weight distribution of the particle set. In particular, using the importance weight distribution, it is possible to identify the inconsistency of the particle filter and possibly of the combined system/fault model.

4. Since our PB-FDD is model based, a violation of the property can be directly traced to a particular component, thus naturally providing the fault diagnosis.

5. Multiple violations can easily be detected and diagnosed in parallel in the proposed property-based FDD.

5.2 Background

The motivation for the property-based FDD work is product design verification for automotive products, where DFMEA is the key resource/input to plan for effective design verification.

An efficient framework of FDD can help expedite the design-verification-debugging phase and dramatically cut the time and cost.

5.2.1 DFMEA and Product Design

FMEA techniques have been around in safety applications for more than fifty years, and now is widely applied as standard in industries, especially in automotive industry for quality control purpose.

DFMEA is conducted in the product design stage, focused on preventing defects, enhancing safety and increasing customer satisfaction. The objective of DFMEA is to identify how well the system requirements are met and perform product improvement early in the product design phase. A design verification procedure is created according to the results from DFMEA to maximize the effectiveness of product verification or testing to detect the defects in the product.

In the DFMEA process, based on the knowledge of every member in the DFMEA team, all the possible ways of potential failure should be considered in the process, including the problems in product design as well as the possible user mistakes and environmental disturbances. For each potential failure mode, all the potential effects and corresponding potential causes (one or multiple for each potential effect) are listed, and a Risk Priority Number (RPN) is assigned to each pair of effect and cause.

$$RPN = S \times O \times D$$

where

- S is the Severity ranking of the potential effects on a scale of $1-10$;
- O is the Occurrence ranking of the potential causes on a scale of $1-10$;
- D is the Detection ranking of the detection control strategy on a scale of $1-10$.

To eliminate the failure modes with high RPN or reduce RPN for some failure modes, the occurrence ranking can be reduced/eliminated by the investigation the root cause and improvement/fix of the defect, and the detection ranking can be reduced by the improvement of test/verification to increase the detectability of the failure. An update RPN is calculated again upon the action taken.

An important advantage of FMEA is its completeness. With an effective DFMEA, failures in the design can be easily detected and their root causes identified. However, in practice, since brainstorming to list all the potential failure modes and the corresponding effects and causes is the main method applied, DFMEA cannot be guaranteed to cover all the effects and causes of the possible failure modes. If a failure is detected, and the actual root cause is unlisted, then the debugging could be costlier, including ruling out all the listed root causes and identifying the actual defects. Moreover, since a proper RPN is not assigned for the unknown root cause of the failure, the action has to be put on hold, too.

In the product development process in automotive industry, since an unlisted root cause is likely to be a product design defect, it has important practical meaning to recognize the unlisted root causes and locate the most recently working configuration. While in the design phase, it is necessary for design and testing to be repeated several times to debug all the software/hardware

defects. Therefore, diagnosing an unlisted root cause can greatly increase the chance to product design improvement.

We propose a framework of property-based fault detection and diagnosis to formalize the process where FMEA result is taken into the account. On the other side, the proposed framework is able to supplement the missing pieces of FMEA analysis.

5.2.2 A Motivating Project

5.2.2.1 Auxiliary Transmission Fluid Pump Motor Controller

For hybrid electric vehicles, one important feature is "idle-off" which turns off the conventional engine when the vehicle is stopped temporarily at stoplights or in traffic to save fuel, also known as a *start-stop system*. The electric motor can start the vehicle again once the stop condition is removed. An auxiliary transmission fluid pump is used to provide the transmission with proper hydraulic pressure required to support the restart process.

A Brushless DC (BLDC) motor is used to drive the pump so it is necessary to design a pump motor Electronic Control Unit (ECU). The controller takes inputs from the Engine Control Module (ECM), and provides information on its own operation status. A project at Magna Electronics Inc is the controller design for the BLDC motor.

5.2.2.2 System Requirements

The most critical system requirement for the ECU (the auxiliary transmission fluid pump motor controller) is that the pump should run continuously at the desired speed once a start signal followed by a proper speed command is received from the ECM, unless an exception is detected, in which case the cause of the exception should be reported.

Motor Operation	Status	PWM output DC (%)
Running	Normal	47.5 ± 2.5
Stopped	Over Current Exception	22.5 ± 2.5
Stopped	Over Temperature Exception	37.5 ± 2.5
Stopped	Over Speed Exception	67.5 ± 2.5

TABLE II: MOTOR STATUS VS. PWM OUTPUT DC

The start signal and the speed command are mapped to Pulse-Width Modulation (PWM) signals. The start signal has a fixed duty cycle (DC) of 15%. The relationship between the DC of the PWM signal and the speed command is described by a linear equation $RPM = 20{*}DC{+}300$. Basically, the speed command linearly maps the interval$[800, 2000]rpm$ to the DC range of $[25, 85]\%$.

The output of the ECU is a PWM signal, with the DC indicating its operation status, i.e., normal operation or an exception condition. The mapping is described in Table II:

5.2.2.3 Software Design Diagram

The software design diagram is shown in Figure 12. In software design, there are seven modes of operation, *Init*, *Wait*, *Shutdown*, *Alignment*, *Startup* and *Operation* and *Exception*. The system enters *Init* when the ECU is powered up, and switched to *Wait* when a PWM signal is received regardless of its DC. Upon receiving a start signal, the system switches to *Shutdown*. In

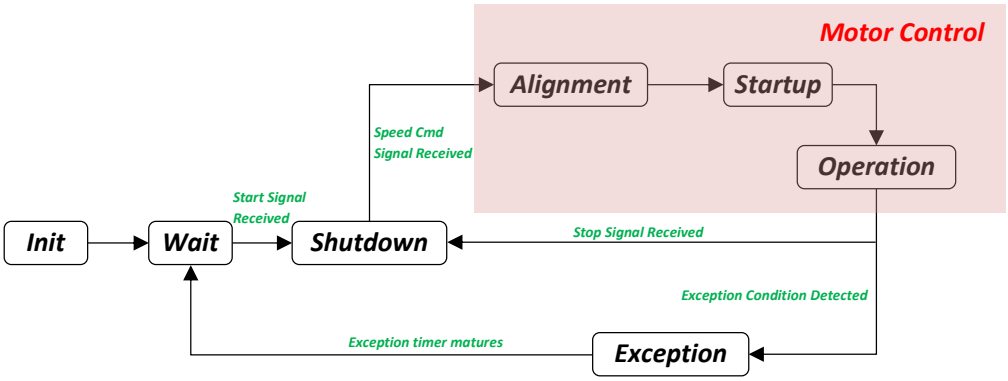

Figure 12: Stateflow of Software design for ePump Controller

Shutdown, if a valid speed command is received, the motor control starts to work. The motor control algorithm has three modes: *Alignment*, *Startup* and *Operation*. The system first switches to *Alignment* to reset the rotor position of the BLDC motor, and then in *Startup*, the motor is controlled to start in an open-loop mode without rotor position feedback. In *Operation*, the motor is controlled to operate at the desired speed continuously in a closed-loop mode. When a stop command signal is received, the system switches back to *Shutdown*. If any exception listed in Table II is found in *Operation*, the system would switch to *Exception* mode and trigger a maturing timer. If the exception vanishes before the timer matures, the system switches back to *Operation*; if the exception persists until the timer matures, the system would switch to *Wait*, and wait for the next start signal.

5.2.2.4 Problems

DFMEA is performed for the pump motor Electronic Control Unit (ECU) in the early stage shown in Table III. The three exceptions listed in Table II are root causes for the failure mode in which the motor stops, it has high severity since the engine would be stopped without enough hydraulic pressure. The occurrence is obtained with actual testing data. The detection is built in the software design, so it has low detection ranking. More potential failure modes are investigated, including that the motor couldn't start, the motor runs at a lower speed and the motor speed fluctuates.

During the actual testing, another fault, rough motor start, occurred. The motor attempted to start for about 3 seconds and stopped permanently without any known exception indicated by PWM out DC. There are also no matching root causes according to the FMEA worksheet in Table III. Battery voltage was in good range; motor was not stalled; all faults were cleared before the motor started; transmission fluid was good condition.

5.3 Modeling

5.3.1 Hybrid Systems

First of all, property-based FDD is a model-based FDD where the nominal system and fault model are known. We focus on hybrid systems which is modeled by Probabilistic Hybrid Automata (PHA).

Please refer to Section 4.2.1.1 for a detailed description of a PHA and its semantics.

Function	Potential Failure Mode	Potential Effects	Severity	Potential Causes	Occurrence	Current Control	Detection	RPN
Motor spins at desired speed	Motor stops	Engine stopped	8	Stall (Over Current Fault matures)	2	None	4	48
			8	Ambient temperature too high (Over Temperature Fault matures)	1	None	1	8
			8	Over Speed Fault matures	4	None	3	32
	Motor couldn't start	Engine not starting up	9	An old matured fault was not cleared	1	Software fault table check	1	9
	Motor runs at a lower speed	Engine may not starting up	4	Battery level gets low	2	Battery level	2	8
			4	Transmission Fluid viscosity increases	1	None	3	12
	Motor speed oscillates	Rough Engine start up	6	Motor control algorithm is not sensitive to the EMF waveform zero crossing	6	None	5	180

TABLE III: FMEA WORKSHEET

5.3.2 Fault Models

The modeling of faults is one of the most important parts of model-based fault detection and diagnosis.

Generally, a fault is defined as follows:

Definition 9 (Fault) *Generally speaking, a fault is an unpermitted violation of at least one of the requirements.*

Since in PB-FDD, faults are defined implicitly through requirements, it helps to clearly define the scope of these requirements. Following the industrial product development process, the system requirements are first collected and analyzed without determination of the software/hardware design. The ability to define faults at this level, also called *system level faults*, is an important advantage of PB-FDD. As the design of individual components starts, the corresponding requirements for each component are developed; the violations of these are called *component level faults*.

An example would be an automotive embedded electronic control unit (ECU). System requirements describe the expected behavior of the product from the user's perspective including but not limited to the functional, performance, and security requirements. Instead, component requirements are limited to the function of specific components, e.g., application software, control algorithm, hardware, etc. This motivates the following definitions:

Definition 10 (System level fault) *A system level fault is a fault which violates the system requirements independent of the system design.*

Definition 11 (Component level fault) *A component level fault is a fault which violates the component requirement. The component requirement may or may not refer to the component design.*

5.3.2.1 System Level Faults

A system level fault indicates the violation of the system requirements. This can be caused by many factors including a bad component design, a stochastic failure, an uncontrollable disturbance or a slow deterioration of hardware. System level faults are the potential failure modes in FMEA and system level fault detection is often an integral part of the design verification process.

The system requirements are created from a user's perspective, and thus, are independent of the system design (model). The system requirements are defined on the hybrid state of the system, and quite often, *time* is an important element in the description of the desired behavior. Because of that, it is convenient to describe system requirements using Linear Temporal Logic (LTL) formulas, typically in the form of safety and/or liveness properties [11, 104]. Any LTL formula can be transformed to a Streett automaton (4.2.2.1) $\mathcal{P}$ such that the language accepted by $\mathcal{P}$ is equivalent to the LTL formula [105,106]. While classical FDD typically relies on residual signals or parameter change detection, PB-FDD provides a much more general decision rule. Namely, a fault is detected when an execution of a system is rejected by the Streett automaton $\mathcal{P}$ (the absorbing *error state* of $\mathcal{P}$ is reached in infinite horizon).

5.3.2.2 Component Level Faults

Component level faults refer to the violation of component requirements. They correspond to failures or malfunctions of components. Within the framework of FMEA, they are the root causes of the failure modes. In PB-FDD, we only consider the component level faults that would lead to a system level fault, i.e., the violation of system requirements.

Using hybrid automata formalism, it is natural to model a component level fault as a discrete mode $q^F \in Q^F$, where Q^F is the set of all the discrete fault modes. The structural knowledge of the component level faults is given by results obtained from FMEA process. In FMEA, for each failure mode (formalized to be a system level fault), multiple potential root causes are identified. From either data-driven or analytic analysis, $\forall q^F \in Q^F$, the behavior of the system under the condition of a particular root cause is modeled by continuous dynamics $\epsilon(q^F)$. The onset of the root cause is triggered by a transition from some other discrete state q which has a set of transitions$\mathcal{T}(q) = \{(\phi, p)_{q,\lambda}\}_{\lambda \in J_q}$ when$\exists \lambda \in J_q$ such that $p_{q,\lambda}(q^F) > 0$. Compared with the classical FDD where faults are modeled either as additive or multiplicative faults, our proposed model of component level faults is much more general.

With component level faults properly incorporated into the system model, their detection essentially diagnoses a system level fault at the same time.

However, as discussed in Section 5.2.1, a root cause that is not listed is also important in product design since it is likely to be a neglected design defect. We thus formally define unlisted potential root causes as *unknown component level faults*.

Unknown Component Level Faults

Similar to known component level faults, an unknown component level fault is also modeled by a discrete mode q^{UF}. Without any information from FMEA, the dynamics of the unknown faults as well as the triggering transition are unknown. Therefore, a single discrete mode q^{UF} represents an entire family of unknown component level faults. Unknown component level faults can occur at any discrete mode $q \in Q$. So, implicitly we assume that there is a transition from any $q \in Q$ to q^{UF}.

5.4 Structure of Property-Based Fault Detection and Diagnosis

The PB-FDD scheme, illustrated in Figure 13, consists of two parts: fault detection and fault diagnosis.

Fault detection is to detect system level faults, the violation of correctness defined in systems requirements. The correctness is specified by a deterministic Streett automaton $\mathcal{P}$, referred as *property automaton* [12]. The component level faults are incorporated into the system model $\mathcal{A}$. The input alphabet for $\mathcal{P}$ is a sequence of hybrid state of $\mathcal{A}$. The rejection probability given a (finite) sequence of output α is computed and compared with a probability threshold z. The fault occurrence is confirmed when $RejProb(\alpha) > z$. The lower bound on the rejection probability is the probability $P_{fault}(\alpha)$ that any absorbing error state of $\mathcal{P}$ is reached. The computation $P_{fault}(\alpha)$ is formulated as belief propagation on the product automaton $\mathcal{B} = \mathcal{A} \times \mathcal{P}$ using PF.

In classical FDD, fault diagnosis is triggered by the detection of a fault based on a generated residual. However, in PB-FDD, fault diagnosis is performing at the same time as fault detection.

The diagnoser is monitoring on *particle inconsistency* in PF, which basically indicates that PF doesn't yield a reliable estimate. Formally,

Definition 12 (Particle Inconsistency) *There are no particles in the predicted set of particles in the true discrete mode or that adequately describes the true continuous state.*

If *particle inconsistency* is not detected, diagnosis is essentially the identification of the component level faults modeled by q^F in the system model; if a particle inconsistency is detected, two hypotheses of the cause are proposed. One hypothesis is *particle propagation depletion* where no particle is propagated to be close to the actual state during SIS, and the other hypothesis is an *unmodeled component level fault*. Unmodeled component level fault indicates *system model inconsistency*, thus PF doesn't have a precise model to keep track of a specific component level fault. Hypotheses testing is employed to detect both *particle inconsistency* in PF and *system model inconsistency*.

In Section 4.4, we focused on particle propagation depletion. However, it was assumed that this is caused by the state space explosion due to a specific type of concurrency in subsystem operation. Then, hierarchical PF was proposed to avoid the estimation failure. In this chapter, we focus on the detection of general particle inconsistency rather than one solution in a specific case.

5.5 Fault Detection

The fault detection is confirmed when $P_{fault}(\alpha) > z$, where z is a preset threshold. This is the threshold based monitor proposed in Section 3.1.3. The belief propagation of $P_{fault}(\alpha)$ is essentially the implemented by using the particle filter , a recursive hybrid estimation technique.

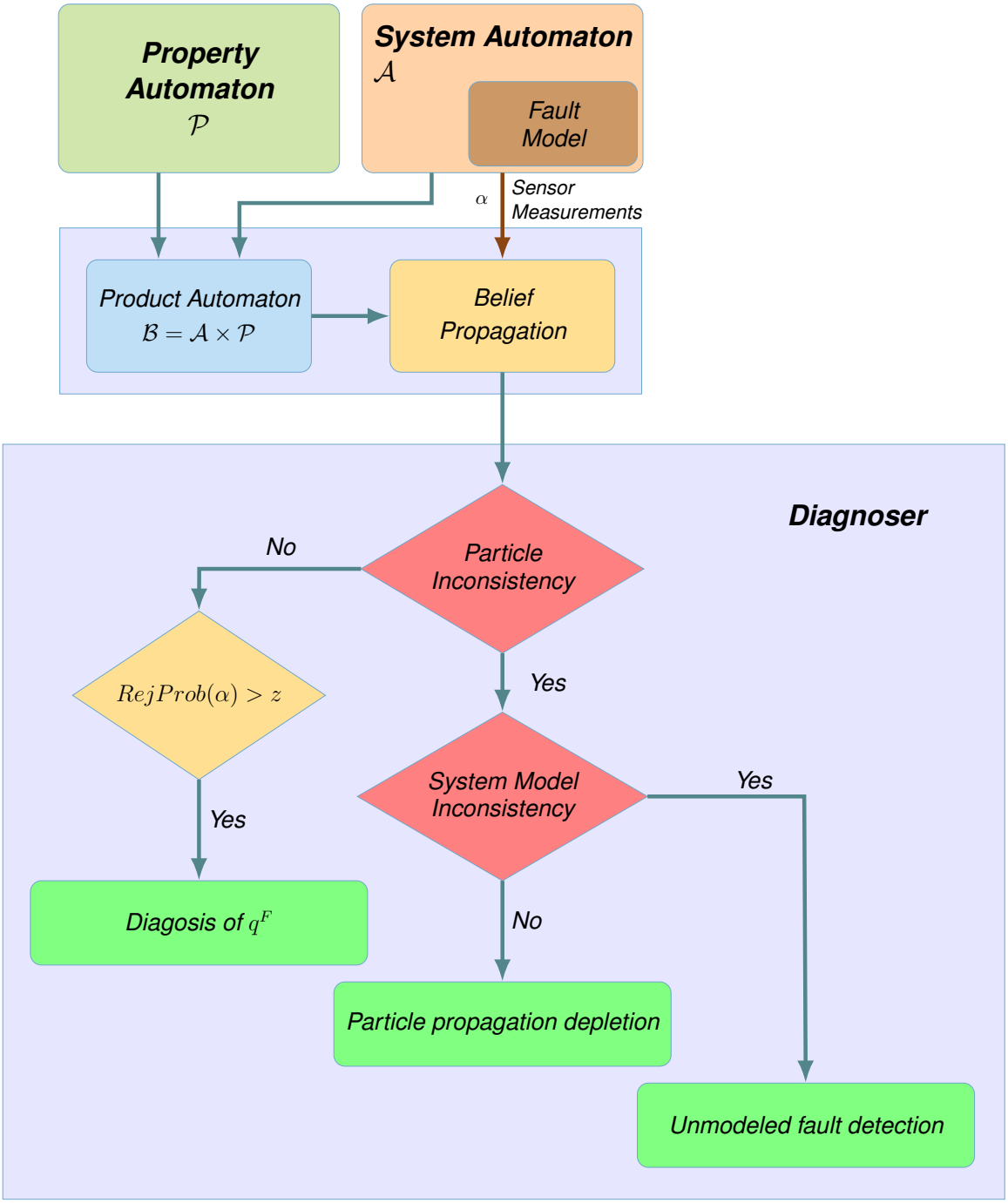

Figure 13: PB-FDD architecture

In estimation problem of the complex hybrid systems, the high dimensional hybrid state space, nonlinearity and stochastic state-dependent transitions, are intractable using the conventional estimation techniques. PF is considered as an efficient tool for such a computational problem. Please refer to the detail of a generic PF and its variations in Section 4.3.

Moreover, in the proposed property-based fault detection strategy, multiple requirements can be represented by multiple property automata, and there is no restriction to detect multiple violations in parallel.

5.5.1 Monitorability

Before a particle filtering algorithm is implemented for fault detection problem for hybrid systems as described above, note that the aim is to estimate the probability that the error state of the property automaton has been reached; we are not specifically interested in the information about the internal states of the system automaton. Therefore, rather than observability, fault detection in the case of PB-FDD only requires monitorability [12] of the system with respect to the specific property.

5.5.2 Design of property automaton

We limit the system requirements in the thesis to be Linear Temporal Logic (LTL) formulas, which can be always represented by the conjunction of a safety and a liveness properties [9, 10].

For a safety property, *safety automaton* specifies a set of absorbing error modes/states. For a liveness property, modeled by a *liveness automaton*, infinite horizon would be required to verify that the property has been violated. To implement the fault detection in that case, an extra timer is introduced to convert a liveness automaton to a safety automaton so that after

a certain amount of time specified by the timer, if the system remains in the error state the liveness property is considered to be violated.

Multiple properties can be converted to multiple parallel property automata where the error states are defined independently. Thus, in fault detection, by evaluating the probability of the error state for each property automaton, multiple faults can be detected simultaneously.

5.6 Fault Diagnosis

The purpose of fault diagnosis is to locate the root cause of a fault. In PB-FDD, FMEA analysis results are used to model component level faults. Then, when a system level fault is detected, diagnosis entails identifying which component level fault occurred. With particle filter, the estimate of the discrete state variable of the system automaton identifies the component level fault.

However, diagnosing the fault in this way assumes that the particle filter yields reliable results. To test this, we should evaluate the performance of the particle filter. In the PF algorithm, the importance weight contains the information that can be used for this purpose.

5.6.1 Importance Weights

In the particle filtering algorithm introduced in Section 4.3.1, in *prediction* step, the prior distribution $\overline{bel}(s_t) = p(s_t|Y_{t-1})$, represented by a set of Monte Carlo particles $\overline{\mathcal{S}}_t$, is obtained by SIS, and each particle $s_t^{(i)}$ in $\overline{\mathcal{S}}_t$ is weighted by the likelihood of the observation y_t given $s_t^{(i)}$, $\mathrm{w}_t^{(i)} = p(y_t|s_t^{(i)})$. In the *correction* step, SIR is performed on the set of particles $\overline{\mathcal{S}}_t$ according to their importance weights $\{\mathrm{w}_t^{(i)}, i = 1, \ldots, M\}$.

The importance weights reflect the *distance* between two distributions, $\overline{bel}(s_t)$ and $bel(s_t)$, named proposal and target distribution, respectively,

$$\mathrm{w}_t(s_t) = p(y_t|s_t) \propto \frac{bel(s_t)}{\overline{bel}(s_t)}.$$

Definition 13 *A set of random samples* $\{s^{(i)}, i = 1, \ldots, M\}$ *is* properly weighted *by* $\{\mathrm{w}(s^{(i)}),\ i = 1, \ldots, M\}$ *with respect to the posterior distribution* π *if for any integrable function* h*,*

$$\lim_{M\to\infty} \frac{\sum_{i=1}^{M} h(s^{(i)})\mathrm{w}(s^{(i)})}{\sum_{i=1}^{M} \mathrm{w}(s^{(i)})} = E_\pi(h(S)).$$

If a set of random samples $\{s^{(i)}, i = 1, \ldots, M\}$ is properly weighted by $\{\mathrm{w}(s^{(i)}),\ i = 1, \ldots, M\}$ then $\{(s^{(i)}, \mathrm{w}(s^{(i)})), i = 1, \ldots, M\}$ are statistically equivalent to the set of samples resampled from $\{s^{(i)}, i = 1, \ldots, M\}$ according to $\{\mathrm{w}(s^{(i)}), i = 1, \ldots, M\}$. For particle filters, it can be shown that the proposal distribution is properly weighted by the weight function which is the likelihood of the particle in the proposal distribution.

It is commonly known that in a particle filter, particle degeneracy and particle impoverishment are the two difficult challenges [13]. If the set of propagated particles after SIS tend to have all but a few particles with negligible weights, particle degeneracy occurs. It can be usually addressed by the procedure of resampling, implemented in the PF introduced in Section 4.3.1. However, resampling could bring another problem, particle impoverishment, where the "important" particles are duplicated many times and other particles with small weights are discarded.

It could dramatically decrease the diversity of the particles. The extreme case would be that all the survived particles are duplicated from one particle.

An important measure, *Effective Sample Size (ESS)* based on importance weights was proposed to detect particle degeneracy in the proposal distribution.

$$ESS = \frac{M}{1 + var_{\overline{bel}(s_t)}(\mathbf{w}_t)}$$

Heuristically, ESS can be interpreted as the number of i.i.d. random samples drawn from the target distribution. The detection of degeneracy is usually based on a certain threshold, i.e., when ESS is below the threshold.

5.6.2 Detection of Particle Inconsistency

In PF, resampling doesn't generate new particles but duplicates or discards particles, so with a limited number of particles M, if the actual state which is being observed is missing from the proposal distribution, ineffective resampling would occur, resulting in PB-FDD failure. Thus, we focus on such a specific phenomenon, *particle inconsistency* (12). It occurs before resampling but should be distinguished from the degeneracy problems which ESS is helpful to detect. Compared with particle impoverishment where the general diversity of particle is reduced, particle inconsistency is associated with the observed actual state which may be depleted.

Moreover, by using RBPF, in the assumption that the process noise for the continuous state evolution is Gaussian, the continuous state in each particle is represented by a Gaussian distribution with the two parameters, mean and variance. Even though particle impoverishment

may occur after resampling, the actual state may not be depleted. In hybrid systems, the discrete mode evolution is based on the continuous state, so we resample every step to increase ESS at time t so that the particle propagation to the next time step $t+1$ can cover all the possible transitions and avoid particle inconsistency in the next time step.

When *particle inconsistency* in proposal distribution occurs, no particle in the set of propagated particles is in the vicinity of the actual state. So one manifestation is that the importance weights for all the particles are low, in other words, the importance weight sample mean is low. Therefore, the importance weight sample mean is selected to detect the occurrence of particle inconsistency.

A hypothesis test is employed for the detection. Recall that hypothesis testing has four steps introduced in Section 2.4:

1. Null and Alternative Hypotheses formulation
2. Test statistics selection
3. Decision rule making
4. Decision to accept or reject the null hypothesis based on observed data.

The hypothesis test is used for every time step sequentially. So, first, we assume that the estimate result from the last time step is reliable.

Assumption. At time $t-1$, in the proposal particle set $\overline{\mathcal{S}}_{t-1} = \{s_{t-1}^{(i)}, i = 1, \dots M\}$, it is assumed that there is no particle inconsistency.

Null and Alternative Hypotheses. The hypothesis to be tested is

H_0 : There is no particle inconsistency in the proposal particle set $\overline{\mathcal{S}}_t$

and the alternative hypothesis is

H_1 : There is particle inconsistency in the proposal particle set

Test Statistic. We choose the importance weight sample mean $\overline{\mathrm{w}}_t = \frac{1}{M}\sum_{i=1}^{M} \mathrm{w}_t^{(i)}$.

Sampling Distribution of Importance Weight Sample Mean. Let the sample mean of random samples $\{x^{(i)}, i = 1, \ldots, N\}$ drawn from a distribution $p(x)$ be denoted as $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$. The mean and the variance of $p(x)$ are denoted as μ and σ^2, respectively. According to central limit theorem, with a large sample size ($N > 30$), the sample mean is approximately normally distributed, $\bar{x} \sim N(\mu_{\bar{x}}, \sigma_{\bar{x}}^2)$ with mean $\mu_x = \mu$, and variance $\sigma_{\bar{x}}^2 = \frac{\sigma^2}{N}$.

Let $\mu_{\mathbf{w}_t}$ and $\sigma_{\mathbf{w}_t}^2$ be the mean and variance of the importance weight variable $\mathbf{w}_t$ at time t, respectively. Then, the sampling distribution of importance weight sample mean is approximately normally distributed with mean of $\mu_{\mathbf{w}_t}$ and variance of $\frac{\sigma_{\mathbf{w}_t}^2}{M}$.

Next, we will need to find the $\mu_{\mathbf{w}_t}$ and $\sigma_{\mathbf{w}_t}^2$ of the importance weight variable $\mathbf{w}_t$.

For $\mathrm{w} \in [0, 1]$, the underlying cumulative distribution function of $\mathbf{w}_t$ can be obtained:

$$
\begin{aligned}
F_{\mathbf{w}_t}(\mathrm{w}) &= \mathbb{P}(\mathbf{w}_t \leqslant \mathrm{w}) \\
&= \mathbb{P}_{\overline{bel}(s_t)}(p(y_t|s_t) \leqslant \mathrm{w}) \\
&= \int_{p(y_t|s_t)\leqslant \mathrm{w}} \overline{bel}(s_t) ds_t \\
&= \int_{p_{\boldsymbol{w}_{y,t}}(y_t - r_{q_t}(x_t))\leqslant \mathrm{w}} \overline{bel}(s_t) ds_t
\end{aligned}
\tag{5.1}
$$

$\mathcal{S}_{t-1} = \{s_{t-1}^{(i)}, i = 1, \ldots, M\}$ is the particle representation of distribution $bel(s_{t-1})$. Note particles in $\mathcal{S}_{t-1}$ are equally weighted. Then analytically, $\overline{bel}(s_t)$ can be represented by

$$
\begin{aligned}
\overline{bel}(s_t) &= \frac{1}{M} \sum_{i=1}^{M} p(s_t|s_{t-1}^{(i)}) \\
&= \frac{1}{M} \sum_{i=1}^{M} p_{w_{x,t}}(x_t - f_{q_t}(x_{t-1}^{(i)}))
\end{aligned}
\tag{5.2}
$$

By replacing $\overline{bel}(s_t)$ by (5.2), we get

$$
F_{\mathbf{w}_t}(\mathrm{w}) = \frac{1}{M} \int_{p_{\boldsymbol{w}_{y,t}}(y_t - r_{q_t}(x_t))\leqslant w} \sum_{i=1}^{M} p_{w_{x,t}}(x_t - f_{q_t}(x_{t-1}^{(i)})) ds_t \tag{5.3}
$$

If $p(y_t|s_t)$ is defined to be a distribution with parametric model, $p(y_t|s_t) \leqslant \mathrm{w}$ can be solved, and $F_{\mathbf{w}_t}(\mathrm{w})$ can be obtained analytically.

The mean $\mu_{\mathbf{w}_t}$ and variance $\sigma^2_{\mathbf{w}_t}$ of $\mathbf{w}_t$ can be computed given cumulative distribution function $F_{\mathbf{w}_t}(\mathrm{w})$. So, the sample mean $\overline{\mathrm{w}}_t = \frac{1}{M}\sum_{i=1}^{M} \mathrm{w}_t^{(i)}$ is approximately normally distributed with $\overline{\mathrm{w}}_t \sim N(\mu_{\mathbf{w}_t}, \frac{\sigma^2_{\boldsymbol{w}_t}}{M})$.

Decision. Upon evaluating the test statistics, if the test falls into the critical region, the null hypothesis is rejected. Otherwise, the null hypothesis is accepted.

For a significance level γ, the critical value for the test is:

$$t^*_\gamma = \mu_{\mathbf{w}_t} + \frac{\sigma_{\mathbf{w}_t}}{\sqrt{M}}\Phi^{-1}(\gamma)$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution.

Therefore, if the observed sample mean $\overline{\mathrm{w}}_t < t^*_\gamma$, the null hypothesis is rejected.

5.6.3 Causes for particle inconsistency

It could be that the error resulted from the random propagation in SIS step following the system model, where the observation is not taken into account, and the actual state is depleted, referred to as *particle propagation depletion*. This is the error introduced by the algorithm. However, it's not the only cause. Another cause is related to the unknown discrete fault, which is the lack of a priori knowledge to be modeled in the system model, referred to as *system modeling inconsistency*. By identifying the occurrence of the system modeling inconsistency, an unmodeled fault is detected. Both causes would lead to particle inconsistency, causing a PF failure.

Distinguishing these two causes has significant meaning in verification/testing in industrial applications. Particle propagation depletion is the error from the particle filtering algorithm, which is the tool to perform the verification task. Meanwhile, with system modeling inconsistency, the system model provided is not consistent with the behavior of the system, which could be a design problem.

5.6.3.1 Particle Propagation Depletion

At time t, during the propagation step from $bel(s_{t-1})$, if the set of propagated particle, $\overline{\mathcal{S}}_t = \{s_t^{(i)}, i = 1, \ldots, M\}$ doesn't keep track of a discrete mode or the vicinity of a continuous state which turns out to be the actual state, it would lead to inefficient resampling, thus estimation failure. Particle propagation depletion could be due to a large hybrid state space and/or low-probable transitions so that the propagated particles miss the critical piece of information from the system model.

5.6.3.2 System Modeling Inconsistency

System model is used as input for PF. In the case of system modeling inconsistency, there is no particle in PF which carry the information of the unmodeled mode. If the system is in a specific unmodeled mode, then particle inconsistency occurs.

The difference between the two causes is whether the missing mode is known by the PF algorithm. In the first scenario, the correct system model is used in SIS, but the set of propagated particles doesn't cover the "important" states which have relatively higher weights, therefore the corresponding importance weight distribution doesn't meet the expected $F_{\mathbf{w}_t}(\mathrm{w})$. However, in the second scenario, the distribution of the importance weights of the propagated particles is close to the expected weight distribution.

As analyzed, the cause of particle inconsistency is essentially determined by how well the importance weights of the particles in the propagated set fit the expected distribution of importance weights. Thus, the decision is made by a goodness of fit test (2.4.1).

5.6.3.3 Kolmogorov-Smirnov Test

The Kolmogorov–Smirnov test (K-S test) [68] is a statistical hypothesis test that is widely used for non-parametric goodness of fit test. It is used to compare a empirical distribution to a hypothesized distribution. When particle inconsistency is detected, we compare the importance weights of the particles in the propagated set to the expected importance weight distribution $F_{\mathbf{w}_t}(\mathrm{w})$ derived in (5.3). If the importance weights matches $F_{\mathbf{w}_t}(\mathrm{w})$, system modeling inconsistency occurs; otherwise, particle propagation depletion occurs.

Null and Alternative Hypotheses. The importance weights of particles in the propagated set are independently distributed with the expected distribution $F_{\mathbf{w}_t}(\mathrm{w})$. The null hypothesis is the occurrence of system modeling inconsistency. And the alternative hypothesis is the occurrence of particle propagation depletion in the proposal distribution.

$$H_0: \text{ There is system modeling inconsistency}$$

$$H_1: \text{ There is particle propagation depletion}$$

Test statistic. The test statistic is the distance between the two cumulative distribution functions (cdf).

The expected cdf of $\mathbf{w}_t$ (5.3) is

$$F_{\mathbf{w}_t}(\mathrm{w}) = \int_{p_{\boldsymbol{w}_{y,t}}(y_t - r_{q_t}(x_t)) \leqslant \mathrm{w}} \overline{bel}(s_t) ds_t$$

where

$$\overline{bel}(s_t) = \frac{1}{M}\sum_{i=1}^{M} p_{w_t}(x_t - f_{q_t}(x_{t-1}^{(i)}))$$

The empirical distribution function is $F_{\mathbf{w}_t}^M(\mathrm{w}) = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_{(-\infty,\mathrm{w}]}(\mathrm{w}_t^{(i)})$.

So, the test statistics is

$$D^M = \sup_{\mathrm{w}} \left| F_{\mathbf{w}_t}^M(\mathrm{w}) - F_{\mathbf{w}_t}(\mathrm{w}) \right|$$

By the Glivenko–Cantelli theorem [107] , if the sample comes from distribution $F_{\mathbf{w}_t}(\mathrm{w})$, then D^M converges to 0 almost surely in the limit when M goes to infinity.

Under null hypothesis that the sample comes from the hypothesized distribution $F_{\mathbf{w}_t}(\mathrm{w})$, $\sqrt{M}D^M$ is converged to Kolmogorov distribution [68].

Decision. For a given significance level γ, the null hypothesis is rejected when $\sqrt{M}D^M > d_\gamma^*$, where d_γ^* is the critical value for γ, $d_\gamma^* = K_\gamma/\sqrt{M}$. By looking up the Kolmogorov distribution table [68], for $\gamma = 0.01$, $K_\gamma = 1.52$.

5.7 Case study

5.7.1 Controller Design for a Auxiliary Transmission Fluid Pump Motor

As introduced in Section 5.2.2, the software design for the auxiliary transmission fluid pump motor controller is illustrated in Figure 12. As system initiated, it stays in *Wait*. Upon receiving a start signal, a PWM signal with DC 15%, system transit to *Shutdown*. With a valid speed command, a PWM signal with DC *SpdCmd*% (*SpdCmd* $\in$ [25.85]), the speed is controlled by the motor control algorithm. The motor stops given a stop signal, a PWM signal with DC 15%.

5.7.1.0.1 ePump System Automaton with Modeled Faults

To verify the safety requirements using the the proposed framework, PB-FDD. FMEA results in Table III are incorporated to the nominal system model, which results again a probabilistic hybrid automaton represented in Figure 14. In $Wait$ and *Standby*, the motor is not spinning and PWM output is sending signals with DC of 87.5% and 47.5%, respectively. In motor control, *spd* in *Alignment* depends on the initial rotor position, and is modeled as a one step noise. After $100ms$, system transits to *Startup*. *Spd* builds fast to reach 90% of the expected RPM with maximum torque in a open loop control. Then, in *Operation*, *Spd* is controlled in a closed loop with speed estimation based on the Back Electromotive Force (BEMF) zero-crossing timing. Gaussian noises are assumed for both the speed evolution in the three modes in motor control and the DC of the PWM out signal across the whole system.

The exceptions listed in Table II are not reported until the system is in *Operation*. The three exceptions are modeled to be triggered by stochastic events. The probabilities p_1, p_2 and p_3 are calculated beforehand, and known to the system automaton. If a exception matures in *Exception*, the system transits back to$Wait$. The exception code is cleared when the system transits to *Shutdown* upon receiving another start signal. $F1$ is the component level fault mode that the exception code cannot be cleared modeled by a discrete mode transitioning from $Wait$. Mode $F2$ is the component level fault that Motor runs at a lower speed (less than 95% of the expected speed) due to low battery or the increased fluid viscosity. $F2$ can be reached from *Startup* or *Operation* depending on the peak motor speed *spd* the pump reaches. Mode $F3$ represents the the situation that the motor speed *spd* oscillates in an amplitude bigger than 5%

System Automaton with Fault Model Incorporated

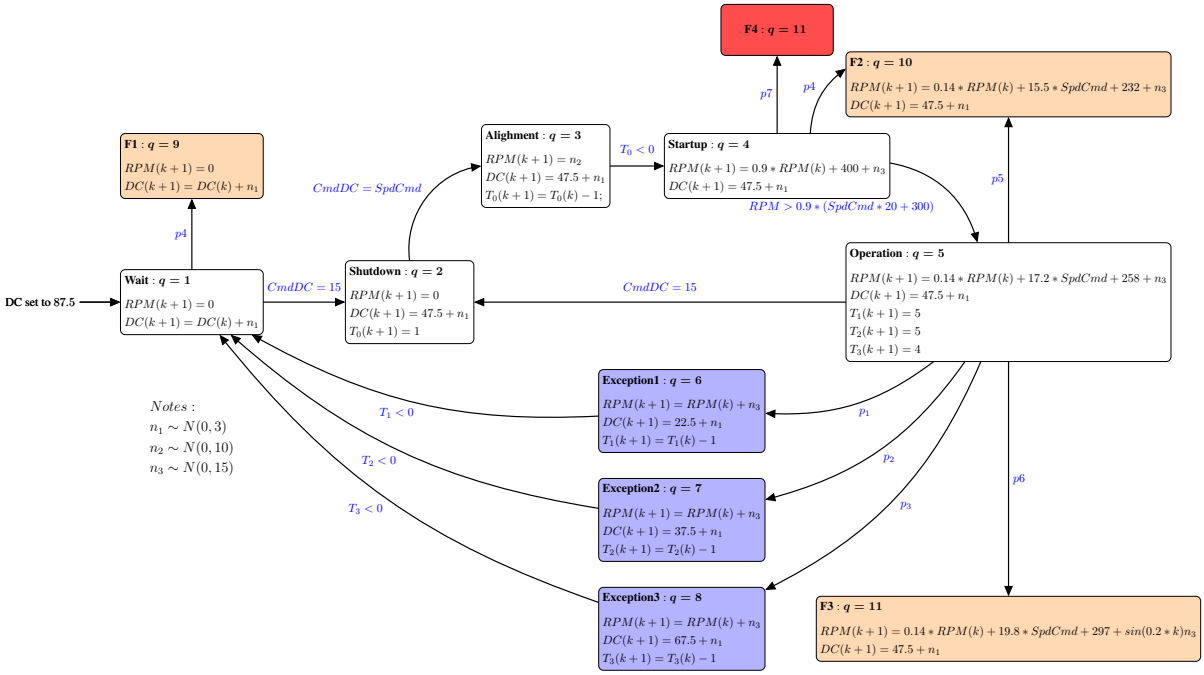

Figure 14: System Automaton with Modeled Faults for ePump Control System

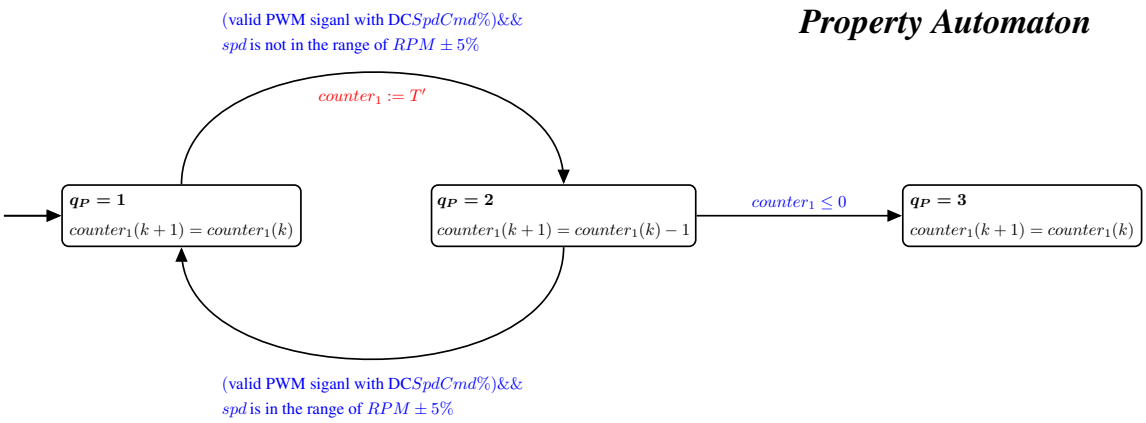

Figure 15: Property Automaton for ePump Control System

of the expected speed. $F3$ can be reached from *Operation* mode. All the three component level faults are also triggered by stochastic event and represented by probabilities, p_4, p_5 and p_6.

The observations y_p and y_d are the motor speed measured by a encoder with white Gaussian noises, and the PWM out DC measured by a digital channel with white Gaussian noises, respectively.

5.7.1.0.2 ePump Property Automaton

The safety requirement to verify is that after a valid PWM speed command with DC $SpdCmd\%$ is received followed by a start signal, the motor is able to run continuously in the speed range $RPM \pm 5\%$, where $RPM = 20 * SpdCmd + 300$ where $SpdCmd \in [25.85]$. This is a liveness property, shown in Figure 15.

The system run is simulated with another component level fault $F4$ unknown to the system model with all the FMEA analysis results incorporated. The fault is modeled as a discrete

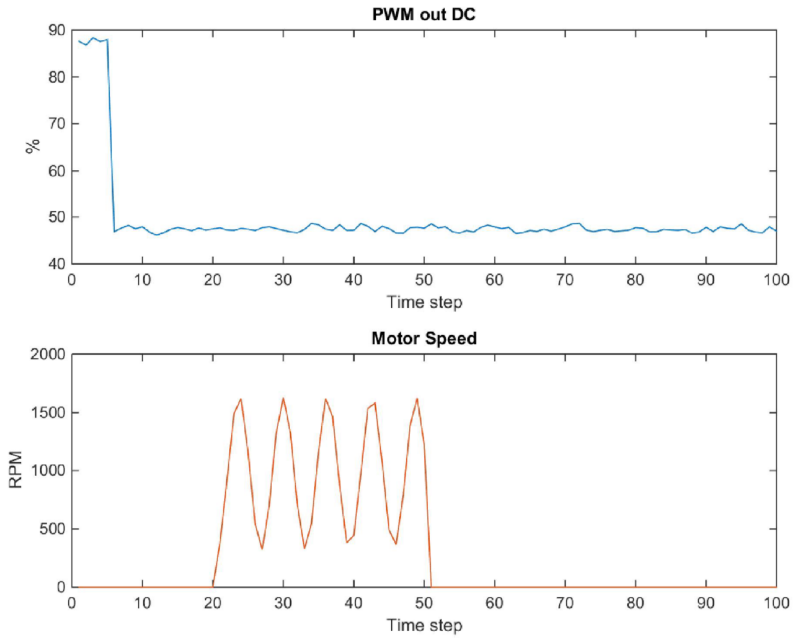

Figure 16: Sample Run for the ePump Control System

mode transitioning from *Startup*. In $F4$, the motor control is reading a faulty sensing of DC bus current due to a ADC channel overflow such that motor control stops driving *spd* to increase to the desired level. Then, the system tries to restart the pump and keep repeating the same startup process. The controller is forced to quit after 3 seconds due to the hardware limit. A sample run with this unknown fault is given in Figure 16. The PWM command is given in a sequence of DC 5% for 5 time steps, 15% for 15 time steps, and 85% for 80 time stamps.

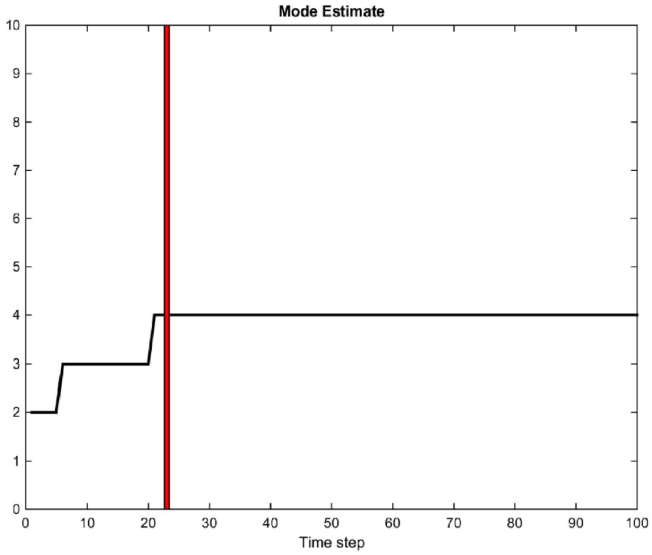

Figure 17: Unmodeled Fault Detection in ePump Control System

Most of the system modes are easily distinguished by PWM out DC and motor speed. But in motor control without any known exception, it always outputs DC 47.5%, and then for an unknown fault, it is difficult to tell from motor speed where an it is originated.

To locate this fault using PB-FDD, the importance weight distribution in SIS step using PF is evaluated at every time step, and two hypothesis tests are performed sequentially.

The result shown in Figure 17 successfully gives the detection of an unmodeled fault when $t = 23$ when the mode estimation stuck at $q = 4$ which is the *Startup* mode.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we proposed a novel framework of fault detection and diagnosis, property-based fault detection and diagnosis (PB-FDD). Inspired by our previous work on runtime monitoring on hybrid systems with respect to a given property, we hierarchically defined faults according to the scope of the requirements. The system level fault is described as the violation of the system requirements which is independent of the system design, and the component level fault is defined as violation of component requirements based on the results from the FMEA process. So the detection of system level faults is essentially another application of the monitoring technique and the diagnosis turns out to be the identification of component level faults which are modeled in each subsystem.

The particle filtering algorithm is the core part of a threshold based monitor [12] This sequential Monte Carlo method subjects to the deterioration of performance when particle inconsistency is detected. We studied two different causes for particle inconsistency, particle propagation depletion and system model inconsistency.

In the thesis, we studied a class of CPSs, concurrent CPSs with N components. In monitoring of such systems, particle inconsistency is caused by particle propagation depletion due to the exponential state explosion introduced by interaction among components. We formally defined

concurrent probabilistic hybrid automata with subsystems and analyzed different types of concurrency in subsystem operation. With notions of local and link variables, we studied a special type of concurrency, locally parallel operation. In this case, we proposed a new particle filtering approach, hierarchical Particle Filter to balance the computation load and the performance of the algorithm.

In PB-FDD where particle filter is adapted in fault detection algorithm, an important improvement of the proposed PB-FDD is the diagnosis of unknown component level faults. The occurrence of an unknown component level faults would inherently lead to particle inconsistency in PF due to system model inconsistency. By analyzing the importance weights for the predicted set of particles, we construct hypothesis testing using importance weights to detect particle inconsistency and further unmodeled component level faults.

A train system with multiple cars equipped with ECP brakes was simulated. Particle propagation depletion happens when cars are braking independently. Experimental results showed the effectiveness of hierarchical particle filter using a small set of particles.

A project in Magna Electronics Inc, controller design for an auxiliary transmission fluid pump motor, is used to validate the hypothesis testing to detect the particle inconsistency, and further to recognize an unmodeled fault.

6.2 Future Work

6.2.1 Output Decoupling Analysis

In the analysis of a cPHA, based on the interaction among subsystems, we have considered different types of concurrency in subsystem operation. In one specific type of concurrency,

locally parallel operation, the subsystem operation can be interpreted as a parallel operation of the local variables in locally input decoupled subsystems followed by a coupled operation of the other local variables and the link variables.

However, this concurrency analysis is based on the dependency of input of each subsystem. It is often the case that the output in a subsystem is not decoupled (the output of the subsystem only depends on the state variables in the same subsystem). In the train example used in this thesis, the model of the link force observation is specified on all the braking subsystems. In Section 4.4.2.3, we discussed scenarios of different types of dependency of output, but it is only focused on how specific types of output dependency affect HPF implementation in locally parallel operation.

Similar to HPF – Case 3 in Section 4.4.2.3, if the observations are divided into two groups,

$$y_t = (y_t^a, y_t^b)$$

and y_t^a and y_t^b are specified on 2 exclusive subsets of s_t, s_t^a and s_t^b, respectively, then y_t^a and y_t^b are conditional independent given the state s_t. The importance weight can be accordingly rewritten as the product of two partial importance weights,

$$\begin{aligned} \mathrm{w}_t^{(j)} &= p(y_t|s_t^{(j)}) = p(y_t^a, y_t^b|s_t^{(j)}) \\ &= p(y_t^a|s_t^{(j)})p(y_t^b|s_t^{(j)}) \\ &= p(y_t^a|s_t^{a\,(j)})p(y_t^b|s_t^{b\,(j)}) \\ &= \mathrm{w}_t^{a\,(j)}\mathrm{w}_t^{b\,(j)} \end{aligned}$$

Then w_t^a may be used to adapt the partial s_t^a of the particles in the predicted set. Therefore, the analysis of the subsystem output dependency can directly connect the partial importance weight to a subset of the state variables, and thus improving the adaptation of the set of particles representing the proposal distribution, $\overline{\mathcal{S}}_t$.

6.2.1.1 Particle Filter Performance Related to Resampling

In this thesis, we studied a phenomenon that affects the performance of particle filter, namely particle inconsistency and also further looked into its causes. By the definition 12, particle inconsistency is defined for set of propagated particles in prediction step.

In the hypothesis testing used to detect particle inconsistency, the assumption made to derive the distribution of importance weight variable $\mathbf{w}_t$ is for the proposal particle set $\overline{\mathcal{S}}_{t-1}$ at time $t-1$, and it is assumed that the resampling step is implemented properly to obtain

$\mathcal{S}_t$, the set of particles representing target distribution. Furthermore, this assumption is made across the whole thesis. However, the second step in particle filter, correction, is implemented by resampling which could also bring error/failure for particle filter. One example is particle impoverishment discussed in Section 5.6.1. In the context of sequential Monte Carlo techniques, another hypothesized cause of particle inconsistency at time t could be *resampling inconsistency* at time $t-1$.

APPENDICES

Appendix A

COPYRIGHT PERMISSIONS

In this appendix, we present the copyright permission for the articles being cited as previously published work in this thesis. The list of articles includes [12] and [71].

SPRINGER NATURE LICENSE
TERMS AND CONDITIONS

Sep 02, 2018

This Agreement between Ms. Yao Feng ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number	4420871309839
License date	Sep 02, 2018
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	Runtime Monitoring of Stochastic Cyber-Physical Systems with Hybrid State
Licensed Content Author	A. Prasad Sistla, Miloš Žefran, Yao Feng
Licensed Content Date	Jan 1, 2012
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	electronic
Portion	Text extract
Number of text extracts	6
Will you be translating?	no
Circulation/distribution	<501
Author of this Springer Nature content	yes
Title	Property-Based Fault Detection and Diagnosis for Safety-Critical Cyber-Physical Systems
Instructor name	Milos Zefran
Institution name	University of Illinois at Chicago
Expected presentation date	Sep 2018
Portions	Section 1, 2, 3, 4, 5 and 6
Requestor Location	Ms. Yao Feng 6-604, Lane 1841, Changyang Rd Shanghai, Shanghai 200090 China Attn: Ms. Yao Feng
Billing Type	Invoice
Billing Address	Ms. Yao Feng 6-604, Lane 1841, Changyang Rd

Shanghai, China 200090
Attn: Ms. Yao Feng

Total 0.00 USD

Terms and Conditions

Springer Nature Terms and Conditions for RightsLink Permissions

Springer Nature Customer Service Centre GmbH (the Licensor) hereby grants you a non-exclusive, world-wide licence to reproduce the material and for the purpose and requirements specified in the attached copy of your order form, and for no other use, subject to the conditions below:

1. The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of this material. However, you should ensure that the material you are requesting is original to the Licensor and does not carry the copyright of another entity (as credited in the published version).

   If the credit line on any part of the material you have requested indicates that it was reprinted or adapted with permission from another source, then you should also seek permission from that source to reuse the material.

2. Where **print only** permission has been granted for a fee, separate permission must be obtained for any additional electronic re-use.

3. Permission granted **free of charge** for material in print is also usually granted for any electronic version of that work, provided that the material is incidental to your work as a whole and that the electronic version is essentially equivalent to, or substitutes for, the print version.

4. A licence for 'post on a website' is valid for 12 months from the licence date. This licence does not cover use of full text articles on websites.

5. Where **'reuse in a dissertation/thesis'** has been selected the following terms apply: Print rights of the final author's accepted manuscript (for clarity, NOT the published version) for up to 100 copies, electronic rights for use only on a personal website or institutional repository as defined by the Sherpa guideline (www.sherpa.ac.uk/romeo/).

6. Permission granted for books and journals is granted for the lifetime of the first edition and does not apply to second and subsequent editions (except where the first edition permission was granted free of charge or for signatories to the STM Permissions Guidelines http://www.stm-assoc.org/copyright-legal-affairs/permissions/permissions-guidelines/), and does not apply for editions in other languages unless additional translation rights have been granted separately in the licence.

7. Rights for additional components such as custom editions and derivatives require additional permission and may be subject to an additional fee. Please apply to Journalpermissions@springernature.com/bookpermissions@springernature.com for these rights.

8. The Licensor's permission must be acknowledged next to the licensed material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract, and must be hyperlinked to the journal/book's homepage. Our required acknowledgement format is in the Appendix below.

9. Use of the material for incidental promotional use, minor editing privileges (this does not include cropping, adapting, omitting material or any other changes that affect the meaning, intention or moral rights of the author) and copies for the disabled are permitted under this licence.

10. Minor adaptations of single figures (changes of format, colour and style) do not require the Licensor's approval. However, the adaptation should be credited as shown in Appendix

below.

Appendix — Acknowledgements:

For Journal Content:
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

For **Advance Online Publication papers:**
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)

For Adaptations/Translations:
Adapted/Translated by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

Note: For any republication from the British Journal of Cancer, the following credit line style applies:

Reprinted/adapted/translated by permission from [**the Licensor**]: on behalf of Cancer Research UK: : [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT]** (year of publication)

For **Advance Online Publication** papers:
Reprinted by permission from The [**the Licensor**]: on behalf of Cancer Research UK: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM])

For Book content:
Reprinted/adapted by permission from [**the Licensor**]: [**Book Publisher** (e.g. Palgrave Macmillan, Springer etc) [**Book Title**] by [**Book author**(s)] [**COPYRIGHT**] (year of publication)

Other Conditions:

Version 1.1

SPRINGER NATURE LICENSE
TERMS AND CONDITIONS

Sep 02, 2018

This Agreement between Ms. Yao Feng ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number	4420880992200
License date	Sep 02, 2018
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	Monitorability of Stochastic Dynamical Systems
Licensed Content Author	A. Prasad Sistla, Miloš Žefran, Yao Feng
Licensed Content Date	Jan 1, 2011
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	electronic
Portion	Text extract
Number of text extracts	5
Will you be translating?	no
Circulation/distribution	<501
Author of this Springer Nature content	yes
Title	Property-Based Fault Detection and Diagnosis for Safety-Critical Cyber-Physical Systems
Instructor name	Milos Zefran
Institution name	University of Illinois at Chicago
Expected presentation date	Sep 2018
Portions	Section 1,2,3,4 and 5
Requestor Location	Ms. Yao Feng 6-604, Lane 1841, Changyang Rd Shanghai, Shanghai 200090 China Attn: Ms. Yao Feng
Billing Type	Invoice
Billing Address	Ms. Yao Feng 6-604, Lane 1841, Changyang Rd Shanghai, China 200090 Attn: Ms. Yao Feng

Total 0.00 USD

Terms and Conditions

Springer Nature Terms and Conditions for RightsLink Permissions

Springer Nature Customer Service Centre GmbH (the Licensor) hereby grants you a non-exclusive, world-wide licence to reproduce the material and for the purpose and requirements specified in the attached copy of your order form, and for no other use, subject to the conditions below:

1. The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of this material. However, you should ensure that the material you are requesting is original to the Licensor and does not carry the copyright of another entity (as credited in the published version).

   If the credit line on any part of the material you have requested indicates that it was reprinted or adapted with permission from another source, then you should also seek permission from that source to reuse the material.

2. Where **print only** permission has been granted for a fee, separate permission must be obtained for any additional electronic re-use.

3. Permission granted **free of charge** for material in print is also usually granted for any electronic version of that work, provided that the material is incidental to your work as a whole and that the electronic version is essentially equivalent to, or substitutes for, the print version.

4. A licence for 'post on a website' is valid for 12 months from the licence date. This licence does not cover use of full text articles on websites.

5. Where **'reuse in a dissertation/thesis'** has been selected the following terms apply: Print rights of the final author's accepted manuscript (for clarity, NOT the published version) for up to 100 copies, electronic rights for use only on a personal website or institutional repository as defined by the Sherpa guideline (www.sherpa.ac.uk/romeo/).

6. Permission granted for books and journals is granted for the lifetime of the first edition and does not apply to second and subsequent editions (except where the first edition permission was granted free of charge or for signatories to the STM Permissions Guidelines http://www.stm-assoc.org/copyright-legal-affairs/permissions/permissions-guidelines/), and does not apply for editions in other languages unless additional translation rights have been granted separately in the licence.

7. Rights for additional components such as custom editions and derivatives require additional permission and may be subject to an additional fee. Please apply to Journalpermissions@springernature.com/bookpermissions@springernature.com for these rights.

8. The Licensor's permission must be acknowledged next to the licensed material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract, and must be hyperlinked to the journal/book's homepage. Our required acknowledgement format is in the Appendix below.

9. Use of the material for incidental promotional use, minor editing privileges (this does not include cropping, adapting, omitting material or any other changes that affect the meaning, intention or moral rights of the author) and copies for the disabled are permitted under this licence.

10. Minor adaptations of single figures (changes of format, colour and style) do not require the Licensor's approval. However, the adaptation should be credited as shown in Appendix below.

Appendix — Acknowledgements:

For Journal Content:
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

For **Advance Online Publication papers:**
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)

For Adaptations/Translations:
Adapted/Translated by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

Note: For any republication from the British Journal of Cancer, the following credit line style applies:

Reprinted/adapted/translated by permission from [**the Licensor**]: on behalf of Cancer Research UK: : [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT]** (year of publication)

For **Advance Online Publication** papers:
Reprinted by permission from The [**the Licensor**]: on behalf of Cancer Research UK: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM])

For Book content:
Reprinted/adapted by permission from [**the Licensor**]: [**Book Publisher** (e.g. Palgrave Macmillan, Springer etc) [**Book Title**] by [**Book author**(s)] [**COPYRIGHT**] (year of publication)

Other Conditions:

Version 1.1

CITED LITERATURE

1. Lee, E.: Cyber Physical Systems: Design Challenges. In Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on, pages 363 –369, May 2008.

2. Cyber-Physical Systems (CPS). `https://www.nsf.gov/pubs/2008/nsf08611/nsf08611.htm`. nsf08611.

3. Alur, R., Courcoubetis, C., Henzinger, T., and Ho, P.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. Hybrid systems, pages 209–229, 1993.

4. Lynch, N., Segala, R., Vaandrager, F., and Weinberg, H.: Hybrid i/o automata. Hybrid Systems III, pages 496–510, 1996.

5. Hespanha, J.: Stochastic Hybrid Systems: Application to Communication Networks. In Hybrid Systems: Computation and Control, eds. R. Alur and G. Pappas, volume 2993 of Lecture Notes in Computer Science, pages 47–56. Springer Berlin / Heidelberg, 2004.

6. Pola, G., Bujorianu, M. L., Lygeros, J., and Benedetto, M. D. D.: Stochastic hybrid models: An overview. In Proc. of the IFAC Conference on Analysis and Design of Hybrid Systems, pages 45–50, 2003.

7. Blom, H. A. and Lygeros, J.: Stochastic hybrid systems. Springer-Verlag Berlin/Heidelberg, 2006.

8. Henzinger, T. A., Kopke, P. W., Puri, A., and Varaiya, P.: What's Decidable about Hybrid Automata? Journal of Computer and System Sciences, 57(1):94–124, August 1998.

9. Kindler, E.: Safety and liveness properties: A survey. Bulletin of the European Association for Theoretical Computer Science, 53:268–272, 1994.

10. Alpern, B. and Schneider, F. B.: Recognizing safety and liveness. Distrib Comput, 2(3):117–126, September 1987.

11. Mukund, M.: Linear-Time Temporal Logic and Büchi Automata. 1997.

12. Sistla, A. P., Žefran, M., and Feng, Y.: Runtime Monitoring of Stochastic Cyber-Physical Systems with Hybrid State. In Runtime Verification, number 7186 in Lecture Notes in Computer Science, pages 276–293. Springer Berlin Heidelberg, January 2012.

13. Thrun, S., Burgard, W., and Fox, D.: Probabilistic Robotics. The MIT Press, August 2005.

14. Carlson, J. and Murphy, R.: Reliability analysis of mobile robots. In IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03, volume 1, pages 274–281 vol.1, 2003.

15. Verma, V., Gordon, G., Simmons, R., and Thrun, S.: Real-time fault diagnosis [robot fault diagnosis]. Robotics Automation Magazine, IEEE, 11(2):56 – 66, June 2004.

16. Gertler, J.: Fault Detection and Diagnosis in Engineering Systems. CRC Press, May 1998. Google-Books-ID: fmPyTbbqKFIC.

17. Isermann, R.: Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance. Springer Science & Business Media, January 2006.

18. Funiak, S. and Williams, B.: Multi-modal particle filtering for hybrid systems with autonomous mode transitions. In Workshop on Principles of Diagnosis, 2003.

19. Koutsoukos, X., Kurien, J., and Zhao, F.: Estimation of distributed hybrid systems using particle filtering methods. In Hybrid Systems: Computation and Control, volume 2623 of Lecture Notes in Computer Science, pages 298–313. Springer, 2003.

20. McIlraith, S., Biswas, G., Clancy, D., and Gupta, V.: Hybrid systems diagnosis. In Hybrid Systems: Computation and Control, volume 1790 of Lecture Notes in Computer Science, pages 282–295. Springer, 2000.

21. Lerner, U., Moses, B., Scott, M., McIlraith, S., and Koller, D.: Monitoring a complex physical system using a hybrid dynamic bayes net. In Proceedings of the 18th Annual Conference on Uncertainty in AI (UAI), pages 301–310, 2002.

22. Balluchi, A., Benvenuti, L., Di Benedetto, M., and Sangiovanni-Vincentelli, A.: Design of observers for hybrid systems. In Hybrid Systems: Computation and Control, volume 2289 of Lecture Notes in Computer Science, pages 59–80. Springer, 2002.

23. Wilcox, C. and Williams, B.: Runtime verification of stochastic, faulty systems. In Runtime Verification, volume 6418 of Lecture Notes in Computer Science, pages 452–459. Springer Berlin / Heidelberg, 2010.

24. Blom, H. and Bloem, E.: Particle filtering for stochastic hybrid systems. In Decision and Control, 2004. CDC. 43rd IEEE Conference on, volume 3, pages 3221 –3226 Vol.3, December 2004.

25. Hofbaur, M. W. and Williams, B. C.: Mode Estimation of Probabilistic Hybrid Systems. In Hybrid Systems: Computation and Control, eds. C. J. Tomlin and M. R. Greenstreet, number 2289 in Lecture Notes in Computer Science, pages 253–266. Springer Berlin Heidelberg, March 2002. DOI: 10.1007/3-540-45873-5_21.

26. Russell, S. J. and Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, second edition, December 2002.

27. d'Amorim, M. and Roşu, G.: Efficient monitoring of ω-languages. In Computer Aided Verification, volume 3576 of Lecture Notes in Computer Science, pages 311–318. Springer, 2005.

28. Sistla, A. P. and Srinivas, A.: Monitoring temporal properties of stochastic systems. In Verification, Model Checking, and Abstract Interpretation, volume 4905 of Lecture Notes in Computer Science, pages 294–308. Springer, 2008.

29. Gondi, K., Patel, Y., and Sistla, A.: Monitoring the full range of ω-regular properties of stochastic systems. In Verification, Model Checking, and Abstract Interpretation, volume 5403 of Lecture Notes in Computer Science, pages 105–119. Springer, 2009.

30. Sammapun, U., Lee, I., and Sokolsky, O.: Rt-mac:runtime monitoring and checking of quantitative and probabilistic properties. In Proc. of 11th IEEE International Conference on Embedded and Real-time Computing Systems and Applications (RTCSA 2005), pages 147–153, 2005.

31. Grunske, L. and Zhang, P.: Monitoring probabilistic properties. In Proceedings of 7th Joint meeting of European Software Engineering Conference and ACM Symposium on Foundations of Software Engineering ESEC-FSE'09, pages 183–192. ACM, 2009.

32. Pnueli, A., Zaks, A., and Zuck, L. D.: Monitoring interfaces for faults. In Proceedings of the 5^{th} Workshop on Runtime Verification (RV'05), 2005. To appear in a special issue of ENTCS.

33. Margaria, T., Sistla, A., Steffen, B., and Zuck, L.: Taming interface specifications. In CONCUR 2005 – Concurrency Theory, volume 3653 of Lecture Notes in Computer Science, pages 548–561. Springer, 2005.

34. Sistla, A. P., Zhou, M., and Zuck, L.: Monitoring off-the-shelf components. In Verification, Model Checking, and Abstract Interpretation, volume 3855 of Lecture Notes in Computer Science, pages 222–236. Springer, 2006.

35. Stoller, S. D., Bartocci, E., Seyster, J., Grosu, R., Havelund, K., Smolka, S. A., and Zadok, E.: Runtime Verification with State Estimation. In Runtime Verification, Lecture Notes in Computer Science, pages 193–207. Springer, Berlin, Heidelberg, September 2011.

36. DeCarlo, R. A., Branicky, M. S., Pettersson, S., and Lennartson, B.: Perspectives and results on the stability and stabilizability of hybrid systems. Proceedings of the IEEE, 88(7):1069–1082, 2000.

37. Alur, R., Henzinger, T. A., Lafferriere, G., and Pappas, G. J.: Discrete abstractions of hybrid systems. Proceedings of the IEEE, 88(7):971–984, 2000.

38. van der Schaft, A. J. and Schumacher, J. M.: An Introduction to Hybrid Dynamical Systems, volume 251 of Lecture Notes in Control and Information Sciences. Springer, 1999.

39. Liberzon, D.: Switching in Systems and Control. Boston, MA, Birkhäuser, 2003.

40. Hu, J., Lygeros, J., and Sastry, S.: Towards a Theory of Stochastic Hybrid Systems. In Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, pages 160–173. Springer, Berlin, Heidelberg, March 2000.

41. Ghosh, M., Arapostathis, A., and Marcus, S.: Ergodic Control of Switching Diffusions. SIAM J. Control Optim., 35(6):1952–1988, November 1997.

42. Li, X., Omotere, O., Qian, L., and Dougherty, E. R.: Review of stochastic hybrid systems with applications in biological systems modeling and analysis. EURASIP Journal on Bioinformatics and Systems Biology, 2017:8, June 2017.

43. Kloetzer, M. and Belta, C.: A fully automated framework for control of linear systems from LTL specifications. In Hybrid Systems: Computation and Control, volume 3927 of Lecture Notes on Computer Science, pages 333–347. Springer, 2006.

44. Kress-Gazit, H., Fainekos, G., and Pappas, G.: Where's Waldo? Sensor-based temporal logic motion planning. In IEEE Int. Conf. on Robotics and Automation, pages 3116–3121, 2007.

45. Pnueli, A.: Verifying liveness properties of reactive systems. In Hybrid and Real-Time Systems, volume 1201 of Lecture Notes on Computer Science. New York, NY, Springer, 1997.

46. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T. A., Ho, P.-H., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical Computer Science, 138(1):3–34, 1995.

47. Xiang, W., Xiao, J., and Iqbal, M. N.: Fault detection for switched nonlinear systems under asynchronous switching. International Journal of Control, 84(8):1362–1376, 2011.

48. Hu, Y. and El-Farra, N. H.: Robust fault detection and monitoring of hybrid process systems with uncertain mode transitions. AIChE Journal, 57(10):2783–2794, 2011.

49. Yang, H., Jiang, B., and Cocquempot, V.: Fault tolerant control design for hybrid systems, volume 397 of Lecture Notes in Control and Information Sciences. Springer Verlag, 2010.

50. Cocquempot, V., Mezyani, T. E., and Staroswiecki, M.: Fault detection and isolation for hybrid systems using structured parity residuals. In 2004 5th Asian Control Conference (IEEE Cat. No.04EX904), volume 2, pages 1204–1212 Vol.2, July 2004.

51. Gadeyne, K., Lefebvre, T., and Bruyninckx, H.: Bayesian Hybrid Model-State Estimation Applied to Simultaneous Contact Formation Recognition and Geometrical Parameter Estimation. The International Journal of Robotics Research, 24(8):615–630, August 2005.

52. Hofbaur, M. and Williams, B.: Hybrid estimation of complex systems. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 34(5):2178 –2191, October 2004.

53. Boers, Y. and Driessen, H.: A multiple model multiple hypothesis filter for Markovian switching systems. Automatica, 41(4):709–716, April 2005.

54. Hanlon, P. D. and Maybeck, P. S.: Multiple-model adaptive estimation using a residual correlation Kalman filter bank. IEEE Transactions on Aerospace and Electronic Systems, 36(2):393–406, April 2000.

55. Doucet, A., de Freitas, N., Murphy, K., and Russell, S.: Rao-blackwellised particle filtering for dynamic Bayesian networks. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence, UAI'00, pages 176–183, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

56. Hutter, F., Dearden, R., Darmstadt, T. U., and Informatik, F.: The gaussian particle filter for diagnosis of non-linear systems. In In Proceedings of the 14th International Conference on Principles of Diagnosis(DX'03, pages 65–70, 2003.

57. Narasimhan, S.: Combining Particle Filters and Consistency-Based Approaches for Monitoring and Diagnosis of Stochastic Hybrid Systems. January 2004.

58. Hahnel, D., Burgard, W., Fox, D., and Thrun, S.: An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), volume 1, pages 206–211 vol.1, October 2003.

59. Freitas, N. d.: Rao-Blackwellised particle filtering for fault diagnosis. In Proceedings, IEEE Aerospace Conference, volume 4, pages 4–1767–4–1772 vol.4, 2002.

60. Cornebise, J., Moulines, r., and Olsson, J.: Adaptive methods for sequential importance sampling with application to state space models. Stat Comput, 18(4):461–480, December 2008.

61. Pitt, M. K. and Shephard, N.: Filtering via Simulation: Auxiliary Particle Filters. Journal of the American Statistical Association, 94(446):590–599, 1999.

62. Fox, D.: Adapting the Sample Size in Particle Filters Through KLD-Sampling. The International Journal of Robotics Research, 22(12):985–1003, December 2003.

63. Hwang, I., Kim, S., Kim, Y., and Seah, C. E.: A survey of fault detection, isolation, and reconfiguration methods. IEEE Transactions on Control Systems Technology, 18(3):636–653, 2010.

64. Frank, P. M. and Ding, X.: Survey of robust residual generation and evaluation methods in observer-based fault detection systems. Journal of Process Control, 7(6):403–424, December 1997.

65. Goodwin, G. C., Graebe, S. F., and Salgado, M. E.: Control System Design. Upper Saddle River, NJ, USA, Prentice Hall PTR, 1st edition, 2000.

66. Antsaklis, P. J., Passino, K. M., and Wang, S. J.: An introduction to autonomous control systems. IEEE Control Systems, 11(4):5–13, June 1991.

67. Bertsekas, D. P. and Tsitsiklis, J. N.: Introduction to Probability, 2nd Edition. Belmont, Massachusetts, Athena Scientific, 2nd edition edition, July 2008.

68. Conover, W. J.: Practical nonparametric statistics. Wiley, September 1980. Google-Books-ID: m54s2puW_5AC.

69. Lehmann, E. L.: Testing statistical hypotheses. Wiley series in probability and mathematical statistics. New York, Wiley, 1986.

70. Daniel, W. W.: Applied nonparametric statistics. The Duxbury advanced series in statistics and decision sciences. Boston, PWS-KENT Pub., 1990.

71. Sistla, A. P., Žefran, M., and Feng, Y.: Monitorability of stochastic dynamical systems. In Proceedings of the 23rd international conference on Computer aided verification, CAV'11, pages 720–736, Berlin, Heidelberg, 2011. Springer-Verlag.

72. Alpern, B. and Schneider, F. B.: Defining Liveness. Technical report, Cornell University, Ithaca, NY, USA, 1984.

73. Papoulis, A. and Pillai, S. U.: Probability, Random Variables and Stochastic Processes. NewYork, McGrawHill, 2002.

74. Vardi, M.: Automatic verification of probabilistic concurrent systems. In 26th annual Symposium on Foundations of Computer Science, pages 327–338. IEEE Computer Society Press, 1985.

75. Cappe, O., Moulines, E., and Riden, T.: Inferencing in Hidden Markov Models. Springer, 2005.

76. Sontag, E.: Nonlinear regulation: The piecewise linear approach. Automatic Control, IEEE Transactions on, 26(2):346–358, 1981.

77. Bemporad, A. and Morari, M.: Control of systems integrating logic, dynamics, and constraints. Automatica, 35:407–428, 1999.

78. Franklin, G., Workman, M., and Powell, D.: Digital control of dynamic systems. Addison-Wesley, 1997.

79. Viswanathan, M. and Kim, M.: Foundations for runtime monitoring of reactive systems - fundamentals of the mac language. In ICTAC, volume 3407 of Lecture Notes in Computer Science, pages 543–556. Springer, 2004.

80. Doucet, A., de Freitas, N., Murphy, K., and Russell, S.: Rao-Blackwellised particle filtering for dynamic Bayesian networks. In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pages 176–183, 2000.

81. Isard, M. and Blake, A.: Condensation–conditional density propagation for visual tracking. International journal of computer vision, 29(1):5–28, 1998.

82. Thrun, S., Fox, D., Burgard, W., and Dellaert, F.: Robust Monte Carlo localization for mobile robots. Artificial Intelligence, 128(1-2):99–141, 2001.

83. Verma, V., Gordon, G., Simmons, R., and Thrun, S.: Real-time fault diagnosis. IEEE Robotics & Automation Magazine, 11(2):56–66, 2004.

84. Doucet, A. and De Freitas, N.: Sequential Monte Carlo methods in practice, volume 1. Springer, 2001.

85. Grisettiyz, G., Stachniss, C., and Burgard, W.: Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 2432–2437, April 2005.

86. Sadhu, S., Bhaumik, S., Doucet, A., and Ghoshal, T. K.: Particle-method-based formulation of risk-sensitive filter. Signal Process., 89(3):314–319, March 2009.

87. Orguner, U. and Gustafsson, F.: Risk-Sensitive Particle Filters for Mitigating Sample Impoverishment. IEEE Transactions on Signal Processing, 56(10):5001–5012, 2008.

88. Yavolovsky, A., Žefran, M., and Sistla, A. P.: Decision-Theoretic Monitoring of Cyber-Physical Systems. In Runtime Verification, Lecture Notes in Computer Science, pages 404–419. Springer, Cham, September 2016.

89. Brogan, W. L.: Modern Control Theory. Englewood Cliffs, N.J, Pearson, 3 edition edition, October 1990.

90. DeCarlo, R. A. and Saeks, R.: Interconnected dynamical systems. New York : Dekker, 1981.

91. Karlsson, R. and Gustafsson, F.: Particle filter for underwater terrain navigation. In 2003 IEEE Workshop on Statistical Signal Processing, pages 526–529, 2003.

92. Shen, C., Brooks, M., and van den Hengel, A.: Augmented particle filtering for efficient visual tracking. In IEEE International Conference on Image Processing, 2005. ICIP 2005, volume 3, pages III–856–9, 2005.

93. Liu, J. S. and Chen, R.: Sequential Monte Carlo Methods for Dynamic Systems. Journal of the American Statistical Association, 93(443):1032–1044, 1998. ArticleType: research-article / Full publication date: Sep., 1998 / Copyright © 1998 American Statistical Association.

94. Hastie, T., Tibshirani, R., and Friedman, J.: The Elements of Statistical Learning - Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer-Verlag New York, second edition edition, 2009.

95. Ding, S.: Model-based Fault Diagnosis Techniques: Design Schemes, Algorithms, and Tools. Springer Science & Business Media, February 2008.

96. Sobhani-Tehrani, E. and Khorasani, K.: Fault Diagnosis of Nonlinear Systems Using a Hybrid Approach. Springer Science & Business Media, June 2009.

97. Ackerson, G. A.: On state estimation in switching environments. In Seventh Symposium on Adaptive Processes, pages 56–56, December 1968.

98. Tugnait, J. K.: Detection and estimation for abruptly changing systems. In 1981 20th IEEE Conference on Decision and Control including the Symposium on Adaptive Processes, pages 1357–1362, December 1981.

99. Li, X.-R. and Bar-Shalom, Y.: Multiple-model estimation with variable structure. IEEE Transactions on Automatic Control, 41(4):478–493, April 1996.

100. Narasimhan, S. and Biswas, G.: Model-Based Diagnosis of Hybrid Systems. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 37(3):348–361, May 2007.

101. Khorasgani, H. and Biswas, G.: Structural Fault Detection and Isolation in Hybrid Systems. IEEE Transactions on Automation Science and Engineering, PP(99):1–15, 2017.

102. Mikulak, R. J., McDermott, R., and Beauregard, M.: The Basics of FMEA, 2nd Edition. CRC Press, December 2008.

103. J1739: Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA) - SAE International.

104. Clarke, E.: Model checking. In Foundations of Software Technology and Theoretical Computer Science, eds. S. Ramesh and G. Sivakumar, volume 1346 of Lecture Notes in Computer Science, pages 54–56. Springer Berlin / Heidelberg, 1997.

105. Gerth, R., Peled, D., Vardi, M. Y., and Wolper, P.: Simple On-the-fly Automatic Verification of Linear Temporal Logic. In Protocol Specification, Testing and Verification XV, IFIP Advances in Information and Communication Technology, pages 3–18. Springer, Boston, MA, June 1995. DOI: 10.1007/978-0-387-34892-6_1.

106. Wolper, P., Vardi, M. Y., and Sistla, A. P.: Reasoning about infinite computation paths. In 24th Annual Symposium on Foundations of Computer Science (sfcs 1983), pages 185–194, November 1983.

107. Vaart, A. W. v. d.: Asymptotic Statistics, October 1998. DOI: 10.1017/CBO9780511802256.

VITA

NAME: Yao Feng

EDUCATION: B.S., Automation, Northeastern University, Shenyang, China, 2008

M.S., Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, Illinois, 2016

Ph.D., Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, Illinois, 2018

HONORS: China Scholarship Council scholarship, 2009

EXPERIENCE: Computer Vision and Robotics Lab, University of Illinois at Chicago, 2010-2017

PUBLICATIONS: Sistla, A. P., Žefran, M., and Feng, Y.: Monitorability of stochastic dynamical systems. In Proceedings of the 23rd international conference on Computer aided verication, CAV'11, pages 720736, Berlin, Heidelberg, 2011. Springer-Verlag.

Sistla, A. P., Žefran, M., and Feng, Y.: Runtime Monitoring of Stochastic Cyber-Physical Systems with Hybrid State. In Runtime Verication, number 7186 in Lecture Notes in Computer Science, pages 276–293. Springer Berlin Heidelberg, January 2012.

Sistla, A.P., Žefran, M., Feng, Y., & Ben, Y.: Timely monitoring of partially observable stochastic systems. HSCC, 2014.