Correct-By-Construction Fault-Tolerant Control

by

Liren Yang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2020

Doctoral Committee:

Associate Professor Necmiye Ozay, Chair
Professor Jessy Grizzle
Professor Stéphane Lafortune
Professor Jing Sun

Liren Yang

yliren@umich.edu

ORCID iD: 0000-0002-7677-8543

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Necmiye Ozay, my committee members Prof. Jessy Grizzle, Prof. Stéphane Lafortune, Prof. Jing Sun, my colleagues in publication Prof. Al-Thaddeus Avestruz, Dr. Matthew Castanier, Glen Chou, Xiaofan Cui, Dr. Amey Karnik, Zexiang Liu, Oscar Mickelin, Dr. Petter Nillson, Prof. Necmiye Ozay, Dr. Bejamin Pence, Dr. Denise Rizzo, Kwesi Rutledge, Yunus Emre Sahin, Dr. Md Tawhid Bin Waez, and my parents Siyao Yang and Shuxia Lyu.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Correct-by-construction control synthesis methods refer to a collection of model-based techniques to algorithmically generate controllers/strategies that make the systems satisfy some formal specifications. Such techniques attract much attention as they provide formal guarantees on the correctness of cyber-physical systems, where corner cases may arise due to the interaction among different modules. The controllers synthesized through such methods, however, may still malfunction due to faults, such as physical component failures and unexpected operating conditions, which lead to a sudden change of the system model. In these cases, we want to guarantee that the performance of the faulty system degrades gracefully, and hence achieve fault tolerance.

This thesis is about 1) incorporating fault detection and detectability analysis algorithms in correct-by-construction control synthesis, 2) formalizing the graceful degradation specification for fault tolerant systems with temporal logic, and 3) developing algorithms to synthesize correct-by-construction controllers that achieve such graceful degradation, with possible delay in the fault detection. In particular, two sets of approaches from the temporal logic planning domain, i.e., abstraction-based synthesis and optimization-based path planning, are considered.

First, for abstraction-based approaches, we propose a recursive algorithm to reduce the fault tolerant controller synthesis problem into multiple small synthesis

problems with simpler specifications. Such recursive reduction leverages the structure of the fault propagation and hence avoids the high complexity of solving the problem monolithically as one general temporal logic game. Furthermore, by exploring the structural properties in the specifications, we show that, even when the fault is detected with delay, the problem can be solved by a similar recursive algorithm without constructing the belief space.

Secondly, optimization-based path planning is considered. The proposed approach leverages the recently developed temporal logic encodings and state-of-art mixed integer programming (MIP) solvers. The novelty of this work is to enhance the open-loop strategy obtained through solving the MIP so that it can react contingently to faults and disturbance.

Finally, the control synthesis techniques developed for discrete state systems is shown to be applicable to continuous states systems. This is demonstrated by fuel cell thermal management application. Particularly, to apply the abstraction-based synthesis methods to complex systems such as the fuel cell thermal management system, structural properties (e.g., mixed monotonicity) of the system are explored and leveraged to ease abstraction computation, and techniques are developed to improve the scalability of synthesis process whenever the system has a large number of control actions.

CHAPTER I

Introduction

1.1 Background and Motivation

Control system design uses a mathematical model of a physical process, usually given by a differential equation $\dot{x} = f(x, u, d)$ (or by difference equations for the discrete-time systems), that describes how the system state x is affected by the control input u and the disturbance d. The term control design refers to searching for control input u under which the dynamical system achieves the desired behavior, e.g., stability, trajectory following, optimizing certain performance metric, etc., regardless of the disturbance d.

Fault-tolerant control is a branch of control theory that consists of a set of techniques to detect and identify possible faults in the control system, and to design controllers to still achieve a desired (possibly degraded) closed-loop behavior with the knowledge of the fault occurrence [87]. Often times, the degradation of the system is captured by a change of a performance metric [14]. Designing such resilient systems that can operate in the presence of failures is crucial in many application domains, especially for safety critical systems like aircraft flight control systems [21, 35], manufacturing systems [68], and automobile systems [5]. The typical control design paradigm used for fault-tolerant control systems design, similar to that used in many

other system design problems, is the "design V" process [37], where the designers iterate between testing and tuning/redesigning the control systems until all the test cases are passed by the designed controller. As the systems being tested for fault tolerance and the tasks these systems should fulfill get more complicated, however, the design-testing procedure becomes harder and more time-consuming. First, the fault-tolerant controller designed against a single performance metric may not be sufficient to lead to a complicated system behavior that is necessary for its safe operation. Moreover, the testing procedure is not complete in the sense that undesired behaviors may still exist under some untested corner cases created by the interactions between different modules in the system. Hence more expressible specification formalisms and more reliable design paradigms are needed.

Correct-by-construction control synthesis methods, on the other hand, can algorithmically generate a controller under which the closed-loop system is guaranteed to satisfy a formally defined specification. Such techniques are helpful to remove bad designs at an early state and avoid the time-consuming "design V" iterations. The term "correct-by-construction" was originally invented by the computer science community and used to generate software with correct logical behavior. In recent years, there has been significant amount of research on using similar ideas to synthesize controllers for cyber physical systems, where both digital software and continuous-state dynamics exist, to accomplish more complicated tasks. Applications include robotic motion planning [32, 62], autonomous driving and cruise control [29, 81, 121], control of aircraft subsystems [74], and building thermal management [41, 82], just to mention a few. Figure 1.1 shows a simple flow diagram of correct-by-construction synthesis methods, where a formally defined system model and a formally defined specification are required. In practice, however, the system dynamics may experi-

ence a sudden change due to component failures, in which case achieving the original specification may not be feasible and the system is expected to satisfy certain relaxed requirements, and hence degrades gracefully. This suggests that more complicated (yet formal) models and specifications are necessary to take the effect of faults into account. To achieve graceful degradation, the controller needs to be reasonably more conservative compared to the one designed for the unfaulty system even when the system is operating healthily, so that the system will not enter a situation where achieving the relaxed specification is not feasible after fault occurrence. Moreover, in practice, the fault can be only detected with a certain amount of delay after its occurrence. Control synthesis problems with such delayed fault detection can be viewed as a special class of partial information games, which are usually solved with belief space construction. This dramatically increases the computational effort required to solve the problem.

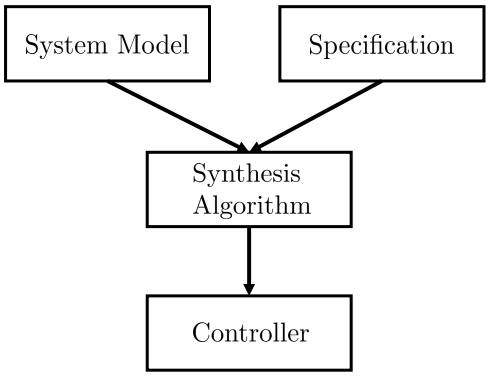

Figure 1.1: Illustration: correct-by-construction control synthesis

Motivated by the issues mentioned above, the correct-by-construction fault-tolerant control problem is studied in this work. In particular, we concern three aspects of the problem. First, we consider model-based detectability analysis that provides a guarantee on finite time fault detection. Such finite time detectability not only enables online fault detection with a receding horizon scheme, but can be also incorporated

in correct-by-construction control synthesis. Second, we consider synthesizing fault-tolerant controllers/strategies that achieve a graceful degradation whenever fault occurs. In particular, the obtained fault-tolerant controllers/strategies should be robust to the detection delay. Last but not least, we concern the computational efficiency of the fault-tolerant control synthesis algorithms. To this end, the structural properties in the continuous system dynamics and the specification are explored for a more tailored synthesis algorithms.

1.2 Literature Review

Fault detection is a key step towards fault tolerant systems and is studied by different communities. For discrete (e.g., software-based) systems, monitoring and run-time verification techniques have been proposed [6, 10, 49, 97, 102]. Similarly, fault detection algorithms have also been studied for continuous-state dynamical systems using ideas from learning, filtering or optimization [38, 53]. In particular, model invalidation [47, 92, 108], a robust system identification technique that is closely related to set membership fault detection [52, 59, 116], can be used for this purpose. Such model invalidation based detection is more suitable for providing a finite detection guarantee rather than an asymptotical one given by, for example, classical residual generation approaches [38]. Recently, a topic called detectability analysis [40, 46, 45, 47, 48] are studied, concerning finding the worst case detection delay via offline analysis. Such worst case detection delay, once found, can be incorporated in correct-by-construction control synthesis. However, due to its worst case nature, the computed detection delay can be overly conservative and needs to be improved.

The concept of correct-by-construction systems was first introduced by computer scientists to algorithmically construct a software that fulfills certain requirements

specified by, e.g., temporal logic [67, 120]. In the past two decades, the idea of correct-by-construction synthesis attracts great attention in the control theory domain. One major difference between the problems considered by the two communities is that, there is a dynamical model that must be respected in the controller synthesis problem. Under this setting, the correct-by-construction design methodology gets significantly extended using the ideas from control theory, especially from model based constraint control. For simple specification like invariance, control barrier functions [4, 55], Hamilton Jacobi (HJ) methods [75, 122] and controlled invariant sets are proposed [13, 98]. For more general linear temporal logic (LTL) specifications, the two most commonly used approaches are abstraction-based synthesis and mixed-integer-linear-programming (MILP) based temporal logic path planning.

In abstraction-based synthesis, a discrete graph structure is constructed to over-approximate the behavior of the underlying continuous-state system (or concrete system). Then two-player game solving techniques developed for such discrete systems are leveraged to synthesize controllers that guarantee the correctness of the continuous state system by the behavior overapproximation relation [110]. Such methods are usually conservative due to the behavior overapproximation relation, and computationally expensive due to the abstraction computation and the high complexity of game solving [11]. While a line of research focuses on reducing the conservatism via refining [41, 51, 63, 82, 85, 103] and post processing [26, 89, 109, 136] the abstraction, other work concentrates on improving the scalability, by i) exploring and exploiting structural properties of the underlying continuous system and of certain LTL fragments, and ii) decomposing the synthesis problem into smaller ones. For example, system structural properties like linearity [60], multiaffine property [61], monotonicity [69], mixed monotonicity [26], incremental stability [42], polynomial dynamics

[89] and symmetry [83] are explored. For structural properties of the LTL specification, mode target game [9], reach-stay-avoid game [82], general reactivity of rank 1 (GR(1)) [86] and other fragments in [80] are considered, and tailored polynomial-time algorithms are developed for these gfragments. For decompositional methods for control synthesis [66, 73, 135], contract design is usually used and similar ideas are also used for controlled invariant set computation of high dimensional systems whenever only safety is of our concern [23, 34, 84]. Finally, a relevant topic is games under partial information, see, for example, [22, 76, 101, 134]. Synthesis with partial information is in general quite harder as it requires an exponential power set construction to keep track of belief states, which exponentially increases the complexity of the controller synthesis.

For MILP-based approaches, unlike the problems solved with the abstraction-based approach, where a winning set (i.e., a set of initial conditions starting from where the specification is achievable) is to be searched, an initial condition is given as part of the problem setup. This allows one to encode LTL/STL with mixed integer linear constraints and search for an open-loop strategy associated with the given initial condition, which achieves the specification [58, 95, 119]. Comparing to abstraction-based synthesis, the MILP-based approaches are less conservative and scale more favorably, but the resulting open-loop controllers tend not to be robust/reactive due to lack of feedback. Recent work aimed at addressing this latter issue includes counter-example-guided methods [96], considering the robust [100] or probabilistic [99] satisfaction of LTL/STL, searching for feedback controllers parametrized by disturbance [39, 104] together with MILP-based trajectory planning. Though, the type of environment uncertainties against which robustness/reactiveness can be achieved by these methods is still limited. Therefore, there is a need for research

that provides reactiveness for different classes of uncertainties, e.g., the potential fault occurrence in the system.

1.3 Main Contribution

The contributions of this thesis are threefold.

In the first part, the detectability analysis is considered together with extra LTL constraints that the correct system must satisfy. In this setting, we combine the idea of LTL monitoring and model invalidation based detection for a less conservative fault detectability analysis. It is shown via a unmanned aerial vehicle consensus problem that the worst case detection delay can be reduced from infinite to finite.

In the second part, the fault-tolerant control problem is defined by introducing a hierarchical control system that captures the changes in the system dynamics caused by unrecoverable faults, and by clarifying the meaning of graceful degradation with LTL. To solve the problem, we look into both abstraction-based synthesis by game solving and MILP based LTL path planning.

For the abstraction-based approach, we consider the fault-tolerant control problem on finite transition systems. By exploiting the fact that the occurrences of faults can be represented by a directed acyclic graph leading to a partial order, we show that the synthesis problem can be decomposed, which leads to an algorithm that avoids solving the problem as a general LTL game and hence a more efficient solution. The proposed approach can be seen as a special case of decompositional synthesis, where the decomposition naturally comes from the structure of the faults. The other half of the work regarding this topic tries to incorporate the detection delay. Such detection delay results in controller not having full information of the state at decision time, and is a special case of partial information. We show how the proposed

algorithm can be modified to handle detection delays, without constructing the belief space. The soundness and completeness of the proposed algorithm is proved for a special fragment of LTL. This work follows the line of research that develops efficient algorithms for specification with special structural properties.

To show that the approach developed for finite transition systems also applies to continuous state systems, we synthesize a correct-by-construction controller for a fuel cell thermal management system with complicated dynamics and requirements. We first develop a control-oriented model for the thermal management system of a fuel cell stack, list the requirements associated with thermal management and formalize them using LTL. Then the existing synthesis tools are extended in two ways to solve the control problem. First, we leverage a structural property of the fuel cell thermal dynamics called weak sign-stability to compute the abstraction more efficiently. Second, we extend the notion of progress group [89] to further reduce the spurious behavior of the obtained abstraction. This work contributes to achieving less conservative and computational more scalable abstractions.

The last part of the thesis presents an optimization-based approach to solve the fault tolerant path planning problem for linear systems to achieve graceful degradation. We propose a hierarchical fault-tolerant controller with a MILP-based trajectory generation at the higher-level and an output-feedback regulator at the lower-level. It is further shown that when the system dynamics are linear, the lower-level regulator design problem reduces to a quasi-convex optimization problem. Our MILP formulation encodes complicated tasks specified with LTL and incorporates reactiveness to faults while being robust to finite detection delays, and hence can be viewed as a move towards obtaining reactive/robust LTL path planning.

Thesis Organization. The rest of the thesis is organized as follows. Chapter II presents preliminary results on system and specification structural properties, which will be used in later chapters. In Chapter III, an improved fault detectability analysis with LTL constraints is presented. Then the fault-tolerant control problems is formally defined in Chapter IV. Chapter V presents an abstraction-based approach to solve the fault-tolerant control problem. The specification properties introduced previously are used to verify the correctness and completeness of the proposed algorithms. In Chapter VI, we define and solve a fuel cell thermal management problem with abstraction-based approach. The system structural properties introduced in Chapter II are used to ease the abstraction computation for this fuel cell system. Chapter VII presents an optimization-based approach to solve the fault-tolerant path planning problem. Finally, the thesis is concluded by Chapter VIII.

Acknowledgment of previous publications. Most of the results in this thesis have previously appeared as published works or technical reports. For Chapter II, the relevant publications are [126, 127, 131, 128]; for Chapter III [130]; for Chapter V [128, 133]; for Chapter VI, [124, 125]; and for Chapter VII, [132]. Other publications [24, 123, 129] developed during the PhD study are also related to correct-by-construction control synthesis but are not included in this thesis.

CHAPTER II

Preliminaries and Structural Properties

To achieve more scalable control synthesis, it is important to understand and exploit the structural properties of the system dynamics and the desired closed-loop system specifications. By structural properties of the system dynamics, we mean the properties of the vector field f of the system described by the differential equation $\dot{x} = f(x, u, d)$; while the structural properties of the specifications refer to some properties of the LTL formula φ that is used to specify the system's desired behavior. Such structure properties, if exist, can be leveraged by the synthesis algorithms and improve the algorithms' scalability.

In this chapter, we present preliminary results on the structural properties of the systems and the specifications considered in this work. A system property called mixed monotonicity is explored, and is shown to be a fairly general property rather than a useful structural property. However, mixed monotonicity provides a nice way to view certain structural properties that can be leveraged by synthesis algorithms. In particular, we introduce a system structural property called weak sign-stability and analyze its usefulness by viewing it as a special case of mixed monotonicity. It is shown that the efficient abstraction computation methods developed for sign-stable systems can be extended to weakly sign-stable systems, e.g., the fuel cell thermal

management system considered in Chapter VI. We then introduce one novel specification structural property called absolutely decomposable property, which leads to a simplified controller synthesis process for the fault tolerant problem, as will be shown in Chapter V. An existing concept called suffix-closed property is also introduced for the same purpose.

Chapter overview. In Section 2.1.1, mixed monotonicity is introduced and it is proved that every function in $\mathbb{R}^n$ is mixed monotone under element-wise order. In Section 2.1.2, we then introduce a structural property called weak sign-stability and show its usefulness in terms of function image approximation. In Section 2.2.1, LTL is introduced as a preliminary, and then in Section 2.2.2 and Section 2.2.3 a novel specification structural property called absolutely decomposable property and an existing specification property called suffix-closedness are introduced respectively.

Related work. Mixed Monotonicity is previously studied in [25, 107] to analyze global stability of nonlinear systems, and recently attracts certain attention in correct-by-construction control community [26, 28, 33]. Although the concept of mixed monotonicity is general, the works mentioned above only focus on a special class of systems called sign-stable systems, whose vector field can be easily verified mixed monotone. The weak sign-stability is a generalization of sign-stability, and extends the computational approaches developed for sign-stable systems to a broader class of systems. See, for example, [70, 71, 72] where the weak sign-stability extension is used. In particular, the abstraction computation technique induced by weak sign-stability can be also seen as a way to bound the Lipschiz constants of the vector field f, and similar ideas appeared in [136] where they are used to compute

abstraction for systems without any stability assumptions.

The concepts of absolute liveness [106] and suffix-closedness [18] are not new, but they are not leveraged in the context of control synthesis to the best knowledge of the author. In the literature, there are also many works studying efficient synthesis algorithms for specific fragements of LTL formulas, such as reach-stay-avoid game [82], mode-target game [8] and GR(1) [86]. The properties presented here are at a more abstract level but the purpose is also efficiency. That is, by studying these properties, we wish to understand when the later proposed efficient fault-tolerant control synthesis algorithms provide sound and complete solutions, especially in the existence of detection delays.

2.1 Structural Properties of Systems

In this part, a system structural property called weak sign-stability is introduced. We analyze the usefulness of this property by viewing it as a special case of mixed monotonicity.

2.1.1 Mixed Monotonicity

We start with introducing the concept of mixed monotonicity. Intuitively, a function is mixed monotone if it can be decomposed into a monotonically increasing part and a monotonically decreasing part. From the definition, it is not immediately clear if this is a general property or a very restrictive structural property. The main result is to show that every function in $\mathbb{R}^n$ is mixed monotone by implicitly constructing its decomposition as a solution to some optimization problems. In particular, this decomposition is tight in the sense that it provides a potential way to tightly approximate the function images with a hyper box. However, this decomposition cannot be

evaluated efficiently unless the function is sign-stable. Therefore mixed monotonicity is not structural property, but it provide a way of viewing functions nonetheless.

Notations: Let $\mathbb{R}^n$ be n-dimensional Euclidean space. We use lower case letter $x \in \mathbb{R}^n$ to denote a n-dimensional vector, and use x_i to denote the i^{th} component of a vector x. When necessary, we will use superscript, e.g., $x^{[i]}$, $x^{[j]}$ to distinguish different vectors (bracket is used to distinguish from the power and the higher order derivatives). We use $\geq$ to denote the element wise order on $\mathbb{R}^n$, i.e., for $x, y \in \mathbb{R}^n$, $x \geq y$ if and only if (iff) $x_i \geq y_i$ for all $i = 1, 2, \ldots, n$. For $\underline{x}, \overline{x} \in \mathbb{R}^n$ such that $\underline{x} \leq \overline{x}$, we denote the hyperinterval $\{x \in \mathbb{R}^n \mid \underline{x} \leq x \leq \overline{x}\}$ by $[\underline{x}, \overline{x}]$.

Definition 1. *(Mixed Monotone Function)* A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is *mixed monotone* if there exists $g : \mathbb{R}^{2n} \to \mathbb{R}^m$ satisfying the following:

1. $g(x, x) = f(x)$;
2. $x \geq x' \Rightarrow g(x, y) \geq g(x', y)$;
3. $y \geq y' \Rightarrow g(x, y) \leq g(x, y')$.

A function g satisfying the above conditions is called a *decomposition function* of f.

The following theorem allows us to approximate the values of a mixed monotone function in some region, using its decomposition function.

Propoition 1. (Theorem 1 in [26]) Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a mixed monotone function and let g be one of its decomposition functions, then

$$\{f(x) \mid x \in [\underline{x}, \overline{x}]\} \subseteq [g(\underline{x}, \overline{x}), g(\overline{x}, \underline{x})]. \tag{2.1}$$

It should also be noticed that decomposition function may not be unique. To see this, consider a simple example where $f(x) = 1$. Clearly both $g(x, y) = 1$ and $g(x, y) = x/y$ are decomposition functions of f. As discussed above in Proposition 1, a decomposition function g can be used to approximate the function value of

f. Motivated by the use of decomposition functions to tightly over approximate $\{f(x) \mid x \in [\underline{x}, \overline{x}]\}$ with a hyperinterval, we introduce the following definition of tightness of a decomposition function.

Definition 2. *(Tight Decomposition)* Let f be a mixed monotone function and g be a decomposition of f. Decomposition function g is called *tight* if for all $\underline{x}, \overline{x} \in \mathbb{R}^n$ s.t. $\underline{x} \leq \overline{x}$, $[g(\underline{x}, \overline{x}), g(\overline{x}, \underline{x})]$ is the smallest (in set inclusion sense) hyperinterval that contains $\{f(x) \mid x \in [\underline{x}, \overline{x}]\}$. That is $[g(\underline{x}, \overline{x}), g(\overline{x}, \underline{x})] = [\inf_{\xi \in [\underline{x}, \overline{x}]} f(\xi), \sup_{\xi \in [\underline{x}, \overline{x}]} f(\xi)]$.

The key result in this part is that every function whose extreme values are well defined has a tight decomposition. To prove this, we introduce the following notation: for $x, y \in \mathbb{R}$ and $h : \mathbb{R} \to \mathbb{R}$, define

$$\text{opt}_{\xi}^{(x,y)} h(\xi) = \begin{cases} \inf_{\xi \in [x,y]} h(\xi) & \text{if } x \leq y \\ \sup_{\xi \in [y,x]} h(\xi) & \text{if } x > y \end{cases}. \tag{2.2}$$

The following simple facts regarding the opt operator are useful in later proofs.

Lemma 1. $\text{opt}_{\xi}^{(x,y)} h(\xi)$ is monotonically increasing in x and monotonically decreasing in y, that is, $x \geq x' \Rightarrow \text{opt}_{\xi}^{(x,y)} h(\xi) \geq \text{opt}_{\xi}^{(x',y)} h(\xi)$ and $y \geq y' \Rightarrow \text{opt}_{\xi}^{(x,y)} h(\xi) \leq \text{opt}_{\xi}^{(x,y')} h(\xi)$.

Proof. We prove $x \geq x' \Rightarrow \text{opt}_{\xi}^{(x,y)} h(\xi) \geq \text{opt}_{\xi}^{(x',y)} h(\xi)$ in the following three cases respectively:

(i) $y \geq x \geq x'$: $\text{opt}_{\xi}^{(x,y)} h(\xi) = \inf_{\xi \in [x,y]} h(\xi) \geq \inf_{\xi \in [x',y]} h(\xi) = \text{opt}_{\xi}^{(x',y)} h(\xi)$;

(ii) $x \geq y \geq x'$: $\text{opt}_{\xi}^{(x,y)} h(\xi) = \sup_{\xi \in [y,x]} h(\xi) \geq h(y) \geq \inf_{\xi \in [x',y]} h(\xi) = \text{opt}_{\xi}^{(x',y)} h(\xi)$;

(iii) $x \geq x' \geq y$: $\text{opt}_{\xi}^{(x,y)} h(\xi) = \sup_{\xi \in [y,x]} h(\xi) \geq \sup_{\xi \in [y,x']} h(\xi) = \text{opt}_{\xi}^{(x',y)} h(\xi)$.

The proof for $y \geq y' \Rightarrow \text{opt}_{\xi}^{(x,y)} h(\xi) \leq \text{opt}_{\xi}^{(x,y')} h(\xi)$ is similar. □

Lemma 2. *Let $\overline{h} : \mathbb{R} \to \mathbb{R}$ and $\underline{h} : \mathbb{R} \to \mathbb{R}$ be such that $\overline{h} \geq \underline{h}$ for all $\xi \in \mathbb{R}$, then $opt_{\xi}^{(x,y)}\overline{h}(\xi) \geq opt_{\xi}^{(x,y)}\underline{h}(\xi)$ for all $x, y \in \mathbb{R}$.*

Proof. This should be clear by the definition of opt. □

With above lemmas, we can prove the following result.

Theorem 1. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be such that $\text{opt}_{\xi_i}^{(x_i,y_i)} f(\xi_i)$ is well defined[1], then the following $g : \mathbb{R}^{2n} \to \mathbb{R}^m$ defined element-wise by

$$g_j(x, y) = \text{opt}_{\xi_1}^{(x_1,y_1)} \text{opt}_{\xi_2}^{(x_2,y_2)} \dots \text{opt}_{\xi_n}^{(x_n,y_n)} f_j(\xi), \quad j = 1, 2, \dots, m \tag{2.3}$$

is a tight decomposition function of f.

Proof. We first prove that g is indeed a decomposition function of f.

1. Clearly, $g(x, x) = f(x)$ by definition.
2. To show that $x \geq x' \Rightarrow g(x, y) \geq g(x', y)$, it is sufficient to show that this is true for a simple case where x and x' differs by only one element, i.e., $x_i \geq x'_i$ and $x_j = x'_j$ for $j \neq i$. For general case, let $x = x^0 \geq x^1 \geq x^2 \geq \cdots \geq x^n = x'$, where x^i and x^{i-1} has exactly the same coordinates except for the i^{th} position. Then applying the result for the simple case for n times yields the desired result for the general case.

   Let x and x' be such that $x_i \geq x'_i$ and $x_j = x'_j$ for $j \neq i$, and define $h^{(x,y)}(\xi_1, \xi_2, \dots, \xi_i) := \text{opt}_{\xi_{i+1}}^{(x_{i+1},y_{i+1})} \dots \text{opt}_{\xi_n}^{(x_n,y_n)} f(\xi)$. Since $x_j = x'_j$ for $j \neq i$, $h^{(x,y)}(\xi_1, \xi_2, \dots, \xi_i) = h^{(x',y)}(\xi_1, \xi_2, \dots, \xi_i)$ and we will use h to denote the function in what follows for

[1]For reachable set computation, it makes sense to assume that f is bounded on any bounded set so that the system $x_{k+1} = f(x_k)$ is forward complete. With such assumption, $\text{opt}_{\xi_i}^{(x_i,y_i)} f(\xi_i)$ is always well defined. However, if we only want to talk about mixed monotonicity of the function f, it is enough to assume that the domain of f is $\mathbb{R}^n$.

simplicity. With this notation, $g(x,y)$ and $g(x',y)$ can be rewritten as

$$g(x,y) = \tag{2.4}$$
$$\mathrm{opt}_{\xi_1}^{(x_1,y_1)}\mathrm{opt}_{\xi_2}^{(x_2,y_2)}\ldots\underbrace{\mathrm{opt}_{\xi_i}^{(x_i,y_i)}h(\xi_1,\xi_2,\ldots,\xi_i)}_{=:\overline{h}(\xi_1,\xi_2,\ldots,\xi_{i-1})},$$

$$g(x',y) = \tag{2.5}$$
$$\mathrm{opt}_{\xi_1}^{(x_1,y_1)}\mathrm{opt}_{\xi_2}^{(x_2,y_2)}\ldots\underbrace{\mathrm{opt}_{\xi_i}^{(x'_i,y_i)}h(\xi_1,\xi_2,\ldots,\xi_i)}_{=:\underline{h}(\xi_1,\xi_2,\ldots,\xi_{i-1})}.$$

Since $x_i \geq x'_i$, by Lemma 1, we know that for all $\xi_1,\xi_2,\ldots\xi_{i-1}$:

$$\overline{h}(\xi_1,\xi_2,\ldots,\xi_{i-1}) \geq \underline{h}(\xi_1,\xi_2,\ldots,\xi_{i-1}). \tag{2.6}$$

Applying Lemma 2 for $i-1$ times leads to $g(x,y) \geq g(x',y)$.

3. Proving that $y \geq y' \Rightarrow g(x,y) \leq g(x,y')$ is similar as bullet 2.

This hence proves that g is a decomposition function of f. Next, we show that g is a tight decomposition. To see this, let $\underline{x},\overline{x} \in \mathbb{R}^n$ to be such that $\underline{x} \leq \overline{x}$. Since $\underline{x}_i \leq \overline{x}_i$ for all i, by definition of g in Eq. (2.3) and Eq. (2.2), we have

$$g(\underline{x},\overline{x}) = \inf_{\xi_1\in[\underline{x}_1,\overline{x}_1]}\inf_{\xi_2\in[\underline{x}_2,\overline{x}_2]}\ldots\inf_{\xi_n\in[\underline{x}_n,\overline{x}_n]} f(\xi_1,\xi_2,\ldots,\xi_n)$$
$$= \inf_{\xi\in[\underline{x},\overline{x}]} f(\xi), \tag{2.7}$$

$$g(\overline{x},\underline{x}) = \sup_{\xi_1\in[\underline{x}_1,\overline{x}_1]}\sup_{\xi_2\in[\underline{x}_2,\overline{x}_2]}\ldots\sup_{\xi_n\in[\underline{x}_n,\overline{x}_n]} f(\xi_1,\xi_2,\ldots,\xi_n)$$
$$= \sup_{\xi\in[\underline{x},\overline{x}]} f(\xi), \tag{2.8}$$

and this shows that g is a tight decomposition function. □

Corollary 1. In general, a mixed monotone function f may not have a unique tight decomposition function.

Proof. Note that the proof of Theorem 1 does not depend on the fact that g is constructed with $\mathrm{opt}_{\xi_i}^{(x_i,y_i)}$ being arranged in an ascending order of i. Therefore one

can rearrange the $\text{opt}_{\xi_i}^{(x_i,y_i)}$ operators and this does not change the fact the resulting g is still a tight decomposition, yet it is well know that g may be different in general after such a rearrangement. For example, let $f : [0,2] \times [0,2] \to \mathbb{R}^2$ be such that

$$f_1(x) = \begin{cases} 0 & \text{if } x \in [0,1) \times [0,1] \cup [1,2] \times (1,2] \\ 1 & \text{otherwise} \end{cases}$$

$$f_2(x) = 0 \ \forall x \in [0,2] \times [0,2]. \tag{2.9}$$

Consider candidate decomposition function g and g' where

$$g_1(x,y) = \text{opt}_{\xi_1}^{(x_1,y_1)} \text{opt}_{\xi_2}^{(x_2,y_2)} f_1(\xi_1, \xi_2), \tag{2.10}$$

$$g_1'(x,y) = \text{opt}_{\xi_2}^{(x_2,y_2)} \text{opt}_{\xi_1}^{(x_1,y_1)} f_1(\xi_1, \xi_2), \tag{2.11}$$

and $g_2(x,y) = g_2'(x,y) = 0$. By Theorem 1, both g and g' are tight decomposition functions of f. However, at point $x = [2,0]^T$ and $y = [0,2]^T$, it can be verified that $g_1(x,y) = 1$ while $g_1'(x,y) = 0$. Hence $g \neq g'$ and we have two different tight decomposition functions of f. □

Remark 1. Note that Corollary 1 is not surprising because decomposition functions are defined on the space of (x,y), while the tightness of a decomposition function g only depends on its value $g(x,y)$ on the set $S := \{(x,y) \mid x, y \text{ are comparable}\}$. Therefore, tight decomposition functions are not unique on the entire space of (x,y), although they do coincide with each other on set S.

The main purpose of Theorem 1 is to show that for any function on $\mathbb{R}^n$ whose supremum and infimum are well-defined, there exists a tight decomposition function, so all these functions are mixed monotone. However, this is an existential result rather than a computational one. Indeed, the tight decomposition function defined

above is not directly useful in the image set approximation as its construction involves computing the infimum and supremum of the function f, which is already equivalent to solving for the extreme coordinates of the image set, and this defeats the purpose of constructing the decomposition function (i.e., approximating the image of function f using Porposition 1.

2.1.2 Weak Sign-stability from the View of Mixed Monotonicity

Although mixed monotonicity is A structural property called weak sign-stability is introduced. We analyze the usefulness of weak sign-stability by viewing it as a special case of mixed monotonicity.

Definition 3. A continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called sign-stable on set $X \subseteq \mathbb{R}^n$ if either $\frac{\partial f_j}{\partial x_i} \geq 0$ for all x or $\frac{\partial f_j}{\partial x_i} \leq 0$ for all x.

Propoition 2. *(Proposition 1 in [26])* A sign-stable functions f has a tight decomposition function in the following form:

$$g_j(x, y) = f_j(z), \quad j = 1, 2, \ldots, m, \tag{2.12}$$

where

$$z_i = \begin{cases} x_i & \text{if } \frac{\partial f_j}{\partial x_i} \geq 0 \quad \forall x \\ y_i & \text{if } \frac{\partial f_j}{\partial x_i} \leq 0 \quad \forall x \end{cases}. \tag{2.13}$$

A notable feature of the decomposition function g defined by Eq. (2.13) is that it can be constructed directly from the expression of function f. We call such decomposition function *evaluable*. By Proposition 1, if g is evaluable, one can easily approximation the image of sign-stable function f when x takes value from a hyper box, by evaluating its decomposition function g. On the contrary, the decomposition

function g in Theorem 1, which is implicitly defined as the solution of an optimization problem, cannot be constructed directly from the expression of f and hence is not evaluable[2].

Definition 4. A differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$ is called weakly sign-stable on a set $X \subseteq \mathbb{R}^n$ if

$$\frac{\partial f_i}{\partial x_j}(x) \in (a_{ij}, b_{ij}), \forall x \in X, \tag{2.14}$$

where $a_{ij}, b_{ij} \in \overline{\mathbb{R}}$, satisfying $a_{ij} < b_{ij}$ but $(a_{ij}, b_{ij}) \neq (-\infty, \infty)$.

Theorem 2. Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a weakly sign-stable function on X, f has an evaluable decomposition function.

Proof. We prove Theorem 2 by constructing a decomposition function for f, then f is mixed monotone by definition.

By assumption $\frac{\partial f_i}{\partial x_j}(x) \in (a_{ij}, b_{ij})$ for all $x \in X$, the interval (a_{ij}, b_{ij}) must satisfy at least one of the following four cases:

case 1:	sign-stable	positive	$a_{ij} \geq 0$
case 2:	sign-unstable	"positive"	$a_{ij} \leq 0$, $b_{ij} \geq 0$, $\lvert a_{ij}\rvert \leq \lvert b_{ij}\rvert$
case 3:	sign-unstable	"negative"	$a_{ij} \leq 0$, $b_{ij} \geq 0$, $\lvert a_{ij}\rvert \geq \lvert b_{ij}\rvert$
case 4:	sign-stable	negative	$b_{ij} \leq 0$.

According to the above cases, define $g : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ as

$$\forall i \in 1 \ldots, m :$$

$$g_i(x, y) = f_i(z) + (\boldsymbol{\alpha}_i - \boldsymbol{\beta}_i)^T (x - y), \tag{2.15}$$

[2]Mathematically, it is hard to define evaluability because it is a property characterized by the computational effort rather than a qualitative criteria.

where $z = [z_1, \ldots, z_n]^T$, $\boldsymbol{\alpha}_i = [\alpha_{i1}, \ldots, \alpha_{in}]^T$, $\boldsymbol{\beta}_i = [\beta_{i1}, \ldots, \beta_{in}]^T$ are n-vectors defined as follows

$$z_j = \begin{cases} x_j \text{ case 1, 2} \\ y_j \text{ case 3, 4} \end{cases}, \tag{2.16}$$

$$\alpha_{ij} = \begin{cases} 0 & \text{case 1, 3, 4} \\ |a_{ij}| + \epsilon & \text{case 2} \end{cases}, \tag{2.17}$$

$$\beta_{ij} = \begin{cases} 0 & \text{case 1, 2, 4} \\ -|b_{ij}| - \epsilon & \text{case 3} \end{cases}, \tag{2.18}$$

where ϵ is a small positive number.

Next we show that g is a decomposition function of f.

1. Obviously $g(x, x) = f(x)$ by equations (2.15) and (2.16).
2. $x_1 \geq x_2 \Rightarrow g(x_1, y) \geq g(x_2, y)$ because

$$\begin{aligned} \forall i : \frac{\partial g_i}{\partial x_j} =& \sum_{k=1}^{n} \frac{\partial f_i}{\partial z_k} \frac{\partial z_k}{\partial x_j} + (\alpha_{ij} - \beta_{ij}) \\ =& \frac{\partial f_i}{\partial z_j} \frac{\partial z_j}{\partial x_j} + (\alpha_{ij} - \beta_{ij}) \\ =& \begin{cases} \frac{\partial f_i}{\partial x_j} & \text{case 1} \\ \frac{\partial f_i}{\partial x_j} + |a_{ij}| + \epsilon & \text{case 2} \\ |b_{ij}| + \epsilon & \text{case 3} \\ 0 & \text{case 4} \end{cases} \\ \geq& 0. \end{aligned} \tag{2.19}$$

3. $y_1 \geq y_2 \Rightarrow g(x, y_1) \leq g(x, y_2)$ because

$$\begin{aligned}
\forall i : \frac{\partial g_i}{\partial y_j} =& \sum_{k=1}^{n} \frac{\partial f_i}{\partial z_k} \frac{\partial z_k}{\partial y_j} - (\alpha_{ij} - \beta_{ij}) \\
=& \frac{\partial f_i}{\partial z_j} \frac{\partial z_j}{\partial y_j} - (\alpha_{ij} - \beta_{ij}) \\
=& \begin{cases} 0 & \text{case 1} \\ -|a_{ij}| - \epsilon & \text{case 2} \\ \frac{\partial f_i}{\partial y_j} - |b_{ij}| - \epsilon & \text{case 3} \\ \frac{\partial f_i}{\partial y_j} & \text{case 4} \end{cases} \\
\leq& 0. \qquad (2.20)
\end{aligned}$$

It follows from definition 1 that g is a decomposition function of f and hence Theorem 2 is proved. □

We now discuss some implications of this result. By Theorem 2, all differentiable functions with continuous partial derivatives are mixed monotone on a compact set, because the partial derivatives are bounded on the compact set, and hence satisfy the hypothesis of Theorem 2.

Theorem 2 is a natural extension of the result in [26], which only handles the case with sign-stable partial derivatives. The idea here is to use linear terms to create additional offset to overcome the sign-unstable partial derivatives, which leads to a decomposition. These linear terms are chosen to be as small as possible so that the decomposition function constructed by Theorem 2 gives a tighter approximation when applying Proposition 1[3]. In the case where all the partial derivatives $\frac{\partial f_i}{\partial x_j}$ are sign-stable, the decomposition function constructed here gives a tight approximation in Proposition 1, that is, the inequality in equation (2.1) reduces to equality

[3]The proof of Theorem 2 would still go through if we combine case 2 and case 3, but we can get smaller coefficients in front of the linear term by treating these two cases separately.

at some $x \in X$ [26]. However this is not true when there are sign-unstable partial derivatives. Thus in general the approximation given by Proposition 1 might be conservative when using the decomposition function constructed in Theorem 2. However, one can reduce such conservatism by dividing region X into smaller subregions and applying the same approximation on each subregion. Then the extremum function value over region X can be obtained by combining the extremum function values on those subregions. This divide-and-conquer approach, of course, requires more computational effort because one needs to approximate the ranges of sign-unstable partial derivatives on each subregion.

Note that the construction of the decomposition function requires to approximate the ranges of the sign-unstable partial derivatives. Therefore, Theorem 2 together with Proposition 1 "shift" the difficulty of approximating the function value of f into approximating its partial derivatives $\frac{\partial f_i}{\partial x_j}$. By doing such, the difficulty may not be reduced in general. However, in many control applications, the considered systems including thermal systems [124] and traffic networks [28], are naturally mixed monotone. If one can approximate the partial derivatives of system flow once and for all and prove its mixed monotonicity, such properties can be used to simplify the system analysis and design techniques.

2.2 Structural Properties of Specifications

In this part, we introduce LTL as the specification formalism and two structural properties of specifications, one called absolutely decomposable property and one called suffix-closed property, that will be useful to analyze the fault-tolerant control synthesis algorithms proposed in the following sections. The concept of absolute safety is novel and some useful esults are established.

Notations: Let S be any set, lowercase letters (e.g., x) are used for denoting a point from set S, bold font lowercase letters are used for finite sequences of points (e.g., $\mathbf{x}$), and blackboard bold font lowercase letters are used for infinite sequences (also called ω-words) of points (e.g., $\mathbb{x}$). We use S^* to denote the set of all finite sequence over set S, and S^ω to denote the set of all ω-words over S. By convention, let $\mathbf{x}(i)$ (or $\mathbb{x}(i)$, respectively) be the i^{th} element in the sequence $\mathbf{x}$ (or $\mathbb{x}$, respectively), and let $\mathbb{x}_i = \mathbb{x}(i)\mathbb{x}(i+1)\mathbb{x}(i+2)\dots$ be the sub-sequence starting from the i^{th} position. We call $\mathbb{x}_i$ a suffix of $\mathbb{x}$, and call $x(1)\cdots x(n)$ a prefix of $\mathbb{x}$ (or a prefix of finite sequence $\mathbf{x}$ if applicable). For a set $P \subseteq (S^\omega)$, we use $pref(P)$ (resp. $suff(P)$) to denote the set of prefixes (resp. suffixes) of ω-words from P.

2.2.1 Formal Specifications in LTL

We use LTL to specify the desired closed-loop system behavior. In what follows we briefly introduce the syntax and the semantics of LTL, and refer the reader to [7] for more details.

1) Syntax: Let AP be a set of atomic propositions, i.e., a set of statements on system and environment variables whose truth values can be determined by checking whether the associated variables are within given sets, the syntax of LTL formulas over AP is given by

$$\varphi ::= \pi \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \tag{2.21}$$

where $\pi \in AP$. Logical operators $\neg$ and $\vee$ correspond to negation and disjunction in boolean logic, while temporal operators $\bigcirc$ and $\mathcal{U}$ are called "next" and "until" operators respectively. With the grammar given in Eq. (2.21), we define the other propositional and temporal logic operators as follows:

- conjunction: $\varphi_1 \wedge \varphi_2 := \neg(\neg\varphi_1 \vee \neg\varphi_2)$,

- implication: $\varphi_1 \rightarrow \varphi_2 := \neg\varphi_1 \vee \varphi_2$,
- eventual: $\Diamond\varphi := True\ \mathcal{U}\ \varphi$,
- always: $\Box\varphi := \neg\Diamond\neg\varphi$,
- release: $\varphi_1\ \mathcal{R}\ \varphi_2 := \neg(\neg\varphi_1\ \mathcal{U}\ \neg\varphi_2)$.

These extra logical operators do not extend the expressive power of LTL but with them an LTL formula φ can be written into a formula φ' in positive normal form, that is, all the negations in φ' only appear in front of the atomic propositions [7]. As will be presented in the later sessions, such formula manipulation comes in handy when the robust satisfaction of an LTL formula is considered.

2) Semantics: Let $\mathbb{x} = x(1)x(2)x(3)\dots$ be a infinite sequence of points in $\mathbb{R}^n$ and let AP be a set of atomic propositions. We define a labeling map $L : \mathbb{R}^n \rightarrow 2^{AP}$ and interpret an LTL formula over the labeling sequence $L\big(x(1)\big)L\big(x(2)\big)L\big(x(3)\big)\dots$ as follows:

- $\mathbb{x} \models \pi$ iff $\pi \in L\big(x(1)\big)$,
- $\mathbb{x} \models \neg\varphi$ iff $\mathbb{x} \not\models \varphi$,
- $\mathbb{x} \models \varphi_1 \vee \varphi_2$ iff $\mathbb{x} \models \varphi_1$ or $\mathbb{x} \models \varphi_2$,
- $\mathbb{x} \models \bigcirc\varphi$ iff $\mathbb{x}_2 \models \varphi$,
- $\mathbb{x} \models \varphi_1\ \mathcal{U}\ \varphi_2$ iff $\exists s \geq 1 : \mathbb{x}_s \models \varphi_2$ and $\forall t < s : \mathbb{x}_t \models \varphi_1$.

Given an infinite word $\mathbb{x}$ and an LTL formula φ, we say φ holds for $\mathbb{x}$ (or $\mathbb{x}$ satisfies φ) iff $\mathbb{x} \models \varphi$.

For a continuous-time signal $\xi : [0, +\infty) \rightarrow \mathbb{R}^n$ one can consider the LTL fragment without next operator $\bigcirc$, denoted by $\text{LTL}_{\setminus\bigcirc}$. The ω-word $\mathbb{x} = x(1)x(2)x(3)\dots$ that is *consistent* with continuous signal ξ can be generated by breaking signal ξ at a collection of time instants $\{t_k\}_{k=1}^{\infty}$, such that $t_{k+1} > t_k$ and $\lim_{k\rightarrow\infty} = \infty$, and $L\big(\xi(t)\big) = x(k)$ for all $t \in [t_k, t_{k+1}]$. Such an consistent ω-word $\mathbb{x}$ is stutter equivalent

with the continuous-time signal ξ [7]. In the rest of this work, we will consider $\text{LTL}_{\setminus\bigcirc}$ formulas when dealing with a continuous-time system.

2.2.2 Absolutely Decomposable Property

A linear time property called absolutely decomposable property is defined. Intuitively, it specifies a property whose satisfaction can be "reset" at anytime, as long as it has not been violated yet. This is the key property that assures the later proposed control synthesis algorithms to be robust to detection delay. In what follows we first introduce some preliminaries on linear time properties and its decomposition.

A linear time (LT) property P over atomic propositions AP is a subset of $(2^{AP})^\omega$. An LT property P is called a safety property if a word $\mathbb{w}$ belonging to P is equivalent to the following: for all $\mathbf{p} \in pref(\mathbb{w})$ there exists $\mathbb{s} \in (2^{AP})^\omega$ such that $\mathbf{p}\mathbb{s} \in P$, where $\mathbf{p}\mathbb{s}$ is an ω-word obtained by appending $\mathbb{s}$ behind $\mathbf{p}$. An LT property P is called a liveness property if for all $\mathbf{p} \in (2^{AP})^*$ there exists $\mathbb{s} \in (2^{AP})^\omega$ such that $\mathbf{p}\mathbb{s} \in P$. It is well known that any LT property can be written as the conjunction of a safety property and a liveness property [7]. In particular, such decomposition is not unique but a canonical sharp one exists [7]. That is, there exists a decomposition $P = P^\star_{\text{safety}} \cap P^\star_{\text{liveness}}$, such that for any other decomposition $P = P_{\text{safety}} \cap P_{\text{liveness}}$, one has $P^\star_{\text{safety}} \subseteq P_{\text{safety}}$ and $P_{\text{liveness}} \subseteq P^\star_{\text{liveness}}$. Here $P^\star_{\text{safety}}$ is called the safety closure of P and $P^\star_{\text{liveness}}$ is called the liveness closure of P.

Whenever an LT property P is defined by an LTL formula φ, i.e., $P = Word(\varphi) := \{\mathbb{w} \in (2^{AP})^\omega \mid \mathbb{w} \models \varphi\}$, one can find LTL formulas φ_{safe}, $\varphi_{\text{liveness}}$ such that $\varphi = \varphi_{\text{safe}} \wedge \varphi_{\text{liveness}}$ and $Word(\varphi_{\text{safety}})$, $Word(\varphi_{\text{liveness}})$ are safety and liveness properties, respectively. This can be done systematically by i) constructing two nondeterministic Buchi automaton (NBA), one generating the safety closure of $Word(\varphi)$ and the other

generating the liveness closure of $Word(\varphi)$ [3], and ii) converting the two NBA's into φ_{safe} and $\varphi_{\text{liveness}}$ respectively[4].

Next, we introduce a special type of LTL formula that specifies an absolutely decomposable property [128], which is defined as follows.

Definition 5. Let $P \subseteq (2^{AP})^{\omega}$ be a property over set AP. Property P is called *absolutely decomposable* if there exists a decomposition $P = P_{\text{safety}} \cap P_{\text{liveness}}$, such that

- P_{safety} is an *absolute safety* property, i.e., it is a safety property, and $\mathbf{p} \in pref(P_{\text{safety}})$, $\mathbb{w} \in P_{\text{safety}}$ implies that $\mathbf{p}\mathbb{w} \in P_{\text{safety}}$;
- P_{liveness} is an *absolute liveness* property, i.e., it is a liveness property, and $\mathbf{p} \in pref(P_{\text{liveness}})$, $\mathbb{w} \in P_{\text{liveness}}$ implies that $\mathbf{p}\mathbb{w} \in P_{\text{liveness}}$.

Note that $pref(P_{\text{liveness}}) = (2^{AP})^*$, thus the definition of absolute liveness coincides with the one given by [2]. In what follows, some useful results regarding absolutely decomposable properties are presented. To this point, we first give a lemma about general safety properties that is used in the later proofs.

Lemma 3. Let P_1 and P_2 be two safety property over AP, $pref(P_1) = pref(P_2)$ implies that $P_1 = P_2$.

Proof. Assume for a contradiction that $pref(P_1) = pref(P_2)$ but $P_1 \neq P_2$. Without loss of generality, this means there exists $\mathbb{w} = w(1)w(2)\cdots \in P_1$ such that $\mathbb{w} \notin P_2$. Since $\mathbb{w} \notin P_2$ and P_2 is a safety property, we immediately know that $\mathbb{w}$ has a bad prefix $\mathbf{p} := w(1)w(2)\cdots w(t) \notin pref(P_2)$. But on the other hand, $\mathbb{w} \in P_1$ and this implies that $\mathbf{p} \in pref(P_1) = pref(P_2)$, which is a contradiction. □

Propoition 3. Let P be an absolutely decomposable property, for all $\mathbf{p} \in pref(P)$,

[4]In general, it is possible that an NBA cannot be described by an LTL formula, but there exist algorithms doing the conversion whenever it is possible [31].

$\mathbb{w} \in P$, $\mathbf{p}\mathbb{w} \in P$.

Proof. Let $\mathbf{p} \in pref(P)$ and $\mathbb{w} \in P$. First, notice the fact that $P = P_{\text{safety}} \cap P_{\text{liveness}}$. This implies that (i) $\mathbf{p} \in pref(P) \subseteq pref(P_{\text{safety}})$, (ii) $\mathbf{p} \subseteq pref(P_{\text{liveness}})$, (iii) $\mathbb{w} \in P_{\text{safety}}$ and $\mathbb{w} \in P_{\text{liveness}}$. By bullet 1 in Definition 5, we have $\mathbf{p}\mathbb{w} \in P_{\text{safety}}$, and by bullet 2, $\mathbf{p}\mathbb{w} \in P_{\text{liveness}}$. Thus $\mathbf{p}\mathbb{w} \in P = P_{\text{safety}} \cap P_{\text{liveness}}$. □

With the following proposition, we show that GR(1) formulas, an LTL fragment of vast interest in reactive synthesis, is absolutely live.

Propoition 4. GR(1) formulas specify absolute liveness properties. That is, suppose

$$\varphi = \bigwedge_{i \in I} \square\lozenge\varphi_i \rightarrow \bigwedge_{j \in J} \square\lozenge\varphi_j, \tag{2.22}$$

where I and J be finite index sets, and φ_i, φ_j are propositional formulas, then $Word(\varphi)$ is an absolute liveness property.

Proof. By definition, it should be clear that

1) absolute liveness properties are closed under union and intersection, and hence LTL formulas specifies such properties must be closed under disjunction and conjunction;
2) $\square\lozenge\psi$ and $\lozenge\square\psi$ specifies absolute liveness properties for any propositional formula ψ.

Note that φ can be rewritten as

$$\begin{aligned}
\varphi &= \bigwedge_{i\in I} \Box\Diamond\varphi_i \rightarrow \bigwedge_{j\in J} \Box\Diamond\varphi_j \\
&= \left(\neg \bigwedge_{i\in I} \Box\Diamond\varphi_i\right) \vee \left(\bigwedge_{j\in J} \Box\Diamond\varphi_j\right) \\
&= \left(\bigvee_{i\in I} \neg\big(\Box\Diamond\varphi_i\big)\right) \vee \left(\bigwedge_{j\in J} \Box\Diamond\varphi_j\right) \\
&= \left(\bigvee_{i\in I} \Diamond\Box\neg\varphi_i\right) \vee \left(\bigwedge_{j\in J} \Box\Diamond\varphi_j\right). \qquad (2.23)
\end{aligned}$$

Applying bullet 1) and 2) to Eq. (2.23) proves that φ is absolute live. □

Propoition 5. Let P_1, P_2 be two absolute safety properties, $P = P_1 \cap P_2$ is also absolute safety property.

Proof. Proposition 5 easily follow from the definition of absolute safety properties.

□

Propoition 6. Let P be an absolutely decomposable property with the specific decomposition $P = P_{\text{safety}} \cap P_{\text{liveness}}$, then $pref(P) = pref(P_{\text{safety}})$.

Proof. From the proof of Proposition 3, we already know that $pref(P) \subseteq pref(P_{\text{safety}})$. Next we show the other direction. For this purpose, let $\mathbf{p} \in pref(P_{\text{safety}})$ and $\mathbb{w} \in P$ be arbitrary. Next we show $\mathbf{p}\mathbb{w} \in P$ and conclude $\mathbf{p} \in pref(P)$.

(a) First, note that $\mathbf{p} \in pref(P_{\text{safe}})$ and that $\mathbb{w} \in P \subseteq P_{\text{safe}}$. By bullet 1 in Definition 5, we have $\mathbf{p}\mathbb{w} \in P_{\text{safety}}$.

(b) Secondly, also note that $\mathbf{p} \in pref(P_{\text{liveness}}) = (2^{AP})^*$, and $\mathbb{w} \in P \subseteq P_{\text{liveness}}$. By bullet 2 in Definition 5, we have $\mathbf{p}\mathbb{w} \in P_{\text{liveness}}$.

Combining (a) and (b), we have $\mathbf{p}\mathbb{w} \in P_{\text{safety}} \cap P_{\text{liveness}} = P$. Therefore $\mathbf{p} \in pref(P)$ and this finishes the proof. □

Propoition 7. Let P be an absolutely decomposable property with a specific decomposition $P = P_{\text{safety}} \cap P_{\text{liveness}}$, and let $P = P^{\star}_{\text{safety}} \cap P^{\star}_{\text{liveness}}$ be the sharpest decomposition, then $P_{\text{safety}} = P^{\star}_{\text{safety}}$.

Proof. By $P = P^{\star}_{\text{safety}} \cap P^{\star}_{\text{liveness}}$, we have $P \subseteq P^{\star}_{\text{safety}}$. This hence gives

$$pref(P) \subseteq pref(P^{\star}_{\text{safety}}). \tag{2.24}$$

On the other hand, since $P^{\star}_{\text{safety}}$ comes from the sharpest decomposition, $P^{\star}_{\text{safety}} \subseteq P_{\text{safety}}$. This implies that

$$pref(P^{*}_{\text{safety}}) \subseteq pref(P_{\text{safety}}). \tag{2.25}$$

Combine (2.24), (2.25), we have

$$pref(P) \subseteq pref(P^{\star}_{\text{safety}}) \subseteq pref(P_{\text{safety}}). \tag{2.26}$$

But by Proposition 6, we know that $pref(P) = pref(P_{\text{safety}})$, which forces all "$\subseteq$" in Eq. (2.26) to be "$=$". Thus we have $pref(P^{\star}_{\text{safety}}) = pref(P_{\text{safety}})$. Applying Lemma 3, we have $P^{\star}_{\text{safety}} = P_{\text{safety}}$. □

Propoition 8. Let P_1 be an absolutely decomposable property under decomposition $P_1 = P_{1,\text{safety}} \cap P_{1,\text{liveness}}$, and let $P_{2,\text{safety}}$ be an absolute safety property, then $P = P_1 \cap P_{2,\text{safety}}$ is absolutely decomposable under $P = P_{\text{safety}} \cap P_{\text{liveness}}$, where $P_{\text{safety}} = P_{1,\text{safety}} \cap P_{2,\text{safety}}$ and $P_{\text{liveness}} = P_{1,\text{liveness}}$.

Proof. First note that P_{safety} is indeed a safety property and $P_{\text{safety}} \cap P_{\text{liveness}} = (P_{1,\text{safety}} \cap P_{2,\text{safety}}) \cap P_{\text{liveness}} = P_{2,\text{safety}} \cap (P_{1,\text{safety}} \cap P_{1,\text{liveness}}) = P$ is a valid decomposition. Moreover, by Proposition 5, P_{safety} is a absolute safety property. By definition P is absolutely decomposable, and $P_{\text{safety}} = P_{1,\text{safety}} \cap P_{2,\text{safety}}$ is the unique absolute safety property involved in the decomposition by Proposition 7. □

2.2.3 Suffix-closed Property

A linear time property called suffix-closed property is introduced.

Definition 6. *(Suffix-closed Property)* Let $P \subseteq (2^{AP})^\omega$ be an LT property over set AP. Property P is called *suffix-closed* if $suff(P) \subseteq P$. We call an LTL formula φ suffix-closed if $Word(\varphi)$ is suffix-closed.

It can be shown that GR(1) formulas is suffix-closed.

Propoition 9. GR(1) formulas are suffixed closed.

Proof. By definition, it should be clear that

1) suffix-closed properties are closed under union and intersection, and hence suffix-closed LTL formulas are closed under disjunction and conjunction;
2) $\square\lozenge\psi$ and $\lozenge\square\psi$ are suffix-closed for any propositional formula ψ.

Similar to the proof of Proposition 4, applying bullet 1) and 2) to Eq. (2.23) proves that φ is suffix-closed. □

We next discuss the connection between suffix-closedness properties and absolutely decomposable properties. An important LT property that bridges the two aforementioned properties is invariance.

Definition 7. *(Invariance)* An LT property $P \subseteq (2^{AP})^\omega$ is an *invariance property* if there exists a propositional formula ψ such that for any $\mathbb{w} = w(1)w(2)\cdots \in P$ and any j, we have $w(j) \models \psi$.

Propoition 10. An LT property $P \subseteq (2^{AP})^\omega$ is both suffix-closed and absolutely safe iff P is an invariance property.

Proof. Clearly, invariance property is both absolutely safe and suffix-closed. To show the "only if" part, define $W_i(P) := \{w \in 2^{AP} \mid w = w(i) \text{ for some } \mathbb{w} =$

$w(1)w(2)w(3)\cdots \in P\}$. We can find a propositional formula ψ in disjunction normal form such that $w \models \psi$ iff $w \in W_1(P)$. Since P is suffix-closed, we have

$$W_{i+1}(P) \subseteq W_i(P), \tag{2.27}$$

whereas by P being absolute safety, we have

$$W_{i+1}(P) \supseteq W_i(P). \tag{2.28}$$

Hence $W_{i+1}(P) = W_i(P)$. This implies that $W_i(P) = W_1(P)$ for all i. Hence for any i and $w \in W_i(P)$, $w \models \psi$ holds, i.e., P is an invariance property. □

Moreover, it can be shown that if an LT property is both absolutely decomposable and suffix-closed, its safety closure must be an invariance property.

Propoition 11. If $P \subseteq (2^{AP})^\omega$ is both absolutely decomposable and suffix-closed, then its safety closure $P^\star_{\text{safety}}$ is an invariance property.

Proof. Since P is absolutely decomposable, we know that $P^\star_{\text{safety}}$ is absolute safety by Proposition 7. It is then enough to show that $P^\star_{\text{safety}}$ is also suffix-closed, and the desired result will follow from Proposition 10.

To show $P^\star_{\text{safety}}$ is indeed suffix-closed, let $\mathbb{w} = w(1)w(2)\cdots \in P^\star_{\text{safety}}$ be arbitrary, and let $\mathbb{s} = w(r)w(r+1)\ldots$ be arbitrary suffix of $\mathbb{w}$ starting from r^{th} position. Since $P^\star_{\text{safety}}$ is the safety closure of P, we have

$$\forall t : \exists \overline{\mathbb{w}} \in P : \forall \tau \leq t : \overline{w}(\tau) = w(\tau). \tag{2.29}$$

Let $\overline{\mathbb{s}} = \overline{w}(r)\overline{w}(r+1)\ldots$, we know $\mathbb{s} \in P$ as P is assumed to be suffix-closed. Together with Eq. (2.29), this implies

$$\forall t : \exists \overline{\mathbb{s}} \in P : \forall \tau \leq t : \overline{s}(\tau) = s(\tau). \tag{2.30}$$

That is $\mathbb{s} \in P^\star_{\text{safety}}$. Recall that $\mathbb{s}$ is arbitrary suffix of arbitrary $\mathbb{w} \in P^\star_{\text{safety}}$, $P^\star_{\text{safety}}$ is hence suffix-closed by definition and the proof is completed. □

CHAPTER III

Fault Detectability Analysis with LTL Constraints

Fault detection is one of the key component of fault tolerant control as contingent actions cannot be taken without knowing if the fault has already occurred or not. Since the correct-by-construction control synthesis techniques are all model-based approaches in essence, they naturally incorporate better with model-based fault detection, which are more promising in terms of providing a formal guarantee. At high level, such model-based fault detection algorithms, regardless of being developed for discrete or continuous state systems, are all trying to solve the same problem at runtime, i.e., checking if the observed data is consistent with the healthy system model or not. Due to the nondeterminisitc nature of these system models, a fault may not be detected immediately after their occurrence because the uncertainties (e.g., disturbance, noise) in the model may hide the fault. For finite discrete systems, a determinization of the system can be done via power set construction. However, this method is intractable for continuous state systems because the memory it requires grows with time. This is one of the key challenge in developing model-based detection for complex systems where both discrete and continuous states are present.

One way to address the aforementioned issue is to consider a detector with finite memory of length T, but also provide a proof that, even in the worst-case, the faulty

system can be distinguished from the healthy system in finite time T as well. This constant T can be also interpreted as the worst case detection delay, and a smaller T is clearly favorable. Whenever the model of the healthy and the faulty system are both available, it is also possible to determine the minimum T. This is called detectability analysis, and it essentially amounts to checking whether the reachable sets of the healthy and faulty models become disjoint within time T. However, such worst case analysis may sometimes conclude conservatively that a fault is not detectable in finite time, while this may not be true in practice because some extra side information (e.g., the LTL specification that the system should satisfy) are not incorporated in the detectability analysis.

In this chapter, we consider switched affine systems, whose detectability analysis can be accomplished by solving a MILP. We further assume that when the system operation is normal (i.e., the system is healthy), the switching mode signal satisfies a certain LTL formula; and in case of faults, it satisfies a different (possibly trivial) formula, capturing the potential switching patterns during normal operation and anomalies, respectively. The main contribution is to show that by combining dynamical models (given as switched affine systems) and behavioral models (given as LTL formula), one can reduce the worst-case fault detection time a receding horizon algorithm guarantees. We express the LTL constraints (restricted in a finite horizon) with a nondeterministic finite state machine called a monitor, which is then transformed into a set of mixed integer linear constraints that can be easily integrated in the MILP used for the detectability analysis.

Chapter overview. In Section 3.1, we formally introduce the concept of switched affine systems, model invalidation based fault detection, MILP based fault detectabil-

ity analysis and LTL monitoring. The connection between model invalidation and monitoring is also discussed. In Section 3.2, the detectability analysis problem with LTL constraints is defined. In Section 3.3, we present the proposed approach. It is shown that detectability analysis in the setting can be conducted by creating a monitor finite state machine for the LTL formula and encoding the restriction the formula imposes on the system behavior as mixed integer linear constraints, which is then incorporated into the MILP for the offline worst-case detectability analysis. Finally, these ideas are illustrated with an example in Section 3.4, where a collection of unmanned aerial vehicles are implementing a consensus protocol over a communication network with time-varying connectivity. We show how detectability can be guaranteed when network connectivity patterns change using the proposed approach.

Related work. The work presented in this chapter is related to several fault detection approaches developed by different communities. For discrete (e.g., software-based) systems, monitoring and run-time verification techniques have been proposed [6, 49, 10, 97, 102]. Similarly, fault detection algorithms have also been studied for continuous-state dynamical systems using ideas from filtering or optimization [53, 38]. This work tries to bring together the ideas from the two communities to obtain a less conservative fault detection method for a class of hybrid systems, governed by both discrete and continuous variables.

In particular, we approach the continuous aspects of this problem from the perspective of model invalidation [47, 92, 108]. Model invalidation is a robust system identification process that checks whether some given input output data can be explained by a given model, and is tightly related to fault detection as pointed out in [90, 91]. Comparing to other widely used model-based fault-detection methods

such as residual generation [38], the model invalidation based approach is more suitable for providing a finite detection guarantee rather than an asymptotical one. The idea of model invalidation is also very similar to the model conformance problem studied in computer science community [17, 113], in particular to the input/output conformance for discrete-time systems [113, 114].

The discrete aspect of the fault detection problem is handled with LTL monitoring [10]. At high level, model invalidation and the LTL monitoring can be both viewed from the standpoint of set membership fault detection [116, 59, 52], because they essentially amount to computing the reachable set of some uncertain model under a given input sequence and checking if the observed output sequence (trajectories) lies in this reachable set or not. This connection provides the key intuition to bring together these two techniques.

The work on the fault detectability presented here tightly follows [47] and its conference versions [46, 45, 48], and is also related to their follow-up work for structured systems [40]. In [47], the detectability analysis with LTL constraints is discussed, but those LTL constraints are only imposed on the switching sequence for the faulty system, which is easy to incorporate in the MILP encoding [58, 117]. The key novelty of this work is to handle the case where another LTL formula φ is used to describe the healthy mode switching sequence. Since this LTL formula φ a is not necessarily evaluated at the beginning of the receding horizon of the detector, its MILP encoding is not trivial without using a monitor finite transition system.

3.1 Preliminaries

3.1.1 Fault Detection for Switched Affine Systems via Model Invalidation

Switched Affine Systems

A discrete-time switched affine system $\mathcal{S}$ is described by the following difference equations:

$$x_{t+1} = A_{s_t} x_t + B_{s_t} u_t + E_{s_t} w_t + K_{s_t},$$

$$y_t = C_{s_t} x_t + D_{s_t} u_t + F_{s_t} v_t, \tag{3.1}$$

where

- $x \in X \subseteq \mathbb{R}^n$ is the **unobserved** internal state,
- $u \in U \subseteq \mathbb{R}^m$ is the **observed** input,
- $w \in W \subseteq \mathbb{R}^l$ is the **unobserved** input,
- $y \in Y \subseteq \mathbb{R}^p$ is the **observed** output,
- $v \in V \subseteq \mathbb{R}^q$ is the **unobserved** measurement noise (can be viewed as input),
- s is the **observed** switching mode from a finite set $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$.

We also assume that sets U, V, W, X, Y are polytopes in their spaces, and that A_s, B_s, C_s, D_s, E_s, F_s, K_s are matrices with proper sizes.

Such models can be used to describe, for instance, physical systems with discrete actuation or closed-loop systems with continuous plants and logic-based controllers. Switching mode captures the discrete, logic-based variables.

Guaranteed Fault Detection via Model Invalidation

In this chapter, we consider fault detection of switched affine systems. By a fault, we mean a sudden and permanent change of the system dynamics in Eq. (3.1), due to physical component failures or extreme operating conditions. Such changes can be reflected by dynamics being governed by different system matrices, and by having larger admissible uncertainty set V and W. Since the uncertainty w and v may "hide" a fault in the worst case, the behavior of the faulty system may not be distinguishable from the healthy one immediately after the fault occurs. Our goal is to detect the fault occurrence as soon as possible. In particular, the correctness

of the detection needs to be guaranteed, meaning that the fault must have already happened, once detected.

It is shown in [47] that such guaranteed fault detection can be done using a model invalidation approach. The model invalidation problem addresses the following question: at time instant t_0, given a sequence $\{u_t, s_t, y_t\}_{t=t_0-N+1}^{t_0}$ of past inputs and outputs over a finite window of length N, can we find an admissible unobserved sequence $\{x_t, w_t, v_t\}_{t=t_0-N+1}^{t_0}$ such that the output $\{y_t\}_{t=t_0-N+1}^{t_0}$ is indeed generated by the system in Eq. (3.1) under input $\{u_t, s_t, w_t, v_t\}_{t=t_0-N+1}^{t_0}$? If no such unobserved sequences can be found, the actual observation cannot possibly be generated by the healthy system model (i.e., the model is invalidated). We can hence claim that a fault that changes the system dynamics must have occurred within the examined time window.

For switched affine systems, the model invalidation problem can be formulated as a linear program (LP)

$$
\begin{aligned}
\text{find} \quad & \{x_t, w_t, v_t\}_{t=t_0-N+1}^{t_0} \\
\text{s.t.} \quad & x_{t+1} = \textstyle\sum_{k=1}^{|\Sigma|} a_k^t \Big(A_{\sigma_k} x_t + B_{\sigma_k} u_t + E_{\sigma_k} w_t + K_{\sigma_k} \Big), \\
& \forall t \in [\![t_0 - N + 1, t_0 - 1]\!], \\
& y_t = \textstyle\sum_{k=1}^{|\Sigma|} a_k^t \Big(C_{\sigma_k} x_t + D_{\sigma_k} u_t + F_{\sigma_k} v_t \Big), \\
& \forall t \in [\![t_0 - N + 1, t_0]\!], \\
& x_t \in X, w_t \in W, v_t \in V, \forall t \in [\![t_0 - N + 1, t_0]\!],
\end{aligned}
\tag{3.2}
$$

where $[\![a, b]\!] := \{c \in \mathbb{N} \mid a \leq c \leq b\}$ for two integers $a < b$, and a_k^t is a binary indicator that takes value 1 if and only if (iff) $s_t = \sigma_k$. Note that $\{u_t, y_t\}_{t=t_0-N+1}^{t_0}$ and a_k^t (which is known from $\{s_t\}_{t=t_0-N+1}^{t_0}$) are all parameters rather than variables in the above feasibility problem, as $\{u_t, s_t, y_t\}_{t=t_0-N+1}^{t_0}$ are observed at each time. This means the feasibility problem in Eq. (3.2) does not contain integer variables

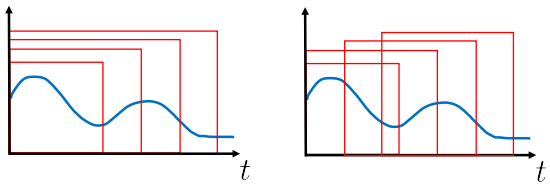

Figure 3.1: Growing horizon scheme (left) versus receding horizon scheme (right). The red boxes mark the growing/shifted time window.

and is hence an LP.

To perform model invalidation based fault detection at run-time, one needs to update the time window (i.e., horizon) to incorporate newly collected data. As pointed out in [47], there are two ways of changing the horizon at run-time: one is called the growing horizon scheme (Fig. 3.1, left) and the other is known as a receding horizon scheme (Fig. 3.1, right). With the growing horizon scheme, we start at time $t = 0$ with a horizon of length $N = 0$, and increase N by one at each time step. In this case, $N \to \infty$ as time grows. Under the receding horizon scheme, we stop growing the horizon length whenever it reaches a certain value, and we start to shift the time window after that. That is, at every time instant t_0, we collect the most recent history of the observed variables from time window $[\![t_0 - N + 1, t_0]\!]$ and perform the above model invalidation procedure. If the system is invalidated, we claim a fault; otherwise we shift the time window to $[\![t_0 - N + 2, t_0 + 1]\!]$ and repeat the procedure with the updated data in the shifted window. Theoretically, the growing horizon scheme may lead to an earlier detection than the receding horizon scheme because the latter one drops older observations. We say the receding horizon detector is more conservative compared to the growing horizon detector in the sense the former may miss a fault that is detectable by the latter. However, the growing horizon scheme is not practical because it requires the memory to grow to infinity. We hence always implement the receding horizon scheme in practice to keep the memory finite.

Detectability Analysis

Note that the fault detection technique via model invalidation may not be complete, in the sense that a fault may remain undetected indefinitely. There are two sources of this: (i) the fault dynamics can be inherently indistinguishable from the nominal dynamics, (ii) the invalidation process is conservative, e.g., due to using a fixed horizon. For the latter issue, a longer window tends to make the detector "closer" to being complete. On the other hand, if the models of both the healthy system $\mathcal{S}$ and the faulty system $\mathcal{S}^{\mathrm{f}}$ are known, it is possible to verify if the detection is complete with a given window length N. That is, if a fault occurs whether it will be detected within N time steps by the receding horizon detector. We call a healthy-faulty system pair $(\mathcal{S}, \mathcal{S}^{\mathrm{f}})$ to be "N-detectable" if this is the case.

For a given healthy-faulty system pair $(\mathcal{S}, \mathcal{S}^{\mathrm{f}})$ and a positive integer N, the detectability analysis answer the following question: is system pair $(\mathcal{S}, \mathcal{S}^{\mathrm{f}})$ N-detectable? If yes, what is the minimal N such that the pair is N-detectable? From a theoretical point of view, it is important if we can prove N-detectability of a system pair because it allows us to use a receding horizon detector without missing any faults due to its conservativeness compared to the growing horizon detector. From a practical point of view, it is also important to find the minimal N so that the receding horizon detector does not need to keep an unnecessarily long memory.

To analyze the detectability of system pair $(\mathcal{S}, \mathcal{S}^{\mathrm{f}})$, we construct the so called N-behavior set $\mathfrak{B}_N(\mathcal{S})$ (and $\mathfrak{B}_N(\mathcal{S}^{\mathrm{f}})$, respectively), i.e., the set of all observed input-output sequences of length N that can be possibly generated by the healthy system $\mathcal{S}$ (or the faulty system $\mathcal{S}^{\mathrm{f}}$, respectively), and check if the two sets intersect. If $\mathfrak{B}_N(\mathcal{S}) \cap \mathfrak{B}_N(\mathcal{S}^{\mathrm{f}}) = \emptyset$, then the healthy and the faulty behavior must differ within N time steps. In this case, the minimal horizon length T that is necessary for

the detection to be complete (i.e., $T := \min\left\{N \mid \mathfrak{B}_N(\mathcal{S}) \cap \mathfrak{B}_N(\mathcal{S}^{\mathrm{f}}) = \emptyset\right\}$) can be computed by a line search over ascending N, starting from $N = 1$.

If the dynamics of system $\mathcal{S}$ satisfies Eq. (3.1), the N-behavior set of system $\mathcal{S}$ can be described by mixed integer linear constraints. In this case, $\mathfrak{B}_N(\mathcal{S}) \cap \mathfrak{B}_N(\mathcal{S}^{\mathrm{f}}) = \emptyset$ is equivalent to a MILP being infeasible. Formally, $\mathfrak{B}_N(\mathcal{S})$ is defined by Eq. (3.3).

(3.3)
$$\mathfrak{B}_N(\mathcal{S}) = \left\{ \begin{array}{l|c} \{u_t, s_t, y_t\}_{t=1}^N \in & \exists\{x_t, w_t, v_t\}_{t=1}^N \in (X \times W \times V)^N : \\ (U \times \Sigma \times Y)^N & \{u_t, v_t, w_t, x_t, y_t, s_t\}_{t=1}^N \text{ satisfy } \mathcal{S}\text{'s dynamics} \end{array} \right\},$$

where the constraints in Eq. (3.3) can be expressed with exactly the same set of the formulas in Eq. (3.2) (after shifting the time window to $[\![1, N]\!]$), except that now the observed sequences $\{u_t, y_t\}_{t=1}^N$ and the auxiliary binary variables a_k^t are also variables rather than parameters of the constraints, and that u_t, a_k^t, y_t must satisfy

$$u_t \in U, \quad y_t \in Y, \quad \forall t \in [\![1, N]\!], \tag{3.4}$$

$$a_k^t \in \{0, 1\}, \quad \forall t \in [\![1, N]\!], k \in [\![1, |\Sigma|]\!], \tag{3.5}$$

and a_k^t must also satisfy

$$\sum_{\sigma_k \in \Sigma} a_k^t = 1, \quad \forall t \in [\![1, N]\!]. \tag{3.6}$$

Note that with a_k^t being variables, the constraints describing $\mathfrak{B}_N(\mathcal{S})$ now contain bilinear terms $a_k^t u_t, a_k^t x_t$, $a_k^t w_t$, $a_k^t v_t$ (see Eq. (3.2)). These bilinear constraints can be transformed into linear ones by introducing some continuous-valued auxiliary variables, which leads to a set of (mixed integer) linear constraints. The detailed transformation procedure can be found in [46]. To simplify the notations, we will denote the obtained overall mixed integer linear constraints by

$$\mathbf{H}_N^{\mathcal{S}}\left(\left\{u_t, \{a_k^t\}_{k=1}^{|\Sigma|}, y_t, x_t, w_t, v_t, \right\}_{t=1}^N, \boldsymbol{\xi}_{\mathrm{ob}}, \boldsymbol{\xi}_{\mathrm{un}} \right) \leq 0,$$

$$a_k^t \in \{0, 1\}, \quad \forall t \in [\![1, N]\!], k \in [\![1, |\Sigma|]\!], \tag{3.7}$$

where $\boldsymbol{\xi}_{\text{ob}}$ is the continuous-valued auxiliary variable that comes from $a_k^t u_t$, and $\boldsymbol{\xi}_{\text{un}}$ is the auxiliary variable that comes from $a_k^t x_t$, $a_k^t w_t$, $a_k^t v_t$, and $\mathbf{H}_N^{\mathcal{S}}$ is an affine function that depends on the system matrices of $\mathcal{S}$ and horizon length N. With this notation, $\mathfrak{B}_N(\mathcal{S}) \cap \mathfrak{B}_N(\mathcal{S}^{\text{f}}) = \emptyset$ is equivalent to the following MILP being infeasible

$$
\begin{aligned}
\text{find} \quad & \left\{ u_t, \{a_k^t\}_{k=1}^{|\Sigma|}, y_t, x_t, x_t^{\text{f}}, w_t, w_t^{\text{f}}, v_t, v_t^{\text{f}} \right\}_{t=1}^{N}, \boldsymbol{\xi}_{\text{ob}}, \boldsymbol{\xi}_{\text{un}}, \boldsymbol{\xi}_{\text{un}}^{\text{f}}, \\
\text{s.t.} \quad & \mathbf{H}_N^{\mathcal{S}} \left(\left\{ u_t, \{a_k^t\}_{k=1}^{|\Sigma|}, y_t, x_t, w_t, v_t, \right\}_{t=1}^{N}, \boldsymbol{\xi}_{\text{ob}}, \boldsymbol{\xi}_{\text{un}} \right) \leq 0, \\
& \mathbf{H}_N^{\mathcal{S}^{\text{f}}} \left(\left\{ u_t, \{a_k^t\}_{k=1}^{|\Sigma|}, y_t, x_t^{\text{f}}, w_t^{\text{f}}, v_t^{\text{f}}, \right\}_{t=1}^{N}, \boldsymbol{\xi}_{\text{ob}}, \boldsymbol{\xi}_{\text{un}}^{\text{f}} \right) \leq 0, \\
& a_k^t \in \{0,1\}, \quad \forall t \in [\![1, N]\!], k \in [\![1, |\Sigma|]\!] \leq 0.
\end{aligned} \tag{3.8}
$$

3.1.2 LTL Monitoring

In this work we consider fault detectability analysis of switched affine systems whose mode sequences $\mathbb{s} = s_1 s_2 s_3 \ldots$ must satisfy certain LTL formulas. In particular, our goal is to show how such side information can improve detectability analysis. In what follows, we briefly recall LTL and some related concepts from automata theory that will be useful to encode the LTL constraints on $\mathbb{s}$.

Monitor

We introduce several concepts related to LTL monitoring that will be used to encode the LTL constraints in the fault detectability analysis.

Definition 8. Let φ be an LTL formula. A finite word $\mathbf{p} \in \Sigma^*$ is called a *bad prefix* of φ if[1] for all $\mathbb{s} \in \Sigma^\omega$, $\mathbf{p}\mathbb{s} \not\models \varphi$, where $\mathbf{p}\mathbb{s}$ is the ω-word obtained by concatenating $\mathbb{s}$ to $\mathbf{p}$. Otherwise we call $\mathbf{p}$ a *(valid) prefix* of φ.

Definition 9. Given and LTL formula φ, a *monitor* $\mathcal{M}^\varphi$ is a tuple $(\Sigma, Q, Q_{\text{init}}, \delta)$, where Σ is a finite set of letters, Q is a finite set of states, $Q_{\text{init}} \subseteq Q$ is a set of

[1]Note that φ has no bad prefixes if it specifies a liveness property, hence a finite word being a bad prefix of φ is equivalent to the word being a bad prefix of the safety closure of the language accepted by φ.

initial states, and partial function $\delta : Q \times \Sigma \to 2^Q$ is the nondeterministic[2] transition map. Moreover, $\mathcal{M}^\varphi$ satisfies the following condition: a finite word $\mathbf{p} = p_1 p_2 \dots p_N$ is a valid prefix of φ if and only if there exists $\mathbf{q} = q_1 q_2 \dots q_{N+1} \in Q^{N+1}$ such that $q_1 \in Q_{\text{init}}$ and $q_{i+1} \in \delta\big(q_i, p_i\big)$ for $i \in [\![1, N]\!]$.

The monitor finite state machine $\mathcal{M}^\varphi$ can be viewed as a model that generates sequences exactly from $\{\mathbf{p} \in \Sigma^* \mid \mathbf{p} \text{ is a valid prefix of } \varphi\}$. It is well known that a monitor $\mathcal{M}^\varphi$ can be constructed for every LTL formula φ [30].

3.1.3 Fault Detection versus Run-time Verification

We would like to point out the connection between the model invalidation based fault detection and run-time verification.

In run-time verification, we are given an LTL formula and desire to verify if a sequence $\mathbf{s} = s_1 s_2 \dots s_M \in \Sigma^*$ is a bad prefix of φ. To this end, we construct monitor $\mathcal{M}^\varphi = (\Sigma, Q, Q_{\text{init}}, \delta)$ and check if $\mathbf{s}$ leads to a valid run on $\mathcal{M}^\varphi$. The monitor $\mathcal{M}^\varphi$ can be viewed as a model with internal state set $q \in Q$, and the sequence $\mathbf{s}$ can be viewed as an N-behavior that may be generated by the model $\mathcal{M}^\varphi$, under some admissible nondeterministic transitions. The run-time verification procedure reduces to computing a set Q_N of the reachable states of $\mathcal{M}^\varphi$ after reading $s_1 s_2 \dots s_N$ and checking if $Q_N = \emptyset$ for some $N \leq M$. If yes, the anomaly (i.e., violation of φ) is claimed.

In the model invalidation based fault detection, we check if a sequence of observation $\{u_t, s_t, y_t\}_{t=1}^N$ is generated by a model described by Eq. (3.1), with internal state x that is analogue to q of a monitor, and with bounded uncertainty w, v that are analogue to the nondeterministic transition of the monitor. Very similar to the idea

[2]Often times in the literature, the term "monitor" are used to refer to the deterministic finite transition system that are determinzed from $\mathcal{M}$ via standard power set construction. Here we follow [30] and use the term "monitor" to refer to the nondeterministic finite transition system.

of run-time verification, the model invalidation reduces to checking the emptiness of a set X_N, which consists of the healthy system's reachable states that are consistent with observation $\{u_t, s_t, y_t\}_{t=1}^N$ under some admissible uncertainty sequence $\{w_t, v_t\}_{t=1}^N$. In fact, set X_N can be viewed as the projection (onto the internal state space) of a high dimensional polytope that is described exactly by the linear constraints in Eq. (3.2). However, unlike the case in the run-time verification where the internal state set Q_N is finite no matter what N is, X_N consists of infinite states and its representation complexity (i.e., the number of linear constraints required to describe X_N) may blow up as $N \to \infty$. This can be viewed as another interpretation of the issue that a growing horizon detector requires infinite memory. Hence we have to use the receding horizon scheme to compute an over approximation of X_N, whose description complexity is bounded. Detectability analysis tells us how tight this over approximation should be so that no fault is missed by the detector.

3.2 Problem Description

In this work we consider fault detectability analysis for switched affine systems whose mode sequences satisfy certain LTL constraints. To define the problem, we first define how the LTL-based side-information can be incorporated in the behavior description of the system. Let $\mathcal{S}$ be the healthy system and $\mathcal{S}^{\mathrm{f}}$ be the faulty system, and φ and φ^{f} be the corresponding LTL formulas. We assume the side-information to be of following form:

(i) if the fault never occurs, $\mathbb{x}, 1 \models \varphi$;

(ii) if the fault occurs at time t_{o}, then $s_1 s_2 \dots s_{t_{\mathrm{o}}-1}$ is a valid prefix of φ, and $\mathbb{x}, t_{\mathrm{o}} \models \varphi^{\mathrm{f}}$.

As mentioned previously, we collect the input-output pairs of length N in the receding horizon fault detection process. The above extra LTL constraints further restrict the sets of N-behaviors of the healthy and faulty systems, which now take the form in Eq. (3.9) and (3.10)

(3.9)

$$\mathfrak{B}_N(\mathcal{S},\varphi)=\left\{\{u_t,s_t,y_t\}_{t=1}^N \;\middle|\; \begin{array}{c}\exists\{x_t,w_t,v_t\}_{t=1}^N\in(X\times W\times V)^N:\\ \{u_t,v_t,w_t,x_t,y_t,s_t\}_{t=1}^N \text{ satisfy } \mathcal{S}\text{'s dynamics}\\ \boxed{\exists \mathbf{p}\in\Sigma^*,\mathbb{w}\in\Sigma^\omega:\mathbf{p}s_1s_2\dots s_N\mathbb{w}\models\varphi}\end{array}\right\},$$

(3.10)

$$\mathfrak{B}_N(\mathcal{S}^{\mathrm{f}},\varphi^{\mathrm{f}})=\left\{\{u_t,s_t,y_t\}_{t=1}^N \;\middle|\; \begin{array}{c}\exists\{x_t,w_t,v_t\}_{t=1}^N\in(X^{\mathrm{f}}\times W^{\mathrm{f}}\times V^{\mathrm{f}})^N:\\ \{u_t,v_t,w_t,x_t,y_t,s_t\}_{t=1}^N \text{ satisfy } \mathcal{S}^{\mathrm{f}}\text{'s dynamics}\\ \boxed{\exists\mathbb{w}\in\Sigma^\omega:s_1s_2\dots s_N\mathbb{w}\models\varphi^{\mathrm{f}}}\end{array}\right\}.$$

The key difference between the definitions of $\mathfrak{B}_N(\mathcal{S},\varphi)$ and $\mathfrak{B}_N(\mathcal{S}^{\mathrm{f}},\varphi^{\mathrm{f}})$ is regarding the constraints on the N-sequence of modes, which are highlighted with the boxes. Fig. 3.2 shows an illustration that may help understanding this difference. The blue line represents the switching sequence when the system is always healthy, whereas the dashed red line represents the switching sequence assuming the fault occurs at time t_{o}. The shaded region highlights the switching mode sequence within time window $[\![t_{\mathrm{o}},t_{\mathrm{o}}+N-1]\!]$, and our goal is to check if the behavior generated by the healthy system under the blue shaded mode sequence differs from the behavior generated by the red faulty system under the red shaded mode sequence. Since we require $\mathbb{x},t_{\mathrm{o}}\models\varphi^{\mathrm{f}}$, this suggests that the most recent N-segment of the mode sequence (the red shaded area) must be a valid prefix of φ^{f}. This hence leads to the boxed condition in Eq. (3.10). On the other hand, we require $\mathbb{x},1\models\varphi$, the N-segment represented by the blue shaded region can be completed into a valid prefix of φ by

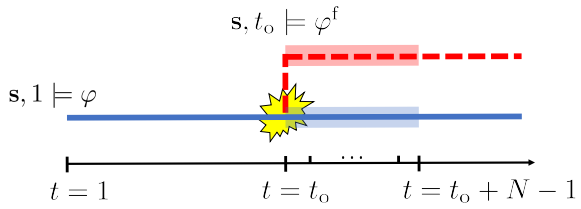

Figure 3.2: Illustration of the timeline in the healthy and faulty case.

adding $\mathbf{p} \in \Sigma^*$ in the front, which leads to the condition marked by the box in Eq. (3.9).

We now formally state the detectability analysis problem.

Problem 1. *Assume the following are given:*

(i) a healthy system $\mathcal{S}$ and a faulty system $\mathcal{S}^\mathrm{f}$, both of which have switched affine dynamics in form of Eq. (3.1)*,*

(ii) LTL formulas φ and φ^f that govern the switching mode sequences of the healthy and the faulty system,

(iii) a positive integer N,

determine whether $\mathfrak{B}_N(\mathcal{S}, \varphi) \cap \mathfrak{B}_N(\mathcal{S}^\mathrm{f}, \varphi^\mathrm{f}) = \emptyset$.

As discussed in Section 3.1.1, the minimal horizon length $T := \min\left\{N \mid \mathfrak{B}_N(\mathcal{S}, \varphi) \cap \mathfrak{B}_N(\mathcal{S}^\mathrm{f}, \varphi^\mathrm{f}) = \emptyset\right\}$ can be found through a line search over N, starting from $N = 1$. The usefulness of studying Problem 1 is that the extra LTL constraints may lead to a smaller T compared to the detectability analysis without such constraints. This is because these LTL constraints further restrict the behavior set so that the healthy and faulty behaviors differ earlier. We state this result with the following proposition.

Propoition 12. Let $T_1 := \min\left\{N \mid \mathfrak{B}_N(\mathcal{S}, \varphi) \cap \mathfrak{B}_N(\mathcal{S}^\mathrm{f}, \varphi^\mathrm{f}) = \emptyset\right\}$ and $T_2 := \min\left\{N \mid \mathfrak{B}_N(\mathcal{S}) \cap \mathfrak{B}_N(\mathcal{S}^\mathrm{f}) = \emptyset\right\}$, we have $T_1 \leq T_2$.

Proof. By definition (see Eq. (3.3), (3.9), (3.10)), we have $\mathfrak{B}_N(\mathcal{S},\varphi) \subseteq \mathfrak{B}_N(\mathcal{S})$ and $\mathfrak{B}_N(\mathcal{S}^{\mathrm{f}},\varphi^{\mathrm{f}}) \subseteq \mathfrak{B}_N(\mathcal{S}^{\mathrm{f}})$. This means $\mathfrak{B}_N(\mathcal{S}) \cap \mathfrak{B}_N(\mathcal{S}^{\mathrm{f}}) = \emptyset \Rightarrow \mathfrak{B}_N(\mathcal{S},\varphi) \cap \mathfrak{B}_N(\mathcal{S}^{\mathrm{f}},\varphi^{\mathrm{f}}) = \emptyset$. Hence $\big\{N \mid \mathfrak{B}_N(\mathcal{S}) \cap \mathfrak{B}_N(\mathcal{S}^{\mathrm{f}}) = \emptyset\big\} \subseteq \big\{N \mid \mathfrak{B}_N(\mathcal{S},\varphi) \cap \mathfrak{B}_N(\mathcal{S}^{\mathrm{f}},\varphi^{\mathrm{f}}) = \emptyset\big\}$, which implies $T_1 \leq T_2$. □

3.3 Solution Approach

In this section, we present a solution to Problem 1. The main challenge is to express the condition in the boxes in Eq. (3.9), (3.10) in a way that can be easily integrated in the MILP in Eq. (3.8). Note that MILP encoding of bounded LTL [58] is not applicable to impose the boxed constraints in Eq. (3.9). Our solution is to first transfer the LTL formula into a monitor that captures the boxed conditions in Eq. (3.9), (3.10) induced from the given LTL formula. We then convert the monitor into its boolean representation that can be easily expressed as mixed integer linear constraints.

3.3.1 Monitor and System Behavior Constraints

We first connect the constraints marked by the boxes in Eq. (3.9), (3.10) with a monitor.

Let φ, φ^{f} be the LTL formulas from Eq. (3.9), (3.10), and let $\mathcal{M}^{\varphi}$, $\mathcal{M}^{\varphi^{\mathrm{f}}}$ be the associated monitors. The condition marked by the box in Eq. (3.10) says that $s_1 s_2 \dots s_N$ is not a bad prefix of φ^{f}, i.e., $s_1 s_2 \dots s_N$ can be generated by $\mathcal{M}^{\varphi^{\mathrm{f}}}$. On the other hand, the boxed condition in Eq. (3.9) says that $s_1 s_2 \dots s_N$ can be "completed" by adding a finite prefix $\mathbf{p} \in \Sigma^*$ in the front so that $\mathbf{p} s_1 s_2 \dots s_N$ can be generated by $\mathcal{M}^{\varphi}$. This suggests that $s_1 s_2 \dots s_N$ can be generated by another monitor $\mathcal{M}^{\varphi'}$, which is exactly the same as $\mathcal{M}^{\varphi}$ except for the initial conditions. We formally state

this fact with the following proposition.

Propoition 13. Given an LTL formula φ and $\mathcal{M}^{\varphi} = (\Sigma, Q, Q_{\text{init}}, \delta)$, the monitor that recognizes the valid prefixes of φ, assume that all states in Q are reachable from Q_{init}[3], the following are equivalent:

(i) there exist $\mathbf{p} \in \Sigma^*$, $\mathbb{w} \in \Sigma^{\omega}$ such that $\mathbf{p}s_1 s_2 \dots s_N \mathbb{w} \models \varphi$;

(ii) there exists $q_1 q_2 \dots q_{N+1} \in Q^{N+1}$ such that $q_1 \in Q$, $q_{t+1} = \delta(q_t, s_t)$ for all $t \in [\![1, N+1]\!]$.

and the following are equivalent:

(iii) there exist $\mathbb{w} \in \Sigma^{\omega}$ such that $s_1 s_2 \dots s_N \mathbb{w} \models \varphi$;

(vi) there exists $q_1 q_2 \dots q_{N+1} \in Q^{N+1}$ such that $q_1 \in Q_{\text{init}}$, $q_{t+1} = \delta(q_t, s_t)$ for all $t \in [\![1, N+1]\!]$.

3.3.2 MILP Encoding of Monitor

We present a technique to encode a monitor with mixed integer linear constraints. The idea is to use Proposition 13 to convert the two boxed constraints w.r.t φ and φ^{f} from Eq. (3.9), (3.10) into two monitors, and then write the monitors in their boolean representations and convert the boolean representations into two sets of MILP constraints. Since the encoding is for the nondeterministic monitor directly, it does not require determinizing the monitor with the power set construction and hence avoids an unnecessarily large MILP.

Let $\mathcal{M}^{\varphi} = (\Sigma, Q, Q_{\text{init}}, \delta)$ be a monitor of LTL formula φ. At each time instant t, we associate each state $q_i \in Q$ with a binary variable b_i^t, which takes value 1 if the state of $\mathcal{M}^{\varphi}$ is equal to q_i at time t and takes value 0 otherwise. Similarly,

[3]This assumption can be made without loss of generality because states in Q that are not reachable from Q_{init} can be removed without changing the sequences generated by $\mathcal{M}$.

we associate each letter $\sigma_k \in \Sigma$ with a binary variable a_k^t that takes value 1 iff the monitor reads letter σ_k at time t. To guarantee that the monitor's state (or the read letter) exists and is unique at any time, we impose the following constraint:

$$\forall t \in [\![1, N+1]\!] \ : \ \sum\nolimits_{i=1}^{|Q|} b_i^t = 1, \tag{3.11}$$

$$\forall t \in [\![1, N]\!] \ : \ \sum\nolimits_{k=1}^{|\Sigma|} a_k^t = 1. \tag{3.12}$$

Moreover, the state indicator b_i^{t+1} must update according to the transition relation δ of the monitor. To this end, we require the following constraints to hold:

$$\forall t \in [\![1, N]\!], i \in [\![1, |Q|]\!] \ : \ b_i^{t+1} \leq \sum_{j,k:q_i \in \delta(q_j,\sigma_k)} p_{ijk}^t, \tag{3.13}$$

where p_{ijk}^t is a binary variable satisfying:

$$\forall t \in [\![1, N]\!], i, j \in [\![1, |Q|]\!], k \in [\![1, |\Sigma|]\!]$$

$$\text{such that } q_i \in \delta(q_j, \sigma_k) :$$

$$1 + p_{ijk}^t \geq b_j^t + a_k^t, \quad p_{ijk}^t \leq b_j^t, \quad p_{ijk}^t \leq a_k^t. \tag{3.14}$$

It might be useful to point out that Eq. (3.14) forces $p_{ijk}^t = b_j^t \wedge a_k^t$. In fact, variable p_{ijk}^t can be viewed as an indicator that takes value 1 iff there is a chance that the monitor's state is taken to q_i (at time $t+1$) from q_j, by reading letter σ_k at time t. Then Eq. (3.13) guarantees that b_i^{t+1} is set to 1 only if there is such a chance for the state to be equal to q_i at time $t+1$. Note that b_i^{t+1} can still be zero if some $p_{ijk}^t = 1$, but Eq. (3.12) and (3.13) together guarantee that there must be one $i' \in \{i \mid \exists j, k : q_i \in \delta(q_j, \sigma_k),\ p_{ijk}^t = 1\}$ such that $b_{i'}^{t+1} = 1$. This hence captures the nondeterministic transition relation of the nondeterministic monitor $\mathcal{M}^\varphi$.

Note that if a transition of $\mathcal{M}^\varphi$ is labeled as "True", i.e., for all $\sigma_k \in \Sigma, q_i \in \delta(q_j, \sigma_k)$, then the constraint in Eq. (3.14) can be simply replaced by $p_{ijk}^t = b_i^t$.

Finally, we constrain that the initial states are from Q_{init}:

$$\sum_{i:q_i \in Q_{\text{init}}} b_i^1 = 1. \tag{3.15}$$

The correctness of the construction so far is summarized with the following proposition, which can be easily verified using Proposition 13.

Propoition 14. Let φ be an LTL formula over mode set Σ and $\mathcal{M}^\varphi = (\Sigma, Q, Q_{\text{init}}, \delta)$ be its monitor. For a finite word $s_1 s_2 \dots s_N \in \Sigma^*$, assume that binary variable a_k^t is such that $a_k^t = 1$ iff $s_t = \sigma_k$, then the following are equivalent:

(i) there exists $q_1 q_2 \dots q_{N+1} \in Q^{N+1}$ such that $q_1 \in Q$, $q_{t+1} = \delta(q_t, s_t)$ for all $t \in [\![1, N+1]\!]$;

(ii) there exist binary variables b_i^t, p_{ijk}^t such that together with a_k^t, Eq. (3.11)-(3.14) hold,

and the following are equivalent:

(iii) there exists $q_1 q_2 \dots q_{N+1} \in Q^{N+1}$ such that $q_1 \in Q_{\text{init}}$, $q_{t+1} = \delta(q_t, s_t)$ for all $t \in [\![1, N+1]\!]$.

(vi) there exist binary variables b_i^t, p_{ijk}^t such that together with a_k^t, Eq. (3.11)-(3.15) hold.

Remark 2. Note that if φ is in the form of conjunction of several shorter formulas φ_i, i.e., $\varphi = \bigwedge_i \varphi$, the overall encoding can be done by stacking the mixed integer linear constraints derived from each $\mathcal{M}^{\varphi_i}$. This may not reduce the size of MILP formulation, but is useful when the size of the monitor for the overall φ is too large and generating the monitor becomes the bottleneck.

3.3.3 Detectability Analysis Augmented with LTL Constraints

Let φ and φ^{f} be the LTL formulas that constrain the mode sequences of the healthy and faulty system respectively. Denote the constraints in Eq. (3.12)-(3.14) that is derived from φ by

$$\mathbf{G}_N^{\varphi}\left(\{a_k^t\}_{k=1,t=1}^{|\Sigma|,N}, \boldsymbol{\eta}\right) \leq 0, \tag{3.16}$$

where $\boldsymbol{\eta}$ is a vector obtained by stacking auxiliary binary variable b_i^t and p_{ijk}^t together, and $\mathbf{G}_N^{\varphi}$ is an affine function that depends on φ and N. Similarly, we denote the constraints in Eq. (3.12)-(3.15) that are derived from φ^{f} by

$$\mathbf{G}_N^{\varphi^{\mathrm{f}}}\left(\{a_k^t\}_{k=1,t=1}^{|\Sigma|,N}, \boldsymbol{\eta}^{\mathrm{f}}\right) \leq 0. \tag{3.17}$$

The MILP used for detectability analysis with LTL constraints can be then formulated. That is, $\mathfrak{B}_N(\mathcal{S},\varphi)\cap\mathfrak{B}_N(\mathcal{S}^{\mathrm{f}},\varphi^{\mathrm{f}})=\emptyset$ is equivalent to the following MILP being infeasible:

$$\begin{aligned}
\text{find}\quad & \left\{u_t, \{a_k^t\}_{k=1}^{|\Sigma|}, y_t, x_t, x_t^{\mathrm{f}}, w_t, w_t^{\mathrm{f}}, v_t, v_t^{\mathrm{f}}\right\}_{t=1}^{N}, \\
& \boldsymbol{\xi}_{\mathrm{ob}}, \boldsymbol{\xi}_{\mathrm{un}}, \boldsymbol{\xi}_{\mathrm{un}}^{\mathrm{f}}, \boldsymbol{\eta}, \boldsymbol{\eta}^{\mathrm{f}} \\
\text{s.t.}\quad & \mathbf{H}_N^{\mathcal{S}}\left(\left\{u_t, \{a_k^t\}_{k=1}^{|\Sigma|}, y_t, x_t, w_t, v_t, \right\}_{t=1}^{N}, \boldsymbol{\xi}_{\mathrm{ob}}, \boldsymbol{\xi}_{\mathrm{un}}\right) \leq 0, \\
& \mathbf{H}_N^{\mathcal{S}^{\mathrm{f}}}\left(\left\{u_t, \{a_k^t\}_{k=1}^{|\Sigma|}, y_t, x_t^{\mathrm{f}}, w_t^{\mathrm{f}}, v_t^{\mathrm{f}}, \right\}_{t=1}^{N}, \boldsymbol{\xi}_{\mathrm{ob}}, \boldsymbol{\xi}_{\mathrm{un}}^{\mathrm{f}}\right) \leq 0, \\
& \mathbf{G}_N^{\varphi}\left(\{a_k^t\}_{k=1,t=1}^{|\Sigma|,N}, \boldsymbol{\eta}\right) \leq 0, \\
& \mathbf{G}_N^{\varphi^{\mathrm{f}}}\left(\{a_k^t\}_{k=1,t=1}^{|\Sigma|,N}, \boldsymbol{\eta}^{\mathrm{f}}\right) \leq 0, \\
& a_k^t \in \{0,1\}, \forall t \in [\![1,N]\!], k \in [\![1,|\Sigma|]\!], \\
& \boldsymbol{\eta} \in \{0,1\}^{|\boldsymbol{\eta}|}, \boldsymbol{\eta}^{\mathrm{f}} \in \{0,1\}^{|\boldsymbol{\eta}^{\mathrm{f}}|},
\end{aligned} \tag{3.18}$$

where $|\boldsymbol{\eta}|$ and $|\boldsymbol{\eta}^{\mathrm{f}}|$ are the length of vectors $\boldsymbol{\eta}$ and $\boldsymbol{\eta}^{\mathrm{f}}$ respectively.

3.3.4 Run-time Fault Detection

With the detectability analysis technique presented above, we can determine the minimal N so that $\mathfrak{B}_N(\mathcal{S},\varphi)\cap\mathfrak{B}_N(\mathcal{S}^{\mathrm{f}},\varphi^{\mathrm{f}})=\emptyset$. As a result of such analysis, we only need to check whether the latest collected $\{u_t,s_t,y_t\}_{t=t_0-N+1}^{t_0}\in\mathfrak{B}_N(\mathcal{S},\varphi)$ at the current time t_0 and claim anomaly iff this does not hold. To this end, it is sufficient to run the monitor $\mathcal{M}^{\varphi}$ (as described in Section 3.1.3) and the model invalidation LP (Eq. (3.2)) with horizon N in parallel. If no fault occurs, the monitor keeps running with current state set $Q_{t_0}\neq\emptyset$ and the model invalidation keeps being feasible, and no anomaly is claimed in this case. If a fault occurs at time t_{o}, either the switching sequence turns into a bad prefix of φ before time instant $t_{\mathrm{o}}+N-1$ and the monitor detects violation of φ immediately, or the switching sequence is still a valid prefix of φ up until time instant $t_{\mathrm{o}}+N-1$, which validates the boxed condition in Eq. (3.9) and hence the model invalidation LP must turn infeasible at time $t_0=t_{\mathrm{o}}+N-1$ by the N-behavior isolation of the faulty and healthy systems. In other words, any fault is detected with at most N-delay without any false alarm.

3.4 Case Study: UAV Altitude Consensus

We use an unmanned aerial vehicle (UAV) altitude consensus protocol to demonstrate the proposed detectability analysis technique. We say that a set of UAVs reaches altitude consensus if their altitude eventually converge to the same value. There are several consensus protocols based on local communication. In particular, we assume the UAVs implement the nearest neighbor rules from [54]. Under this protocol and assuming single integrator dynamics for vertical motion, the altitude dynamics of the UAVs can be modeled as follows. We let $x=[x^1,x^2,\ldots,x^8]^T\in\mathbb{R}^8$ be the state where x^i is the altitude of the i^{th} UAV. We assume that a leader UAV, indexed by

1, reaches a set point while the other UAVs adjust their own altitude according the nearest neighbor protocol [54], induced from the UAVs' communication topologies shown in Fig. 3.4 (Left). Let $A_{\sigma_k}, K_{\sigma_k}$ be the system matrices representing the UAVs dynamics while implementing this protocol. The i^{th} rows of matrices $A_{\sigma_1}, K_{\sigma_1}$ take the following form:

1) $i = 1$, $(A_{\sigma_1})_{11} = 0.9$ and $(A_{\sigma_1})_{1j} = 0$ for $j \in [\![2, 8]\!]$, $(K_{\sigma_1})_1 = 0.3$ this leads to a dynamics that guarantees the leader altitude to converge a set point at $x^1 = 3$ (i.e., $x^1_{t+1} = 0.9x^1_t + 0.3$);

2) $i \neq 1$, the i^{th} UAV update x^i by averaging its own height and those of its neighbors, i.e., $(K_{\sigma_1})_i = 0$ and

$$(A_{\sigma_1})_{ij} = \begin{cases} \frac{1}{1+d(i)} & \text{if } i \text{ connects } j \text{ in topology 1} \\ 0 & \text{otherwise} \end{cases}, \tag{3.19}$$

where $d(i)$ is the number of edges incident to node i.

Similarly, we define A_{σ_2}, K_{σ_2} for topology 2 with the same leader UAV set point (i.e., $x^1 = 3$); and define A_{σ_3}, K_{σ_3} (and A_{σ_4}, K_{σ_4}, respectively) for topology 1 (and topology 2, respectively) with the same leader UAV dynamics but a different set point at $x^1 = 6$.

We assume the changes in the communication topology and fault detection scheme (high-level decisions) run at a slower timescale than the consensus dynamics (low-level control) – i.e., 15 times slower. Then the dynamics relevant for fault detection can be written as the following switched affine system (denoted by $\mathcal{S}_{\text{UAV}}$ in the rest of the paper):

$$x_{t+1} = \overline{A}_{\sigma_i} x_t + \overline{K}_{\sigma_i} + \overline{E}_{\sigma_i} w_t \tag{3.20}$$

where $s_t \in \Sigma := \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, and

$$\overline{A}_{\sigma_i} = A_{\sigma_i}^{15}, \quad \overline{K}_{\sigma_i} = \sum\nolimits_{t=1}^{15} A_{\sigma_i}^{t-1} K_{\sigma_i}$$

$$\overline{E}_{\sigma_i} = [A_{\sigma_i}^{14}, A_{\sigma_i}^{13}, \ldots, A_{\sigma_i}, I]. \tag{3.21}$$

Note that in this setting there is no continuous control input u_t, and for simplicity we assume that the output $y_t = x_t$ with no measurement noise v_t. Finally, we assume that $x_t \in X := \{x \in \mathbb{R}^8 \mid 0 \leq x^i \leq 7,\ \forall i \in [\![1, 8]\!]\}$ and disturbance $w_t \in W := \{w \in \mathbb{R}^{120} \mid -0.1 \leq w^i \leq 0.1,\ \forall i \in [\![1, 120]\!]\}$.

The faulty system model $\mathcal{S}_{\text{UAV}}^{\text{f}}$ we analyze results from a failure in the communication links between nodes 3-5 and 4-6 in topology 2, changing the system matrices A_{σ_2} and A_{σ_4} (both induced by topology 2) and the corresponding $\overline{A}_{\sigma_2}$, $\overline{K}_{\sigma_2}$, $\overline{A}_{\sigma_4}$, $\overline{K}_{\sigma_4}$ in Eq. (3.21). Note that the healthy-faulty system pair $(\mathcal{S}_{\text{UAV}}, \mathcal{S}_{\text{UAV}}^{\text{f}})$ is not N-detectable for any finite N because the fault will never be detected unless the system switch to mode σ_2 or σ_4. However, we know that switching to mode σ_2 or σ_4 infinitely often is required to achieve consensus because communication topology 1 is not a connected graph. We can hence incorporate this information in the detectability analysis to compute worst-case detection delay.

We assume that the mode sequence satisfies the LTL formula $\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3$ under the healthy configuration, where

$$\varphi_1 = \bigwedge_{(k,l)\in\{(1,2),(3,4)\}} \square \left(\left(\bigwedge_{t=0}^{19} \bigcirc^t (\sigma_k \vee \sigma_l) \right) \rightarrow \bigcirc^{20} \neg (\sigma_k \vee \sigma_l) \right), \tag{3.22}$$

$$\varphi_2 = \bigwedge_{(k,l)\in\{(1,2),(3,4)\}} \square \left(\left(\neg(\sigma_k \vee \sigma_l) \wedge \bigcirc(\sigma_k \vee \sigma_l) \right) \rightarrow \bigwedge_{t=2}^{7} \bigcirc^t (\sigma_k \vee \sigma_l)) \right), \tag{3.23}$$

$$\varphi_3 = \bigwedge_{(k,l)\in\{(1,3),(2,4)\}} \square \Big((\sigma_k \vee \sigma_l) \vee \bigcirc(\sigma_k \vee \sigma_l) \vee \bigcirc^2 (\sigma_k \vee \sigma_l) \Big). \tag{3.24}$$

Formula φ_1 and φ_2 together restrict the dwell time for set point changes by the leader UAVs to be within $[\![7, 20]\!]$, while formula φ_3 assures that each of the two

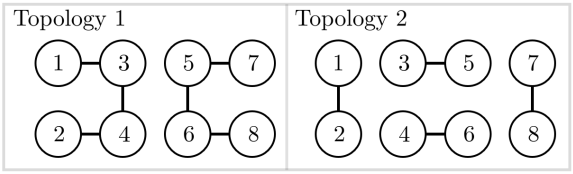

Figure 3.3: Communication topologies of the UAVs, where circles represent UAVs with their indices.

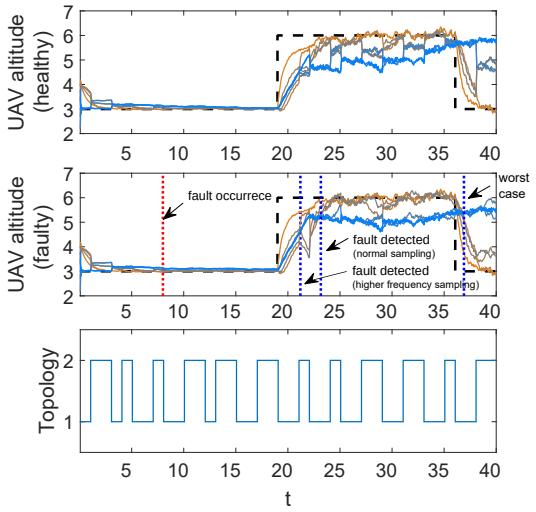

Figure 3.4: Fault detection of the UAV consensus system at run time.

communication topologies are used within every three time steps. They together guarantee enough time and communication for convergence to a consensus. We also assume that the mode sequence does not need to satisfy any LTL formula after the fault occurs, however in the example we choose the mode sequence after the fault to be consistent with φ, therefore a pure discrete monitor will not be able to detect this fault without taking continuous dynamics into account. Gurobi [88] is used to solve the obtained MILP. The obtained minimal length of horizon $T = 30$, which is finite and this result agrees with Proposition 12.

Fig. 3.4 (Right) shows the fault detection results. The upper plot shows the altitude of the eight UAVs (solid lines) and the set point profile (dashed black line) when no fault occurs. The middle plot shows the same set of trajectories of the faulty system. The lower plot shows the alternating of the two communication topologies.

One can check that the given LTL formula φ is satisfied (even after fault). In this illustration, the fault occurs at time $t = 8$, at which time the consensus is already achieved (with set point at 3). Hence the fault does not lead to behavior isolation immediately. However, the fault is detected later at time $t = 23$ after the set point of the leader changes. The detection delay is 15, which is shorter than the delay bound $T = 30$. This experiment hence agrees with the theory. We also run the model invalidation at a higher frequency, using the timescale of the dynamics, for faster detection though this comes at the expense of solving LPs of larger size (15 times larger) and more often.

CHAPTER IV

Graceful Degradation of Faulty Systems

This is a short chapter that formally describes the problem studied in this work. The problem is defined by two ingredients: a system model and a specification. We first introduce a special type of hierarchical hybrid system, called control systems with fault configurations, to model the system whose dynamics experiences a sudden change due to faults. We then introduce two slightly different LTL formula that specify the so called graceful degradation of such systems' desired behavior. The difference between the two LTL specification are briefly discussed.

4.1 Control System with Fault Configurations

A control system Σ is a six tuple (X, U, D, f, AP, L_X). The continuous-time dynamics is define by

$$\dot{x} = f(x, u, d), \tag{4.1}$$

where $x \in X \subseteq \mathbb{R}^n$ is the state, $u \in U$ is the control, and $d \in D \subseteq \mathbb{R}^p$ is the disturbance. Note that control u can be a combination of discrete actions and continuous control input, which leads to a hybrid system. Similarly, a discrete-time system can be defined by replacing Eq. (4.1) with $x^+ = f(x, u, d)$ where x^+ is the updated state

at the next time instant. Set AP consists of the atomic propositions of interest, and $L_X : X \to 2^{AP}$ is the labeling map. We use AP and L_X to describe the behaviors of system Σ (see Section 2.2.1).

Let $F = \{\phi^{[1]}, \ldots, \phi^{[M]}\}$ be a finite set, each element $\phi^{[i]}$ is called of a *faulty mode* (or a *fault* for short). A partial order $\preceq$ is defined on set F to capture the severity of different faults in F. That is, $\phi^{[i]} \preceq \phi^{[j]}$ means that fault $\phi^{[j]}$ is more severe than or equal to fault $\phi^{[i]}$, and $\phi^{[i]} \prec \phi^{[j]}$ means that fault $\phi^{[j]}$ is strictly more severe. The tuple $(F, \preceq)$ is called the a *fault configuration*. We define the set of minimum elements of $E \subseteq F$ to be

$$\min(E) := \{\phi^{[j]} \in E \mid \nexists \phi^{[i]} \in E \text{ s.t. } \phi^{[i]} \prec \phi^{[j]}\}, \tag{4.2}$$

and $\max(E)$ can be defined in a similar way. We will assume F always has a unique minimum element that represents the healthy configuration. By convention we always denote this healthy configuration by $\phi^{[1]}$. Finally we define the successors of a fault $\phi^{[i]} \in F$ to be

$$\text{succ}(\phi^{[i]}) := \min\big(\{\phi^{[j]} \in F \mid \phi^{[i]} \prec \phi^{[j]}\}\big). \tag{4.3}$$

By definition, fault $\phi^{[j]}$ is a successor of fault $\phi^{[i]}$ if $\phi^{[j]}$ is more sever than $\phi^{[i]}$ and there are no other faults in between.

A *control system with fault configurations*, denoted by $\Sigma^{\text{F}} = (F, \preceq, \mathcal{S}, AP^{\text{F}}, L_X^{\text{F}})$, is a hierarchical system where

- $(F, \preceq)$ is a fault configuration.
- $\mathcal{S}$ is a set of sub systems $\Sigma^{[i]} = (X, U^{[i]}, D^{[i]}, f^{[i]}, AP, L_X)$, each associated with a fault $\phi^{[i]} \in F$. In particular, we assume that all the subsystems share a common state domain X, atomic proposition set AP and labeling function L_X,

but different subsystems can have different control input set $U^{[i]}$ and disturbance set $D^{[i]}$ to capture the loss of control authority and extreme operating condition under different faulty modes.

- $AP^{\mathrm{F}} = AP \cup \{\pi^{[1]}, \ldots, \pi^{[M]}\}$, where $\pi^{[i]}$ is an auxiliary atomic proposition associated with fault $\phi^{[i]}$.

- $L_X^{\mathrm{F}} : X \times F \to 2^{AP \cup \{\pi^{[1]}, \ldots, \pi^{[M]}\}}$ is a labeling map such that

$$L_X^{\mathrm{F}}(x, \phi^{[i]}) = \pi^{[i]} \cup L_X(x), \tag{4.4}$$

which is consistent with L_X and allows one to associate atomic proposition $\pi^{[i]}$ with fault $\phi^{[i]}$.

The state of Σ^{F} is a pair (x, ϕ), where $\phi \in F$ indicates the faulty status and x updates according to the subsystem associated with fault ϕ, i.e.,

$$\dot{x} = f^{[i]}(x, u, d) \qquad \text{if} \qquad \phi = \phi^{[i]}. \tag{4.5}$$

The evolution of the fault status ϕ is governed by order $\preceq$, i.e., ϕ may change from $\phi^{[i]}$ to $\phi^{[j]}$ if and only if $\phi^{[j]} \in \{\phi^{[i]}\} \cup \mathrm{succ}(\phi^{[i]})$. A change is called *nontrivial* if $\phi^{[j]} \in \mathrm{succ}(\phi^{[i]})$. Note that the fault status ϕ either maintains to be the current governing system or transits into its successors. This means two things: first, the faults are permanent, i.e., the system will never recover once the faults occur; secondly, the system never "goes down" more than two levels at once.

Example 1. We illustrate the concepts by an engine thermal management system, whose dynamics is defined by

$$\begin{cases} \dot{T}_{\mathrm{e}} &= c_1(T_{\mathrm{e}} - T_{\mathrm{a}}) + c_2 vw(T_{\mathrm{e}} - T_{\mathrm{r}}) + c_3 h \\ \dot{T}_{\mathrm{r}} &= (c_4 + c_5 sg)(T_{\mathrm{r}} - T_{\mathrm{a}}) + c_6 vw(T_{\mathrm{r}} - T_{\mathrm{e}}), \end{cases} \tag{4.6}$$

where the coefficients $c_1, \ldots, c_6$ are known constants and the variables are defined in Table 4.1

Table 4.1: Definition of Variables

Symbol	Physical Meaning	Unit	Range Used
T_{e}	Engine temperature	K	[290, 410]
T_{r}	Radiator temperature	K	[290, 410]
v	Flow valve position	-	{0.25,1}
g	Radiator grill shutter opening	-	{0.25,1}
h	Heat from engine combustion	W	[15000, 19000]
s	Vehicle speed	m/s	[10,20]
w	Coolant pump flow rate	kg/s	[0.03,0.045]
T_{a}	Ambient temperature	K	[282, 288]

The fault configuration considered in this example contains four faults, i.e., $F = \{\phi^{[1]}, \phi^{[2]}, \phi^{[3]}, \phi^{[4]}\}$. Each fault $\phi^{[1]}$ is associated with a subsystem $\Sigma^{[i]} = (X, U^{[i]}, D^{[i]}, f^{[i]})$. For simplicity, we omit AP, AP^{F}, L_X and L_X^{F}. In this example, state $x = [T_{\mathrm{e}}, T_{\mathrm{r}}]^T$ and $X = [290, 410] \times [290, 410]$. The vector field $f^{[i]}$ is given by Eq. (4.1), but the role of the variables could change, i.e., some variables can be control input in one faulty mode but becomes a disturbance in some other faulty modes. The description for each fault and associated subsystem is defined as follows:

(1) $\phi^{[1]}$: the system is healthy,

$u = [v, g]^T$, $U^{[1]} = \{0.25, 1\} \times \{0.25, 1\}$,

$d = [h, s, w, T_{\mathrm{a}}]^T$, $D^{[1]} = [15000, 19000] \times [10, 20] \times [0.03, 0.045] \times [282, 288]$.

(2) $\phi^{[2]}$: the coolant flow valve is stuck at an unknown position from 0.25 to 1.

$u = g$, $U^{[2]} = \{0.25, 1\}$,

$d = [h, s, w, T_{\mathrm{a}}, v]^T$, $D^{[2]} = [15000, 19000] \times [10, 20] \times [0.03, 0.045] \times [282, 288] \times [0.25, 1]$.

(3) $\phi^{[3]}$: the radiator grill is stuck at an unknown position from 0.25 to 1.

$u = v$, $U^{[3]} = \{0.25, 1\}$,

$d = [h, s, w, T_{\mathrm{a}}, g]^T$, $D^{[3]} = [15000, 19000] \times [10, 20] \times [0.03, 0.045] \times [282, 288] \times$

$[0.25, 1]$.

(4) $\phi^{[4]}$: both the coolant flow valve and the radiator grill are stuck at an unknown positions from 0.25 to 1.

$U^{[4]} = \emptyset$,

$d = [h, s, w, T_{\mathrm{a}}, v, g]^T$, $D^{[4]} = [15000, 19000] \times [10, 20] \times [0.03, 0.045] \times [282, 288] \times [0.25, 1] \times [0.25, 1]$.

Finally, the partial order defined on F is visualized by the lattice diagram in Figure (4.1). The arrow in the diagram marks to the possible nontrivial transition between the fault status, i.e., there is an arrow from $\phi^{[i]}$ to $\phi^{[j]}$ iff $\phi^{[j]} \in \mathrm{succ}(\phi^{[i]})$. In this example, It is a natural choice to define $\phi^{[1]} \prec \phi^{[2]} \prec \phi^{[4]}$ and $\phi^{[1]} \prec \phi^{[3]} \prec \phi^{[4]}$, while $\phi^{[2]}$ and $\phi^{[3]}$ are not comparable.

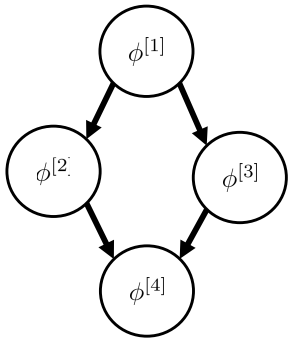

Figure 4.1: Visualization of the partial order defined on F in Example 1.

4.2 Graceful Degradation of Faulty System

In this part we give two LTL formula that defines slightly different graceful degradation of a control system with fault configurations.

Let $F = \{\phi^{[1]}, \ldots, \phi^{[M]}\}$ be the collection of the faults under consideration, and let $AP \cup \{\pi^{[1]}, \ldots, \pi^{[M]}\}$ be the overall set of atomic propositions. The two LTL

formulas that specify the graceful degradation are given by

$$\Phi = \bigwedge_{i:\phi^{[i]}\in F} \left(\Diamond\Box\pi^{[i]} \to \varphi^{[i]}\right), \tag{4.7}$$

$$\Psi = \bigvee_{i:\phi^{[i]}\in F} \left(\neg\pi^{[i]}\,\mathcal{U}\ \left(\Box\pi^{[i]} \wedge \varphi^{[i]}\right)\right), \tag{4.8}$$

where $\varphi^{[i]}$ is an LTL formula over AP, specifying the system's desired behavior when the final fault configuration is $\phi^{[i]} \in F$.

- Eq. (4.7) says: if the fault status eventually stays at $\phi^{[i]}$, the specification $\varphi^{[i]}$ associated with this fault is achieved starting from the very beginning. Note that the fault status of a faulty system is guaranteed to reach a specific configuration $\phi^{[i]} \in F$ and stays there forever. This is because fault set F is finite and a fault only transits into its successors in F, hence there can be only finitely many transitions.
- Eq. (4.8) says: if the fault status eventually stays at $\phi^{[i]}$, specification $\varphi^{[i]}$ should be satisfied immediately after the occurrence of fault $\phi^{[i]}$.

Typically $\varphi^{[j]}$ is chosen to be less stringent than $\varphi^{[j]}$ if $\phi^{[i]} \prec \phi^{[j]}$ (i.e., $\mathbb{w} \models \varphi^{[j]} \to \mathbb{w} \models \varphi^{[i]}$), in which case the LTL formula Φ and Ψ capture a graceful degradation in the system performance.

The difference between Φ and Ψ are resulted from the fact that a word $\mathbb{w}$ satisfying an LTL formula is different from its suffix $\mathbb{w}_t$ satisfying the formula (see the syntax of LTL in Section 2.2.1). Depending on the application, we can either require the entire word $\mathbb{w}$ to satisfy $\varphi^{[i]}$ associated with the final fault (captured by Φ), or only require the suffix $\mathbb{w}_t$ to satisfy $\varphi^{[i]}$ where t is the time instant when the final fault $\phi^{[i]}$ occurs (captured by Ψ).

CHAPTER V

Abstraction-based Fault-tolerant Control Synthesis

In this chapter, we present an abstraction-based approach to solve the fault-tolerant control problem, with immediate and delayed fault detection. A class of hierarchical nondeterministic finite (state) transitions systems are used as the abstractions of the continuous-state control system with fault configurations. By construction, the abstraction overapproximates the behavior of the underlying continuous-time system (i.e., the concrete system). We then leverage the control synthesis algorithms developed for finite transition systems as basic building blocks to construct a bottom-up recursive algorithm that solves the fault-tolerant control synthesis problem on the abstraction. The soundness of the proposed algorithm is verified. In particular, the proposed algorithm achieves LTL specification Ψ (Eq. (4.8), Section 4.2) without any further assumptions, and achieves Φ (Eq. (4.7), Section 4.2) assuming the LTL specification for each subsystem (i.e., $\varphi^{[i]}$ in Eq. (4.7)) specifies an absolute decomposable property. The obtained controller is guaranteed to achieve the graceful degradation specification when applied to the concrete system by the behavior overapproximation relation. Since the focus of this chapter is solving the problem on abstractions (i.e., on finite transition systems), we only provide theoretical analysis of the proposed algorithms and to highlight the type of specifications that enable efficient synthesis in

both full information and delayed information settings. We will omit in this chapter the abstraction construction that leads to such behavior overapprximation. Later, we will illustrate detailed abstraction computation process in the next chapter where we aim at applying the developed methodologies to a fuel cell thermal management problem.

Chapter overview. In Section 5.1, we first introduce the LTL game on a finite transition system with fault configurations and define four variants of the fault-tolerant control synthesis problems, depending on the type of graceful degradation considered and whether detection delay exists. In Section 5.2, we present a set of recursive algorithms to solve these problems. The soundness and the completeness of the proposed algorithm are stated and proved under certain conditions on the specification $\varphi^{[i]}$ for each faulty mode. Finally a simple example is used to illustrate the impact of the detection delay.

Related work. The work presented in this chapter is initiated from and significantly extends [133], which only considers simple faulty mode specifications of invariance and persistence with no detection delay. Essentially, the problem considered in [133] can be viewed as a mode target game [8] but with an extra structure that governs the mode changing. The problem solved in this chapter has the same structure but involves a larger fragment of LTL, and is hence not comparable to mode target game in general. However, the solutions of the two problems are similar because in both cases, it is essential to keep the capability to respond to a mode that may dominate later. In that sense, it is also related to [65] where the system may enters different modes with extra preview information of mode changing.

The fault-tolerant control synthesis problem with detection delay can be viewed as a partial information game, whose solution usually involves belief space construction [22, 29, 101], which allows one to estimate the missing information (to detect the fault in our case). The key difference of this work is, by exploring the structure of the fault configurations and by restricting ourselves to certain fragment of LTL, we can avoid constructing the belief space and this significantly reduces the computational burden.

5.1 Finite Transition Systems with Fault Configurations

We first define finite transition systems with fault configurations, the discrete analogue of continuous-state control systems with fault configurations. The former one can be viewed as the abstraction that overapproximates the behavior of the latter one. To this point, we first define regular finite transition systems.

Definition 10. A *finite transition system*, denoted by TS, is a tuple $(Q, A, \rightarrow, AP, L_Q)$, where

- Q is a finite set of states,
- A is a finite set of actions,
- $\rightarrow \subseteq Q \times A \times Q$ is a transition relation,
- AP is a finite set of atomic propositions,
- $L_Q : Q \rightarrow 2^{AP}$ is a labeling function.

Particularly, we assume that a finite transition system TS can start from any state in Q, and the transitions happen only at time instant $t \in \mathbb{N}$.

The LTL game defined on a regular finite transition system defined above proceeds as follows:

1) at time t, the system's state is at $q(t)$,

2) a strategy $K : Q^* \to 2^A$ determines a control action $a(t) \in K\big(q(1)q(2)\cdots q(t)\big)$,

3) the environment picks the next state $q(t+1)$ such that $\big(q(t), a(t), q(t+1)\big) \in \to$,

4) step 1) to step 3) are repeated.

Let $\mathfrak{q} = q(1)q(2)\cdots$ be an infinite sequence generated by system TS under strategy K with the above procedure. The associated ω-word is defined as $\mathfrak{w} = L_Q\big(q(1)\big)L_Q\big(q(2)\big)\cdots$. Clearly, the set of possible ω-words is uniquely determined by the system TS, the strategy K and the initial state $q(1)$. We denote this set by $\mathfrak{W}\big(TS, K, q(1)\big)$. Let φ be an LTL formula, strategy K is called winning at $q(1)$ if any $\mathfrak{w} \models \varphi$ for any $\mathfrak{w} \in \mathfrak{W}\big(TS, K, q(1)\big)$. A set of such initial conditions at which a winning strategy exists is called a winning set. Clearly a maximal winning set exists and it can be determined by solving a Rabin game [11].

In this work, we are interested in finite transition system with fault configurations (faulty systems for short), which is the discrete analogue of the systems defined in Section 4.1. Such a system consists of a collection of different regular finite transition systems (regular systems for short), each governing the system transitions under a specific faulty situation, and these different regular systems may degrade from one to another in the order of their corresponding fault severity. The formal definition is given below.

Definition 11. A *finite transition systems with fault configurations*, denoted by TS^{F}, is a tuple $= (Q, F, A, \to_{TS}, \to_F, AP^{\mathrm{F}}, L_Q^{\mathrm{F}})$ where

- Q, A have the same meanings as the ones in a regular finite transition systems, $F = \{\phi^{[1]}, \ldots, \phi^{[M]}\}$ is the given fault configuration, and there is an atomic proposition $\pi^{[i]} \in AP^{\mathrm{F}}$ associated with each fault $\phi^{[i]} \in F$;
- $\to_{TS} \subseteq Q \times F \times A \times Q$ is a transition relation that describes the system's evolution under some specific fault;

- $\rightarrow_F \subseteq F \times F$ is the transition relation of the faults, and we assume that the transitions of faults always start from healthy configuration $\phi^{[1]}$, and that $(\phi^{[i]}, \phi^{[j]}) \in \rightarrow_F$ iff $\phi^{[j]} \in \{\phi^{[i]}\} \cup \text{succ}(\phi^{[i]})$, a fault transition $(\phi^{[i]}, \phi^{[j]})$ is called *nontrivial* if $\phi^{[i]} \neq \phi^{[j]}$;
- $L_Q^{\text{F}} : Q \times F \rightarrow AP$ is the labeling function, such that $\pi^{[i]} \in L_Q^{\text{F}}(q, \phi)$ iff $\phi = \phi^{[i]}$.

Similarly to regular systems, we define an execution of system TS^{F} to be an infinite sequence of 3-tuples $\big(q(1), \phi(1), a(1)\big)\big(q(2), \phi(2), a(2)\big) \cdots$, where $\big(q(t), \phi(t), a(t), q(t+1)\big) \in \rightarrow_{TS}$, and $\big(\phi(t), \phi(t+1)\big) \in \rightarrow_F$ for all $t \in \mathbb{N}$. The ω-word generated by this execution is $L_Q^{\text{F}}\big(q(1), \phi(1)\big) L\big(q(2), \phi(2)\big), \cdots$.

A few remarks regarding to the definition above are in order. First, It might be helpful to think TS^{F} as a hierarchical transition system with M different regular finite transition systems as subsystems. Each subsystem $TS^{[i]}$ is associated with the fault configuration $\phi^{[i]} \in F$ of the same subscript. Every $TS^{[i]}$ has a distinct transition relations $\rightarrow^{[i]} := \{(q, a, q') \mid (q, \phi^{[i]}, a, q') \in \rightarrow_{TS}\}^{\text{F}}$ and different labeling functions $L_Q^{[i]} := L_Q^{\text{F}}(\cdot, \phi^{[i]})$. The transition of the overall system TS^{F} can be seen as being governed by $\rightarrow^{[i]}$ that corresponds to the current fault status, while transition relation $\rightarrow_F$ describes the degradation of governing subsystem $TS^{[i]}$ in case the fault status changes.

Secondly, by definition, TS^{F} has a common state set Q, atomic proposition set AP^{F} and action set A that is shared by all of its subsystems $TS^{[i]}$. Note that we may have different control authority and atomic proposition of interest under different fault configurations. However, the assumption for common atomic proposition set and common action set can be made without loss of generality. In case we have different propositions $AP^{[i]}$ of interest in different fault configurations, a common atomic proposition set AP can be simply chosen to be $\bigcup_{i=1}^{M} AP^{[i]}$, and the difference

can be handled by defining non-surjective labeling function $L_Q^{[i]}$. Moreover, the lack of control authority under more severe fault configurations can be captured by a transition relation $\rightarrow^{[i]}$ that is not affected by some inactive control action $a \in A$. State dependent control authority can be also easily incorporated.

Finally, note that the fault transition relation $\rightarrow_F$ is beyond our control, hence it introduces additional nondeterminism into the system, and such nondeterminism can be combined with that of a regular system—whose state set is $Q \times F$—to obtain the faulty system TS^{F}. This means a faulty system is nothing but a special type of regular finite transition system. However, as will be presented later, the special structure of fault configurations can be leveraged to develop a recursive synthesis process when the considered specification is in certain from. We hence distinguish it from regular systems.

5.1.1 Fault Detection on Finite Transition Systems with Faulty Modes

In practice, if the faulty mode of the system changes, we may not know this immediately. For systems that evolve on discrete time, it will take at least one time step for the faulty behavior to distinguish from the healthy behavior and therefore the fault is detected with at least one step delay. Moreover, as shown in Chapter III, the fault detection performed on the continuous state system that a faulty transition system abstracts may be delayed by a time period, and this period can be upper bounded by a constant T. Therefore a true faulty mode sequence $\mathbb{f} = \phi(1)\phi(2)\cdots$ may not be the same as the status sequence $\widehat{\mathbb{f}} = \widehat{\phi}(1)\widehat{\phi}(2)\cdots$.

Definition 12. Let $\mathbf{f} = \phi(1)\phi(2)\cdots\phi(t+1) \in F^{t+1}$ and $\widehat{\mathbf{f}} = \widehat{\phi}(1)\widehat{\phi}(2)\cdots\widehat{\phi}(t+1) \in F^{t+1}$, we say $(\widehat{\mathbf{f}}, \mathbf{f})$ is *valid* if

- further faults do not occur before the latest fault is detected: $\widehat{\phi}(t) \neq \phi(t-1) \Rightarrow$

$\phi(t+1) = \phi(t)$;

- the detection is delayed by at least one step: $\phi(t+1) \neq \phi(t) \;\Rightarrow\; \widehat{\phi}(t+1) = \phi(t)$;
- the detection of $\phi^{[i]}$ is delayed by at most $T^{[i]} \geq 1$: $\phi(t) = \cdots = \phi(t - T^{[i]} + 1) = \phi^{[i]} \;\Rightarrow\; \widehat{\phi}(t+1) = \phi^{[i]}$.

Let $\mathbb{f}, \widehat{\mathbb{f}} \in F^\omega$, the pair $(\widehat{\mathbb{f}}, \mathbb{f})$ is *valid* if $(\widehat{\mathbf{f}}_{t+1}, \mathbf{f}_{t+1})$ is valid for all t, where $\mathbf{f}_{t+1}$ (and $\widehat{\mathbf{f}}_{t+1}$ respectively) is the prefix of $\mathbb{f}$ (and $\widehat{\mathbb{f}}$ respectively) until time $t+1$.

Clearly, the notion of validity only depends on the fault configuration F and the set $T := \{T^{[i]}\}_{i:\phi^{[i]} \in F}$ that specifies the bounds of the detection delay. We will denote the set of all valid pairs by $\mathfrak{V}_{\text{delay}}(F, T)$.

Also note that any valid pair $(\widehat{\mathbf{f}}, \mathbf{f})$ is *extendable*, i.e., if $(\widehat{\mathbf{f}}, \mathbf{f})$ is valid, where $\mathbf{f} = \phi(1)\phi(2)\cdots\phi(t+1)$ and $\widehat{\mathbf{f}} = \widehat{\phi}(1)\widehat{\phi}(2)\cdots\widehat{\phi}(t+1)$, then there always exists valid $(\widehat{\mathbf{g}}, \mathbf{g})$ such that $\mathbf{g} = \phi(1)\phi(2)\cdots\phi(t+1)\phi(t+2)$ and $\widehat{\mathbf{g}} = \widehat{\phi}(1)\widehat{\phi}(2)\cdots\widehat{\phi}(t+1)\widehat{\phi}(t+2)$, where $\phi(t+2), \widehat{\phi}(t+2) \in \{\phi(t)\} \cup \operatorname{succ}\big(\phi(t)\big)$.

In this work, we will assume one of the following is true.

Assumption 1. *(Immediate Detection)* Let $\mathbb{f}$ be the sequence of true faulty modes and $\widehat{\mathbb{f}}$ be the fault status sequence returned by the detector, we assume that $\mathbb{f} = \widehat{\mathbb{f}}$ and the only rule they need to follow is $\big(\phi(t), \phi(t+1)\big) \in \rightarrow_F$. We denote the set of the pairs of such identical faulty mode sequences by $\mathfrak{V}_{\text{no-delay}}(F)$, where F is the fault configuration set.

Assumption 2. *(Delayed Detection)* Let $\mathbb{f}$ be the sequence of true faulty modes and $\widehat{\mathbb{f}}$ be the fault status sequence returned by the detector, we assume that $(\widehat{\mathbb{f}}, \mathbb{f})$ is valid, i.e., $(\widehat{\mathbb{f}}, \mathbb{f}) = \mathfrak{V}_{\text{delay}}(F, T)$, where $T = \{T^{[i]}\}_{i \in F}$ specifies the detection delay bounds.

5.1.2 LTL Games on Finite Transition Systems with Faulty Modes

Given LTL formula Ψ and Φ defined in Chapter IV, we define in what follows the LTL game between the controller and the environment w.r.t the given specification. At any time t,

1) the system state is at $q(t)$, the faulty mode is $\phi(t)$,
2) the controller $\mu : (Q \times F)^* \to 2^A$ determines an action
   $a(t) \in \mu\Big(\big(q(1), \widehat{\phi}(1)\big)\big(q(t), \widehat{\phi}(t)\big), \ldots, \big(q(t), \widehat{\phi}(t)\big)\Big)$,
3) the environment picks the next state $q(t+1)$ s.t. $\big(q(t), \phi(t), a(t), q(t+1)\big) \to_{TS}$,
4) at time $t+1$, the environment pick $\phi(t+1)$ and $\widehat{\phi}(t+1)$ s.t. $(\widehat{\mathbf{f}}_{t+1}, \mathbf{f}_{t+1})$ is valid,
5) step 1) to step 4) are repeated.

Let $q(1)$ be the initial state, and $\mathbb{f}$, $\widehat{\mathbb{f}}$ be the (valid) true and estimated faulty mode sequence, the set of words generated by the system TS^{F} is uniquely defined by $\mathbb{f}$, $\widehat{\mathbb{f}}$, control strategy μ and $q(1)$. We denote the set of such a word by $\mathfrak{W}\big(\mathbb{f}, \widehat{\mathbb{f}}, \mu, q(1)\big)$.

Definition 13. *(Winning Strategy/Set)* Strategy μ is winning at $q(1)$ against LTL formula φ if

$$\forall(\mathbb{f}, \widehat{\mathbb{f}}) \in \mathfrak{V}, \mathbb{w} \in \mathfrak{W}\big(\mathbb{f}, \widehat{\mathbb{f}}, \mu, q(1)\big) : \mathbb{w} \models \varphi, \tag{5.1}$$

where $\mathfrak{V}$ can be either $\mathfrak{V}_{\mathrm{no-delay}}(F)$ or $\mathfrak{V}_{\mathrm{delay}}(F, T)$, depending on if immediate detection is assumed or not. A set of initial states $q(1)$ at which μ is winning is called a winning set, and clearly a maximal winning set exists.

Problem 2. *(Fault-tolerant Control Synthesis)* In the rest of the chapter, we consider the problem of finding a winning set (preferably the maximal one) and the associated winning strategy μ. Depending if immediate detection is assume or not, and also if Ψ or Φ is considered, we have the following four sub-problems:

Porblem(Ψ, no-delay),

Porblem(Φ, no-delay),

Porblem(Ψ, delay),

Porblem(Φ, delay).

Remark 3. When there is a detection delay, a notable feature of the above LTL game is that it might be undetermined [22]. That is, there might be certain states at which neither the environment nor the controller wins for sure. To achieve correct-by-construction, we shall avoid such undetermined states, and this is exactly how the winning set is defined (see Definition 13), i.e., we only search for the initial states from where the controller is guaranteed to win.

5.2 Bottom-up Fault-tolerant Control Synthesis

In this section, a set of bottom up recursive algorithms are proposed to solve Problem 2. The soundness and completeness of the proposed algorithms can be proved under certain assumptions on $\varphi^{[i]}$, and are summarized with Table 5.1.

Table 5.1: Summary of soundness and completeness of the algorithms

Specification	Assumption on $\varphi^{[i]}$	No delay	Delay
Ψ	any LTL	Algorithm 1 Sound & Complete	-
	absolute decomposable		Algorithm 3 Sound & Complete
	absolute decomposable + suffix-closed		Algorithm 3 Sound & Complete
Φ	absolute decomposable	Algorithm 2 Sound	-
	absolute decomposable + suffix-closed		Algorithm 3 Sound & Complete

5.2.1 Synthesis against Specification Ψ with Immediate Detection

In this part, we present Algorithm 1 that solves Problem(Ψ, no-delay).

Algorithm 1 $[W^{[i]}, \mathcal{K}^{[i]}] = \mathbf{Win}^{\mathrm{F}}_{\Psi,\text{no-delay}}(\Psi, TS^{\mathrm{F}}, \phi^{[i]})$

Initialize $W^{[i]} = \emptyset$, $\mathcal{K}^{[i]} = \emptyset$
if $\phi^{[i]} \in \max(F)$ **then**
  $[W^{[i]}, K^{[i]}] = \mathbf{Win}(\varphi^{[i]}, TS^{[i]})$
  $\mathcal{K}^{[i]} = \{K^{[i]}\}$
else
  for $\phi^{[j]} \in \mathrm{succ}(\phi^{[i]})$ **do**
    $[W^{[j]}, \mathcal{K}^{[j]}] = \mathbf{Win}^{\mathrm{F}}_{\Psi,\text{no-delay}}(\Psi, TS^{\mathrm{F}}, \phi^{[j]})$
    $\psi^{[i]} = \varphi^{[i]} \wedge (\Box W^{[j]})$
  end for
  $[W^{[i]}, K^{[i]}] = \mathbf{Win}(\psi^{[i]}, TS^{[i]})$
  $\mathcal{K}^{[i]} = \mathcal{K}^{[j]} \cup \{K^{[i]}\}$
end if
return $W^{[i]}$, $\mathcal{K}^{[i]}$

1) Inputs: Algorithm 1 takes a system TS^{F}, an LTL formula Ψ in form of Eq, (4.8), and a fault configuration $\phi^{[i]}$ as inputs. Fault $\phi^{[i]}$ can be seen as the initial configuration. Note that a faulty system always starts from being healthy by definition, here $\phi^{[i]}$ is used to track the recursion.

2) Outputs: Algorithm 1 returns set $W^{[i]}$ as a winning set w.r.t. specification Φ when system TS^{F} starts from fault configuration $\phi^{[i]}$. $\mathcal{K}^{[i]} := \{K^{[j]}\}_{\phi^{[j]} \succeq \phi^{[i]}}$ is a collection of maps $K^{[j]}$, each map $K^{[j]} : Q^* \to 2^A$ is a sub-strategy that achieves specification $\psi^{[j]}$ if the system starts from state $q \in W^{[j]}$ and stays at fault configuration $\phi^{[j]}$ forever. The fault-tolerant strategy, with initial fault $\phi^{[i]}$ and initial state $q(1)$, can be then extracted from $\mathcal{K}^{[i]}$ by appending strategy fragments of $K^{[j]}(q)$'s according to the latest fault status and the recent states after that fault occurring. Formally, this fault-tolerant strategy at time t is defined as:

$$\begin{aligned} &\mu\Big(\big(q(1), \phi(1)\big) \cdots \big(q(t), \phi(t)\big)\Big) \\ =&K^{[n]}\big(q(s)q(s+1)\cdots q(t-1)q(t)\big), \end{aligned} \tag{5.2}$$

where n in "$K^{[n]}$" is the superscript of latest fault $\phi(t) = \phi^{[n]}$, and

$$s = \min_{\substack{0 \le \tau \le t \\ \phi(\tau) = \phi(t)}} \tau. \tag{5.3}$$

Let $\mathbf{q}_t = q(1)q(2)\dots q(t)$ where $q(1) = q_1$ and $\mathbf{f}_t = \phi(1)\phi(2)\dots\phi(t)$, we denote the set of admissible actions at time t suggested by strategy μ by $\mu(\mathbf{q}_t, \mathbf{f}_t)$.

3) Recursion: Algorithm 1 repetitively calls itself until the worst faults are achieved as base cases. In each round of recursion, we need the following oracle $[W^{[i]}, K^{[i]}] = \mathbf{Win}(\psi^{[i]}, TS^{[i]})$, which returns the maximum winning set $W^{[i]}$, and a map $K^{[i]}$ associating a state from $W^{[i]}$ with a winning strategy, so that all executions of $TS^{[i]}$ under such strategy satisfy LTL formula $\psi^{[i]}$. If a worst fault is reached, function $\mathbf{Win}^{\mathrm{F}}$ simply returns the normal winning set and strategies, because the system will not further degrade from there. Otherwise a further degradation is possible. To avoid generating prefixes that violate the specification for the final fault, we strengthen the current specification by $\Box W^{[j]}$ where $W^{[j]}$ is the winning set returned by deeper recursions. Finally oracle **Win** is called to synthesize the winning set w.r.t. specification $\psi^{[i]}$ and this finishes the round of the recursion. Figure 5.1 shows an illustrative example of the recursion.

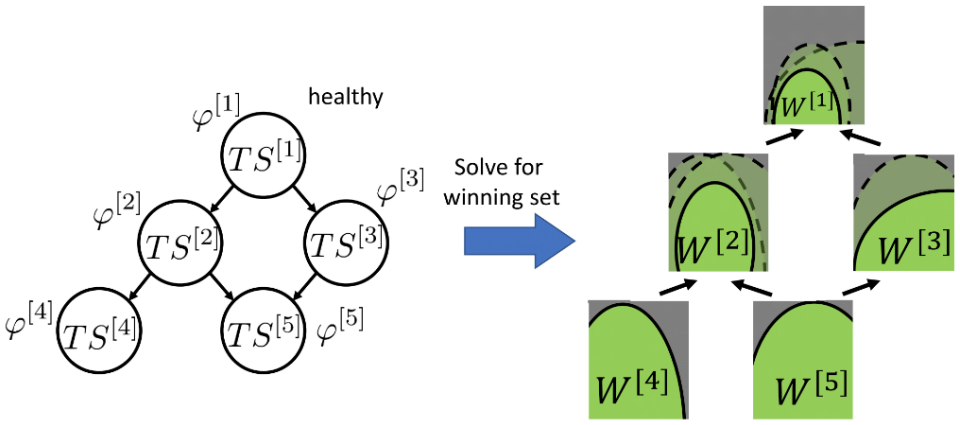

Figure 5.1: Illustration of Algorithm 1.

The correctness and completeness of Algorithm 1 is stated with the following theorem.

Theorem 3. Algorithm 1 solves Problem(Ψ, no-delay) soundly and completely.

Proof. Algorithm 1 is sound and complete essentially because $W^{[i]} \subseteq W^{[j]}$ for all $\phi^{[j]} \in \text{succ}(\phi^{[i]})$ is sufficient and necessary to guarantee correctness. For detailed proof, please see Appendix 1.1 □

5.2.2 Synthesis against Specification Φ with Immediate Detection

In this part, we modify Algorithm 1 to solve Problem(Φ, no-delay). We start with introducing the idea of the modification in an intuitive way, and then explain the algorithms by highlighting its difference from Algorithm 1. Finally, the correctness of the modified algorithm is stated and proved under certain assumptions.

First, we would like to point out the key difference between Φ and Ψ. Suppose $\phi^{[i]}$ is the final fault that occurs at time t, Φ requires the corresponding LTL formula $\varphi^{[i]}$ to be satisfied from the very beginning, while Ψ only requires $\varphi^{[i]}$ to be satisfied from time instant t. To achieve specification, we hence need to be more careful with the system's behavior before $\phi^{[i]}$ occurs. This leads to the following challenge:

(I1) (**bad prefix issue**) the finite word generated by the old strategy may violate the new specification $\varphi^{[j]}$,

The other major challenge in solving Problem(Φ, no-delay) is: the final fault configuration is not known in advance, nor is the time this fault occurs. Therefore, the controller has to assume the current fault configuration $\phi^{[i]}$ is the final one, and give a strategy that achieves the specification associated with the current configuration. However, unless no other faults are strictly more severe than the current one, there is always a chance for the system to further degrade. Therefore, the controller must also maintain the capability to achieve the specifications for possible succeeding faults $\phi^{[j]}$. This leads to another challenge

(I2) (**succeeding strategy issue**) there may not be a new strategy to achieve specification $\varphi^{[j]}$ starting from the current state.

Finally note that the argument also applies to the new fault $\phi^{[j]}$ and its succeeding configurations, if any. This hence suggests the following recursive algorithm similar to Algorithm 1.

Algorithm 2 $[W^{[i]}, \mathcal{K}^{[i]}, \psi^{[i]}] = \mathbf{Win}^{\mathrm{F}}_{\Phi,\text{no-delay}}(\Phi, TS^{\mathrm{F}}, \phi^{[i]})$

1: Initialize $W^{[i]} = \emptyset$, $\mathcal{K}^{[i]} = \emptyset$, $\psi^{[i]} = \varphi^{[i]}$
2: **if** $\phi^{[i]} \in \max(F)$ **then**
3: $[W^{[i]}, K^{[i]}] = \mathbf{Win}(\varphi^{[i]}, TS^{[i]})$
4: $\mathcal{K}^{[i]} = \{K^{[i]}\}$
5: **else**
6: **for** $\phi^{[j]} \in \mathrm{succ}(\phi^{[i]})$ **do**
7: $[W^{[j]}, \mathcal{K}^{[j]}, \psi^{[j]}] = \mathbf{Win}^{F}_{\Phi,\mathrm{no-delay}}(\Phi, TS^{\mathrm{F}}, \phi^{[j]})$
8: $[\psi^{[j]}_{\text{safety}}, \psi^{[j]}_{\text{liveness}}] = \mathbf{Decomp}(\psi^{[j]})$
9: $\psi^{[i]} = \varphi^{[i]} \wedge (\Box W^{[j]}) \wedge \psi^{[j]}_{\text{safety}}$
10: **end for**
11: $[W^{[i]}, K^{[i]}] = \mathbf{Win}(\psi^{[i]}, TS^{[i]})$
12: $\mathcal{K}^{[i]} = \mathcal{K}^{[j]} \cup \{K^{[i]}\}$
13: **end if**
14: **return** $W^{[i]}$, $\mathcal{K}^{[i]}$, $\psi^{[i]}$

One key difference from Algorithm 1 is that Algorithm 2 also returns an LTL formula $\psi[i]$ called the *strengthened formula*, which is obtained by strengthening $\varphi^{[i]}$ by additional safety specifications. $\psi^{[i]}$ can be seen as the specification of an overall system that captures all possible degradations from current fault $\phi^{[i]}$. With this strengthened formula, $\varphi^{[j]}_{\text{safety}}$ propagates in the recursion and is added on top of the requirements for all proceeding modes of $\phi^{[j]}$, i.e., the faulty modes that may degrade to $\phi^{[j]}$ in the future. Such construction will guarantee $\phi^{[j]}$ to be satisfied from the very beginning.

The correctness of Algorithm 2 is stated and proved.

Theorem 4. Assume that each $\varphi^{[i]}$ specifies an absolutely decomposable property, then Algorithm 2 solves Problem(Φ, no-delay) soundly.

Proof. See Appendix 1.2 □

Computational Complexity

Even with immediate fault detection, the proposed algorithms are computationally advantageous as it breaks the synthesis problem in to a collection of smaller synthesis problems using the structure of the fault configuration. We briefly discuss this benefits in what follows.

As mentioned earlier at the end of section 5.1, faulty system TS^{F} can be modeled by a regular transition system with state space $Q \times F$. Problem(Ψ, no-delay) and Problem(Φ, no-delay) hence can be solved theoretically by solving a Rabin game [11], whose complexity is given by

$$\mathcal{O}\left(\left(|A|\,|Q|\,|F|\,2^{(2^{|\Phi|}|\Phi)|}\right)^{2k}\right), \tag{5.4}$$

where $|A|$ is the size of action set, $|Q|$ is size of state space of a regular system for each fault configuration, $|F|$ is number of faults, $|\Phi|$ is the length of LTL formula in Eq. (4.7), and k is the number of accepting pairs in associated Rabin automaton, which is a small number that is usually equal to 1 [27].

The complexity of Algorithm 2, ignoring the complexity for LTL formula decomposition, is given by

$$\mathcal{O}\left(|F|\left(|A|\,|Q|\,2^{(2^{|\varphi|}|\varphi)|}\right)^{2k}\right), \tag{5.5}$$

where $|\varphi| = \max_{i:\phi^{[i]}\in F} |\varphi^{[i]}|$. The complexity of our approach is linear in $|F|$, the number of faults, while the complexity in Eq. (5.4) contains term $\mathcal{O}(|F|\,2^{(2^{|\Phi|}|\Phi)|})^{2k})$ where $|\Phi|$ is linear in $|F|$.

5.2.3 Synthesis against Specification Ψ with Delayed Detection

In this part, we modify Algorithm 1 to solve Problem(Ψ, delay). In what follows, we will first briefly discuss the challenges when the fault detection is delayed. Then the modified algorithm is presented. The correctness of the modification is proved with additional assumptions on $\varphi^{[i]}$ that specifies the desired behavior under faulty mode $\phi^{[i]}$.

Challenges under Detection Delay

We first address the challenges caused by detection delay. Assume that the system degrades from configuration $\phi^{[i]}$ to $\phi^{[j]}$, there will be a time period, called *uninformed execution horizon*, within which the latest degradation is not known to the controller. This time horizon starts from the instant when transition $(\phi^{[i]}, \phi^{[j]})$ happens, and lasts for at most time $T^{[j]}$ by our detectability assumption. Within the uninformed execution horizon, the controller will assume that the evolution is governed by original system $TS^{[i]}$ and apply the old strategy, while the system dynamics evolves according to the transitions of the new system $TS^{[j]}$. As a result, two things may go wrong during the uninformed execution horizon, i.e.,

(I3) (**bad prefix issue**) the wrongly-controlled partial execution may violate specification $\varphi^{[j]}$ for some possible succeeding fault configuration $\phi^{[j]} \succ \phi^{[i]}$;

(I4) (**succeeding strategy issue**) the execution may be led to parts of the state space where no strategies are available to achieve specification $\varphi^{[j]}$ for some succeeding faults.

Note that the bad prefix issue (I3) is related to issue (I1), but different. Essentially, issue (I1) is a result of specification Φ requiring $\varphi^{[i]}$ to be satisfied starting from the very beginning instead of the time instant of fault occurrence, while (I3) is simply resulted from possible improper operation between fault occurrence and its delayed

detection. Issue (I3) cannot be solved by simply applying Algorithm 2. In Algorithm 2 (line 9), the specifications under succeeding faults are taken into consideration when synthesizing controller for current fault $\phi^{[i]}$. However, the synthesis is done for system $TS^{[i]}$ rather than $TS^{[j]}$. Within the uninformed execution horizon, the controller applies strategy designed for system $TS^{[i]}$ while the system evolves as $TS^{[j]}$. Therefore the outcome prefix may still violate strengthened formula $\psi^{[i]}$, hence violate $\psi^{[j]}_{\text{safety}}$ and the specifications for the succeeding faults. For a similar reason, issue (I4) cannot be solved by simply enforcing $\Box W^{[j]}$. Because $\Box W^{[j]}$ may still be violated if the controller applies strategy designed for $TS^{[i]}$ while the system evolves as $TS^{[j]}$.

Synthesis with Invariance Restrictions

We now present the modification of Algorithm 1 to solve Problem(Ψ, delay). The key idea is to synthesize controllers for each faulty mode with extra invariance constraints.

Algorithm 3 $[W^{[i]}, \mathcal{K}^{[i]}] = \mathbf{Win}^{\mathrm{F}}_{\Psi,\text{delay}}(\Psi, TS^{\mathrm{F}}, \phi^{[i]})$

1: Initialize $W^{[i]} = \emptyset$, $\mathcal{K}^{[i]} = \emptyset$
2: **if** $\phi^{[i]} \in \max(F)$ **then**
3: $\quad [W^{[i]}, K^{[i]}] = \mathbf{Win}(\varphi^{[i]}, TS^{[i]})$
4: $\quad \mathcal{K}^{[i]} = \{K^{[i]}\}$
5: **else**
6: $\quad$ define $I : Q \to 2^U$ to be s.t. $I(q) = U,\ \forall q \in Q$
7: $\quad$ **for** $\phi^{[j]} \in \text{succ}(\phi^{[i]})$ **do**
8: $\quad\quad [W^{[j]}, \mathcal{K}^{[j]}] = \mathbf{Win}^{\mathrm{F}}_{\Psi,\text{delay}}(\Psi, TS^{\mathrm{F}}, \phi^{[j]})$
9: $\quad\quad [C^{[j]}, I^{[j]}] = \mathbf{CInv}(W^{[j]}, TS^{[j]})$
10: $\quad\quad \psi^{[i]} = \varphi^{[i]} \wedge (\Box C^{[j]})$
11: $\quad\quad I(q) = I(q) \cap I^{[j]}(q),\ \forall q \in Q$
12: $\quad$ **end for**
13: $\quad TS^{[i]} \odot I := TS^{[i]}$ with $u \notin I(q)$ disabled at state $q \in Q$
14: $\quad [W^{[i]}, K^{[i]}] = \mathbf{Win}(\psi^{[i]}, TS^{[i]} \odot I)$
15: $\quad \mathcal{K}^{[i]} = \mathcal{K}^{[j]} \cup \{K^{[i]}\}$
16: **end if**
17: **return** $W^{[i]}$, $\mathcal{K}^{[i]}$

The key modification on top of Algorithm 1 is that, in line 9, we search for the maximum controlled invariant set $C^{[j]} \subseteq W^{[j]}$, under the dynamics of $TS^{[j]}$. A map $I^{[j]} : C^{[j]} \to 2^A$ is also found, so that $C^{[j]}$ is invariant as long as we keep applying

any action from $I^{[j]}(q)$ at any state $q \in C^{[j]}$. Such set $C^{[j]}$ and map $I^{[j]}$ can be found by fixed-point type algorithms given in [82]. In particular, $I^{[j]}$ can be made maximally permissive [20] in the sense that ${I^{[j]}}'(q) \subseteq I^{[j]}(q)$ for any mapping ${I^{[j]}}'$ that also guarantees invariance of $C^{[j]}$ under $TS^{[j]}$ dynamics. Then oracle **Win** is called as before, except that the synthesis is restricted within state set $C^{[j]}$, while the actions available at each state $q \in C^{[j]}$ are restricted within $I^{[j]}(q)$. When there are multiple possible faults $\phi^{[j]}$ succeeding current fault $\phi^{[i]}$, an intersection set C can be computed as $\bigcap_{j:\phi^{[j]}\in \mathrm{succ}(\phi^{[i]})} C^{[j]}$, and a map I can be defined to be such that $I(q) = \bigcap_{j:\phi^{[j]}\in \mathrm{succ}(\phi^{[i]})} I^{[j]}(q)$. Then winning set $W^{[i]}$ is then synthesized by **Win** restricted to state set C and control actions in $I(q)$. By the above modification, the states will always stay within $C^{[j]} \subseteq W^{[j]}$ even after the system degrades from fault configuration $\phi^{[i]}$ to $\phi^{[j]}$. This hence guarantees a succeeding strategy after an uninformed execution horizon of any length.

Note that, due to the detection delay, the fault tolerant strategy is defined slightly differently from Eq. (5.2). Let $\widehat{\phi}(t)$ be the estimated faulty mode outputted by the detector at time t, define the fault-tolerant strategy at state q_1 as

$$\begin{aligned} &\widehat{\mu}\Big(\big(q(1),\widehat{\phi}(1)\big)\cdots\big(q(t),\widehat{\phi}(t)\big)\Big) \\ =&K^{[n]}\big(q(s)q(s+1)\cdots q(t-1)q(t)\big), \end{aligned} \tag{5.6}$$

where n in "$K^{[n]}$" is the superscript of latest fault $\widehat{\phi}(t) = \phi^{[n]}$, and

$$s = \min_{\substack{0 \le \tau \le t \\ \widehat{\phi}(\tau) = \widehat{\phi}(t)}} \tau. \tag{5.7}$$

Here, we denote the above strategy by $\widehat{\mu}$ to conceptually distinguish it from the strategy associated with the maximal winning set, because it is not yet clear if the two are equal at this point. Similar as before, let $\widehat{\mathbf{f}}_t = \widehat{\phi}(1)\widehat{\phi}(2)\ldots\widehat{\phi}(t)$, we use $\widehat{\mu}(\mathbf{q}_t, \widehat{\mathbf{f}}_t)$ to denote the action set suggested by $\widehat{\mu}$.

We summarize the properties of the above modified algorithm by the following theorem.

Theorem 5. Under Assumption 2, Algorithm 3 is sound and complete for Problem(Ψ, delay) if $\varphi^{[i]}$ is both absolutely decomposable and suffix-closed for all i.

Proof. See Appendix 1.3 □

The assumption that $\varphi^{[i]}$ is both absolutely decomposable and suffix-closed may look strong at the first sight, but it indeed covers an important class of LTL fragments that are widely considered in the literature of reactive synthesis, i.e. invariance and GR(1) formulas that can be solved relatively efficiently.

Corollary 2. Under Assumption 2, Algorithm 3 is sound and complete for Problem(Ψ, delay) if $\varphi^{[i]} = \varphi^{[i]}_{\text{safety}} \wedge \varphi^{[i]}_{\text{liveness}}$ where $\varphi^{[i]}_{\text{safety}}$ specifies an invariance property and $\varphi^{[i]}_{\text{liveness}}$ is a GR(1) formula.

Proof. By Proposition 10 in Chapter II, invariance property is the only property that is both absolute safe and suffix-closed. By Proposition 9 and Proposition 4 in Chapter II, a GR(1) formula specifies a property that is both absolute liveness and suffix-closed. Applying Theorem 5 immediately proves Corollary 2. □

Discussion

The key insight from the proof of Theorem 5, especially the proof of the completeness part, is that tolerating one-step detection delay is equivalent to tolerating any finite detection delay under the assumption on $\varphi^{[i]}$. First, with the soundness proved, we have $W^{[i]} \subseteq W(\Psi, TS^{\text{F}}, T)$ where $W(\Psi, TS^{\text{F}}, T)$ is the maximal winning set. With the completeness proved with Lemma 5 in the Appendix, we have $W^{[i]} \supseteq W(\Psi, TS^{\text{F}}, T_1) \supseteq W(\Psi, TS^{\text{F}}, T)$ where $W(\Psi, TS^{\text{F}}, T_1)$ is the winning set assuming one-step detection delay. Hence we have $W^{[i]} = W(\Psi, TS^{\text{F}}, T_1) = W(\Psi, TS^{\text{F}}, T)$.

This means that, to solve Problem(Ψ, delay), it is necessary and sufficient to tolerant one-step delay. This is resulted from the presumed suffix-closedness of $\varphi^{[i]}$, which leads to Lemma 4 in the Appendix. By Lemma 4 the winning set $W^{[j]}$ assuring a suffix-closed property must be controlled invariant, and hence the control actions that keep the states in $W^{[j]}$ for one step (i.e., tolerating one-step delay) must lead to invariance (i.e., tolerating any finite delay).

The equivalence between tolerating any finite delay and tolerating one-step delay in this settings is the key to avoid belief state space construction. This is because the belief of the faulty mode $\phi(t)$ can only be established after at least one step of evolution so that the fault makes an impact, while at that point the true $\phi(t)$ is known to the controller by one-step delay assumption anyways.

This impact of the feature mentioned above can be also interpreted from the point of view of separating control design and detection. The setting considered here is slightly different from the general partial information game because the fault detection in this work is assumed to be done for the continuous state system directly as described in Chapter III, whereas only the worst case detection delay is incorporated in the control synthesis on the discrete abstraction. Since the detection for the continuous system is less conservative but the detection on the discrete abstraction is not considered, our approach is in general not comparable to those approaches involving belief space construction. However, if we restrict ourselves to a certain fragment of LTL, i.e., if $\varphi^{[i]}$ is both absolutely decomposable and suffix-closed, we do not gain anything from the detection on the discrete system. Hence the fault detection and the fault-tolerant control synthesis that respects the detection delay can be done separately for such LTL fragment.

5.2.4 Synthesis against Specification Φ with Delayed Detection

The key result in this part is that, under the assumption that LTL specification $\varphi^{[i]}$ for each faulty mode is both suffix-closed and absolutely decomposable, solving Problem(Φ, delay) and Problem(Ψ, delay) are equivalent in the following sense.

Theorem 6. Assume $\varphi^{[i]}$ in Φ is both suffix-closed and absolutely decomposable for all i, then Algorithm 3 solves Problem(Φ, delay) soundly and completely.

Proof. We first prove the soundness. Let $W^{[i]}$ be returned by Algorithm 3, we know any generated word $\mathbb{w} = L_Q^{\mathrm{F}}\big(q(1), \phi(1)\big)L_Q^{\mathrm{F}}\big(q(2)\,\phi(2)\big)\ldots \models \Psi$ by Theorem 5, where $q(1)q(2)\ldots$ is the state sequence and $\phi(1)\phi(2)\cdots$ is the faulty mode sequence. Suppose that the final faulty mode is $\phi^{[i]}$ and let t be the first time instant such that $\phi(t) = \phi^{[i]}$. We have:

(i) $L_Q^{[i]}\big(q(t)\big)L_Q^{[i]}\big(q(t+1)\big)\ldots \models \varphi_{\text{liveness}}$, and this implies $\mathbb{w} \models \varphi_{\text{liveness}}$ as $\varphi_{\text{liveness}}$ is absolute live by assumption.

(ii) $L_Q^{[i]}\big(q(t)\big)L_Q^{[i]}\big(q(t+1)\big)\ldots \models \varphi_{\text{safety}}$, and this implies $q(s) \in W^{[i]}$ for all $s \geq t$ because is φ_{safety} specifies an invariance by Proposition 11. Also note that $q(s) \in W^{[\ell]}$ for any $s < t$ where $\ell : \phi^{[\ell]} = \phi(s)$, but $W^{[\ell]} \subseteq W^{[j]}$ by construction. Hence $q(s) \in W^{[i]}$ for all $s < t$ as well and this implies $\mathbb{w} \models \varphi_{\text{safety}}$

Therefore $\mathbb{w} \models \varphi^{[i]}$ and Φ is achieved as $\phi^{[i]}$ is the final faulty mode.

To show the completeness, it is enough to prove that

$$\mathbb{w} \models \Phi \quad \Rightarrow \quad \mathbb{w} \models \Psi, \tag{5.8}$$

and this immediately gives $W(\Phi, TS^{\mathrm{F}}) \subseteq W(\Psi, TS^{\mathrm{F}})$, which verifies the completeness. To see Eq. (5.8), denote the final fault by $\phi^{[i]}$ and let t be the first time instant s.t. $\phi(t) = \phi^{[i]}$. $\mathbb{w} \models \Phi$ implies that $\mathbb{w} \models \varphi^{[i]}$ as $\phi^{[i]}$ is the final mode, and this

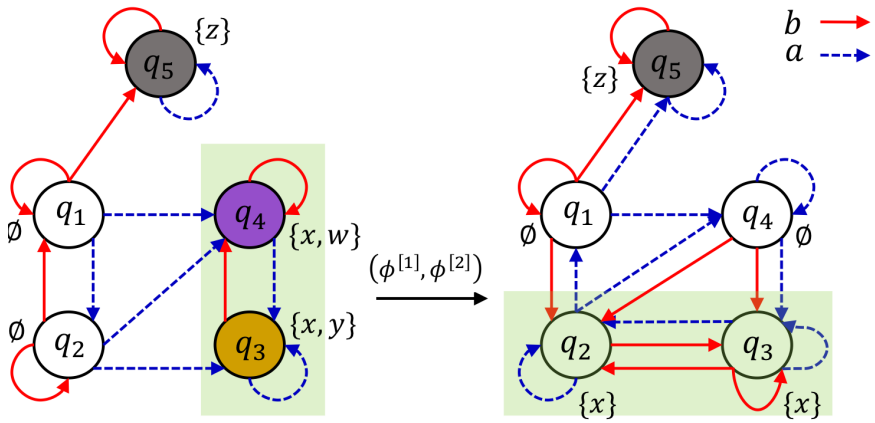

Figure 5.2: Faulty system TS^{F}. Left: regular system $TS^{[1]}$ associated with fault $\phi^{[1]}$, right: $TS^{[2]}$ associated with fault $\phi^{[2]}$. Different colors marked different propositions predicate: w (purple), x (green box), y (orange), z (grey).

further gives $\mathrm{w}_t = w(t)w(t+1)\cdots \models \varphi^{[i]}$ because $\varphi^{[i]}$ is suffix-closed. Hence we have $\mathrm{w} \models \neg\pi^{[i]}\ \mathcal{U}\ (\Box\pi^{[i]} \wedge \varphi^{[i]})$ and hence Ψ. □

An the end of this chapter, a toy example is presented to illustrates the proposed solution approach.

Example 2. Consider a faulty system $TS^{\mathrm{F}} = (Q, F, A, \rightarrow_{TS}, \rightarrow_F, AP, L)$, where $Q = \{q_1, q_2, q_3, q_4, q_5\}$, $F = \{\phi^{[1]}, \phi^{[2]}\}$, $A = \{a, b\}$, $AP = \{w, x, y, z\}$. The only fault transition in $\rightarrow_F$ is $(\phi^{[1]}, \phi^{[2]})$. The system transitions $\rightarrow_{TS}$ and the labeling function are defined in Fig. 5.2. The graceful degradation of system TS^{F} is specified by an LTL formula in form of Eq. 4.7, where

$$\varphi^{[1]} = (\Box\neg z) \wedge (\Diamond\Box x) \wedge (\Box\Diamond w) \wedge (\Box\Diamond y), \tag{5.9}$$

$$\varphi^{[2]} = (\Box\neg z) \wedge (\Diamond\Box x) \tag{5.10}$$

It an be verified that $\varphi^{[1]}, \varphi^{[2]}$ are both absolutely decomposable and suffix-closed. In this case, the graceful degradation Ψ and Φ are equivalent. In what follows we compute the fault tolerant winning set $W(\Psi, TS^{\mathrm{F}}) = W(\Phi, TS^{\mathrm{F}})$ for both cases with and without detection delay, to demonstrate the difference introduced by the delay.

Winning Set without Detection Delay

To find the fault tolerant winning set, Algorithm 2 starts from fault $\phi^{[2]}$ as base case. Winning set W_2 can be found as $\{q_2, q_3, q_4\}$. We then go to fault $\phi^{[1]}$, and strengthen $\varphi^{[i]}$ as $\psi^{[1]} = \varphi^{[1]} \wedge (\Box W^{[2]}) \wedge (\Box \neg x)$, where the last term $(\Box \neg x)$ is the safety part of $\varphi^{[2]}$. Finally we compute $W^{[1]} = \mathbf{Win}(\psi, TS^{[1]}) = \{q_2, q_3, q_4\}$, and we claim that $W^{[1]}$ is a fault tolerant winning set W.

Note that $q_1 \notin W$, even though a strategy can achieve $\varphi^{[1]}$ starting from q_1 on system $TS^{[1]}$. This is because the fault may occur when we are at q_1. In that case the system degrades to $TS^{[2]}$ and no succeeding strategy exists to avoid state q_5, and hence to achieve $\varphi^{[2]}$.

Winning Set with Detection Delay

Assume a finite delay is required to detect the fault, we compute the fault tolerant winning set and show it is different from the above result. Then, similar as before, Algorithm 1 starts from fault $\phi^{[1]}$ and computes winning set $W^{[1]} = \{q_2, q_3, q_4\}$. We then compute $C^{[j]}$ as the largest controlled invariant set in $W^{[2]}$. In this example, $C^{[2]} = W^{[2]}$. Map I_2 is also found such that $I_2(q)$ contains the actions at state q that make $C^{[2]}$ invariant. In this example, $I^{[2]}(q_2) = \{b\}$, $I^{[2]}(q_3) = I^{[2]}(q_4) = \{a, b\}$. Finally the recursion goes back to fault $\phi^{[1]}$, and the fault tolerant winning set is synthesized, with the states restricted to set $C^{[2]}$, and with the actions at state q restricted to $I^{[2]}(q)$. In the example $W^{[1]} = \{q_3, q_4\}$, and we claim the fault tolerant winning set $W = W^{[1]}$.

Unlike the fault tolerant winning set without detection delays, state q_2 is not inside the winning set W. This is because action $a \notin I_2(q_2)$, and is hence forbidden at state q_2 in the synthesis. To see why this it is necessary to exclude state q_2 from W, we consider the following scenario. If fault occurs at state q_2, the controller will

not be informed at once. Instead, the controller assumes that the system evolves as $TS^{[1]}$ and tries to achieve φ_1. Note that to achieve φ_1 on $TS^{[1]}$, the controller has to take action a whenever the state is at q_2. As a result the actual system evolves as $TS^{[2]}$ and may bring the state to s_1, from where the real specification $\varphi^{[2]}$ can not be achieved.

CHAPTER VI

Abstration-based Synthesis of Fuel Cell Thermal Management

In order to apply the fault-tolerant control synthesis algorithms developed in Chapter V to a continuous state system, we still need to find a winning set of the continuous system. This can be done by abstraction-based synthesis. Although the fixed point algorithms developed for finite transition system are quite mature, the computation of abstraction is not a trivial task. The main challenge is to reduce the level of conservatism and to improve efficiency.

In this chapter we show that such abstraction-based synthesis can be done for a fuel cell thermal management system with complicated nonlinear dynamics and complicated specifications. We begin by developing a control-oriented model for the thermal management system of a fuel cell stack. Then, we list the requirements associated with thermal management and formalize them using linear temporal logic. The model and the requirements are then used for controller synthesis with an abstraction-based technique. To make the abstraction-based synthesis algorithm computationally efficient, mixed monotonicity of the fuel cell system dynamics are identified and leveraged. Finally, the closed-loop system behavior with the synthesized controller is demonstrated via simulations.

Chapter overview. This chapter is organized as follows. Section 6.1 introduces the fuel cell thermal management problem. Section 6.2 defines the model and Section 6.3 lists the considered specification in LTL. Seciton 6.4 then describes the abstraction-based synthesis of the system, and Section 6.5 presents the results and some discussion.

Related work. The main technical challenges solved in this Chapter are related to improving the abstraction computation efficiency by leveraging system (mixed) monotonicity [69, 26], and reducing conservatism of a abstraction [63, 103, 82, 85, 41, 51, 26, 136, 89, 109]. In particular, it closely follows the line of research studying so called augmented finite transition systems [89, 109, 82, 85], which are equipped with a progress group mapping that encodes extra transience property of the underlying continuous dynamics.

6.1 Fuel Cell Thermal Management Problem

Fuel cells are electrochemical devices that convert chemical energy of gaseous fuel (i.e., hydrogen) into electricity [94]. In a fuel cell stack, the electrochemical reaction of oxygen and hydrogen generates electrical power, while heat and water are produced as by-products. In this work, we focus on developing the thermal management portion of the controller, which guarantees that the fuel cell operates in a proper temperature range (340K to 350K), for safety and efficiency considerations [57].

A simplified schematic of the fuel cell thermal management system is shown in Fig. 6.1. The two main factors that affect the heat supplied or removed from the fuel cell stack, and hence the stack temperature, are the stack coolant inlet temperature

and the coolant flow-rate. The coolant flow rate is controlled by an electric pump, while the coolant inlet temperature is regulated by appropriately flowing the coolant through a radiator or a heater, where the flow path is selected by a 2-position 3-way valve, thus making the system dynamics hybrid in nature.

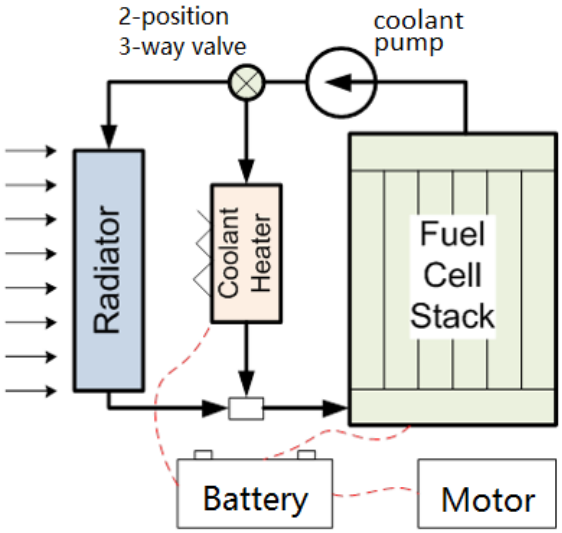

Figure 6.1: Layout of the fuel cell thermal management system. The arrows in the circuit represent the direction of coolant flows.

The electrical power requirements have a direct influence on thermal management, some aspects of which are studied in [50], [79]. First, the power requirements by the motor and the fuel cell temperature constraint need to be met simultaneously through system hybridization, that is, through appropriate power management between the fuel cell stack and an energy storage device. Since the heat generated in the stack increases with increasing fuel cell output power, the stack temperature may exceed the desired range while the fuel cell tries to match its output power with the instantaneous power request from the motor. Hence, a battery is introduced to "moderate" the power generated by the fuel cell. In addition, power requirements from the heater when used for warm-up under cold conditions also affect power management. In this paper, we provide some of the key requirements for fuel cell

thermal management in the presence of battery state of charge energy constraints. These requirements are evaluated in the situation where the ambient temperature is near 283K, where the fuel cell stack loses significant heat to the ambient due to the large temperature gradient.

In works such as [50], [79], requirements due to both power management and thermal management are considered in the controller design. The design approaches in these works are based on optimal control, where the requirements are combined into one objective function and the controller is developed by solving an optimal control problem with the combined objective function. The correctness of the designed controllers need to be verified by running a large number of tests.

6.2 Fuel Cell Model

A block diagram of the fuel cell thermal management system is shown in Fig. 6.2. The solid lines (red) are temperature signals, the dotted lines (purple) are power signals, the dashed lines (blue) are battery SOC signals, and the thinner solid lines (black) are control/reference input signals. The physical meanings of the variables in Fig. 6.2 can be found in the Appendix. Other operating conditions (such as hydrogen and oxygen partial pressure, ambient temperature, vehicle speed) that affect system dynamics are not included in the block diagram for simplicity.

In what follows, we give the formulas describing each block in Fig. 6.2.

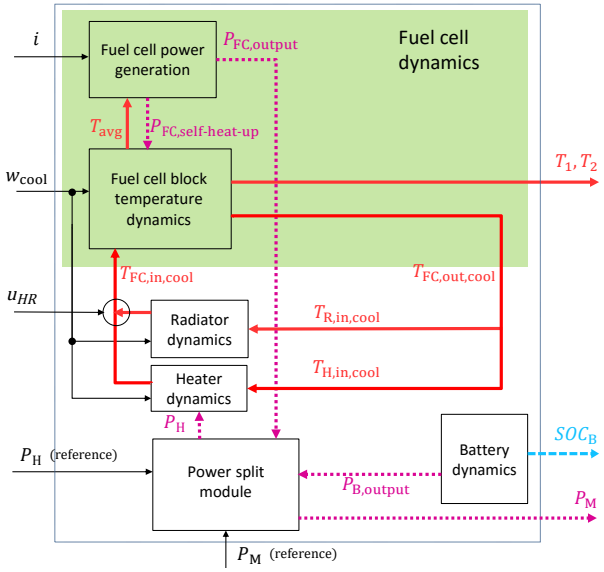

Figure 6.2: Block diagram of the fuel cell thermal management system.

6.2.1 Fuel Cell Power Generation

The fuel cell stack output power and generated heat are computed using the formulas developed in [93],

$$P_{\text{FC,output}} = iA_{\text{G}}E_{\text{FC,stack}}, \tag{6.1}$$

$$P_{\text{FC,self-heat-up}} = iA_{\text{G}}\frac{\Delta h_{\text{rxn}}}{2F}n_{\text{FC,cell}} - P_{\text{FC,output}}, \tag{6.2}$$

and

$$\begin{aligned} E_{\text{FC,stack}} = n_{\text{FC,cell}} \Bigg(& \frac{\Delta h_{\text{rxn}}}{2F} - T_{\text{avg}}\frac{\Delta s_{\text{rxn}}}{2F} \\ & + \frac{RT_{\text{avg}}}{2F}\ln\left(\frac{p_{\text{H}_2}}{P_{\text{ref}}}\left(\frac{p_{\text{O}_2}}{P_{\text{ref}}}\right)^{\frac{1}{2}}\right) - \frac{RT_{\text{avg}}}{\alpha F}\ln\left(\frac{i+i_{\text{x}}}{i_0}\right) \\ & - iR_{\Omega} - a_{\text{MT}}\left(\frac{i}{i_{\text{MT}}}\right)^{b_{\text{MT}}}\Bigg), \end{aligned} \tag{6.3}$$

where $\frac{\Delta h_{\text{rxn}}}{2F}$ and $T_{\text{avg}}\frac{\Delta s_{\text{rxn}}}{2F}$ correspond to the effect of enthalpies and entropies, iR_{Ω} describes Ohmic loss due to cell resistivity, and $a_{\text{MT}}(\frac{i}{i_{\text{MT}}})^{b_{\text{MT}}}$ describes potential loss caused by mass transport limitations. The variables h_{rxn}, s_{rxn}, i_0, R_{Ω} depend on fuel cell average temperature T_{avg} and operating conditions [93].

6.2.2 Fuel Cell Temperature Dynamics

The fuel cell stack is divided into two control volumes to capture its temperature gradient. One control volume is at the coolant inlet side and the other is at the coolant outlet side. The fuel cell temperature dynamics is described in terms of the temperature of the two control volumes, i.e., T_1, T_2. The temperature dynamics are governed by the following differential equation [93]:

$$\begin{aligned}\frac{dT_1}{dt} = \frac{1}{c_{\mathrm{FC}}\rho_{\mathrm{FC}}}\Bigg(&\frac{c_{\mathrm{cool}}w_{\mathrm{cool}}(T_{\mathrm{FC,in,cool}} - T_1)}{n_{\mathrm{FC,cell}}A_{\mathrm{FC}}\delta_{\mathrm{FC}}/2} \\ &+ \frac{\kappa_{\mathrm{T}}(T_2 - T_1)}{(\delta_{\mathrm{FC}}/2)^2} + k_{\mathrm{amb}\to\mathrm{FC}}(T_{\mathrm{amb}} - T_1) \\ &+ \frac{P_{\mathrm{FC,self\text{-}heat\text{-}up}}}{V_{\mathrm{FC}}n_{\mathrm{FC,cell}}} - r_{\mathrm{v}}\Delta h_{\mathrm{v}}\Bigg),\end{aligned} \tag{6.4}$$

$$\begin{aligned}\frac{dT_2}{dt} = \frac{1}{c_{\mathrm{FC}}\rho_{\mathrm{FC}}}\Bigg(&\frac{c_{\mathrm{cool}}w_{\mathrm{cool}}(T_1 - T_2)}{n_{\mathrm{FC,cell}}A_{\mathrm{FC}}\delta_{\mathrm{FC}}/2} \\ &+ \frac{\kappa_{\mathrm{T}}(T_1 - T_2)}{(\delta_{\mathrm{FC}}/2)^2} + k_{\mathrm{amb}\to\mathrm{FC}}(T_{\mathrm{amb}} - T_2) \\ &+ \frac{P_{\mathrm{FC,self\text{-}heat\text{-}up}}}{V_{\mathrm{FC}}n_{\mathrm{FC,cell}}} - r_{\mathrm{v}}\Delta h_{\mathrm{v}}\Bigg),\end{aligned} \tag{6.5}$$

where the inlet coolant temperature $T_{\mathrm{FC,in,cool}}$ in Eq. (6.4) is defined as

$$T_{\mathrm{FC,in,cool}} = u_{\mathrm{HR}}T_{\mathrm{H}} + (1 - u_{\mathrm{HR}})T_{\mathrm{R}}, \tag{6.6}$$

where u_{HR} is the binary variable controlling the 2-position 3-way valve. The average fuel cell temperature used in Eq. (6.3) is defined as $T_{\mathrm{avg}} = (T_1 + T_2)/2$, while $T_{\mathrm{FC,out,cool}}$, the outlet coolant temperature from fuel cell stack, is assumed to be equal to T_2.

6.2.3 Radiator and Heater Temperature Dynamics

The radiator and heater dynamics are given by

$$\frac{dT_{\mathrm{R}}}{dt} = \frac{1}{C_{\mathrm{R}}}\big((1-u_{\mathrm{HR}})c_{\mathrm{cool}}w_{\mathrm{cool}}(T_{\mathrm{FC,out,cool}} - T_{\mathrm{R}})$$

$$+ c_{\mathrm{air}}\varepsilon(v)v(T_{\mathrm{amb}} - T_{\mathrm{R}})\big), \tag{6.7}$$

$$\frac{dT_{\mathrm{H}}}{dt} = \frac{1}{C_{\mathrm{H}}}\big(u_{\mathrm{HR}}c_{\mathrm{cool}}w_{\mathrm{cool}}(T_{\mathrm{FC,out,cool}} - T_{\mathrm{H}}) + P_{\mathrm{H}}\big). \tag{6.8}$$

Note that when binary control $u_{\mathrm{HR}} = 1$ (or 0), the coolant is fed to the heater (or the radiator). The term $\varepsilon(v)$ in radiator dynamics is the vehicle-speed-dependent effectiveness of the radiator, which is modeled as an affine function of vehicle speed v. The outlet coolant temperature from the radiator (the heater, respectively) is assumed to be T_{R} (T_{H}, respectively).

6.2.4 Battery SOC Dynamics

The battery SOC dynamics is adopted from that given in [78],

$$\frac{dSOC_{\mathrm{B}}}{dt} = -n_{\mathrm{s}}n_{\mathrm{p}}E_{\mathrm{B,cell}}\frac{E_{\mathrm{B,cell}} - \sqrt{E_{\mathrm{B,cell}}^2 - \frac{4P_{\mathrm{B,output}}r_{\mathrm{B,cell}}}{n_{\mathrm{s}}n_{\mathrm{p}}}}}{2r_{\mathrm{B,cell}}G_{\mathrm{B,stack,total}}}. \tag{6.9}$$

Note that in Eq. (6.9), $P_{\mathrm{B,output}}$ can be negative, meaning charging the battery.

6.2.5 Power Split Module

The power split module combines the output power from the fuel cell and the battery, and passes part of the combined power to the heater, and the remaining portion to the motor. To deliver the required power to the motor, we assume the battery always provides the right amount of power to compensate for what is generated by the fuel cell, that is,

$$P_{\mathrm{B,output}} = P_{\mathrm{M}} + P_{\mathrm{H}} - P_{\mathrm{FC,output}}. \tag{6.10}$$

6.3 Specifications

In this section, we give the specifications (or requirements) of fuel cell thermal management. The listed specifications are classified into the following three types:

(i) "reach-stay" type specifications require that the system variables (e.g., state or control input) reach a target region in finite time and stay in that region once they arrive;

$\varphi_{\text{reach-stay}} = \Diamond\Box\pi_{\text{target}}$, where π_{target} is a proposition saying that the variable belongs to a designated target set. The formula $\Diamond\Box\pi_{\text{target}}$ says that a time instant exists starting from which π_{target} is always true.

(ii) "avoid" type specifications require that the variables avoid some undesired regions forever (or equivalently, the variables always stay in the complement of the undesired region);

$\varphi_{\text{avoid}} = \Box\pi_{\text{safe}}$, where proposition π_{safe} says that the variable is not in the undesired set, or the variable is in the desired set.

(iii) "recurrence" type specifications require that the variables visit a region repetitively;

$\varphi_{\text{recurrence}} = \Box\Diamond\pi_{\text{recurrence}}$, where $\pi_{\text{recurrence}}$ says the variable belongs to a recurrence target set. The formula $\Box\Diamond\pi_{\text{recurrence}}$ is interpreted as: for all time instants, a time exists in the future at which $\pi_{\text{recurrence}}$ holds, therefore guaranteeing repetition.

For the remainder of this section, we give the fuel cell thermal management specifications in plain English and express them in LTL. The resulting LTL formulas are listed in TABLE 6.1.

6.3.1 Limitations of Fuel Cell Output Power

Some requirements regarding fuel cell output power are imposed in this part.

Spec1: (avoid) The fuel cell output power $P_{\text{FC,output}}$ should not drop below zero.

Fig. 6.3 gives the fuel cell output power predicted by the model in Section 6.2.1. As show in Fig. 6.3, the model-predicted fuel cell output power becomes negative when the current density is too high, which makes the model invalid at that value of current density. This requirement is essential to avoid operating in the region where the model is invalid.

Spec2: (avoid) Let i^* be the fuel cell current density that gives the maximum fuel cell output power, graphically illustrated by Fig. 6.3. The current density should not exceed the one that gives the maximum output power $P_{\text{FC,max}}$ because operating above $P_{\text{FC,max}}$ is inefficient and could lead to irreversible degradation [56]. Note that i^* is a function of state and operating conditions.

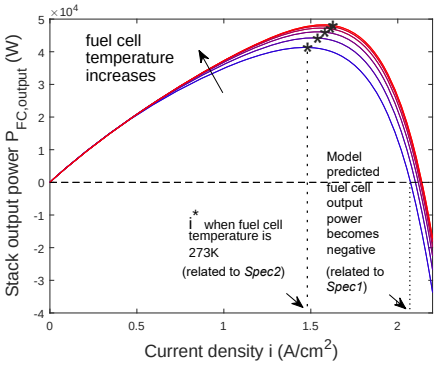

Figure 6.3: Fuel cell power versus current density, fuel cell average temperature varies in [273, 360]K, membrane water content $\lambda = 6$.

Based on Fig. 6.3, it is obvious that *Spec2* actually implies *Spec1*. In this work, we exclude *Spec2* from the correct-by-construction synthesis because of the difficulty in computing i^*. Instead, we handle *Spec2* by restricting the current density i to be smaller than a fixed upper bound $\widetilde{i^*}$, which is found experimentally. In addition, some

control action selecting heuristics are developed to incorporate *Spec2* (see section 6.5). However, since no formal guarantee can be made for achieving *Spec2*, we still consider *Spec1* in the correct-by-construction synthesis as a hard constraint.

6.3.2 Battery Energy & Power Limitations

In this part, we give some requirements regarding the battery SOC and power. These requirements are important for guaranteeing the health of the battery and its incorporation with energy management.

Spec3: (avoid) The battery stack energy should not drop below 10% or exceed 90%.

Spec4: (recurrence) Battery energy should always recover to $SOC_{\mathrm{B,target}}$ (with at most an error δ) in finite time, where $SOC_{\mathrm{B,target}}$ is a set point given by the energy management module.

This specification can be viewed as a relaxation of the charge sustaining requirement of the battery. Since we do not assume any knowledge of the driving cycle in advance, it is impossible for the battery SOC to recover to the starting level exactly at the end of the driving period (unless one restricts the battery SOC to always stay close to the target level, which is a conservative strategy). Hence, we require only that the battery SOC have the capability of recovering to the target level.

Spec5: (avoid) The battery power should not exceed peak power requirements.

Spec6: (avoid) Power for the battery charge should not exceed maximum allowable charging power. Note that by our convention, charging powers (both $P_{\mathrm{B,output}}$ and $P_{\mathrm{B,charge,max}}$) are negative.

6.3.3 Regular Operation Requirements

The following requirements are related to fuel cell temperature regulation.

Spec7: (reach-stay) Fuel cell block temperatures should reach and then stay in the target temperature range $[340, 350]$K.

By this requirement, when the fuel cell is temporarily shut down and motor power is completely delivered by the battery, we still want the fuel cell temperature to stay in the range.

Spec8: (avoid) Fuel cell block temperatures should never exceed the maximum allowable temperature of 353K.

Table 6.1: Specifications in LTL

Specification	LTL formula	type	
Spec1	$\varphi_1 = \Box(P_{\mathrm{FC,output}} \geq 0)$	avoid	✓
Spec2	$\varphi_2 = \Box(i \leq i^*)$	avoid	$\sim$
Spec3	$\varphi_3 = \Box(0.1 \leq SOC_{\mathrm{B}} \leq 0.9)$	avoid	✓
Spec4	$\varphi_4 = \Box\Diamond\big(SOC_{\mathrm{B,target}} - \delta \leq SOC_{\mathrm{B}} \leq SOC_{\mathrm{B,target}} + \delta\big)$	recurrence	✓
Spec5	$\varphi_5 = \Box(P_{\mathrm{B,output}} \leq P_{\mathrm{B,output,max}})$	avoid	✓
Spec6	$\varphi_6 = \Box(P_{\mathrm{B,output}} \geq P_{\mathrm{B,charge,max}})$	avoid	✓
Spec7	$\varphi_7 = \Diamond\Box\big(\wedge_{j=1,2}\,(T_j \in [340, 350])\big)$	reach-stay[1]	✓
Spec8	$\varphi_8 = \Box\big(\wedge_{j=1,2}\,(T_j \leq 353)\big)$	avoid	✓

"✓": the specification is considered in the synthesis.
"$\sim$": the specification is handled by heuristics.

6.4 Solution Approach

We formulate the control problem as a temporal logic game [86] on a hybrid system and solve the game using abstraction-based synthesis technique. This section is divided into three parts. First, we describe the basic steps involved in computing an abstraction, and show how to leverage the system's properties introduced in Section 2.1 at each step to simplify computation. Second, we briefly describe the synthesis

[1]We refer to this LTL specification as "reach-stay" type with a slight abuse of terminology. In fact, $\Diamond\Box(x \in$ target set) does not require that x stays in the target set after its first arrival.

process on the abstraction. Finally, we motivate and propose multi-action state-dependent progress groups, and show how they remove spurious behavior from the abstraction.

6.4.1 Abstraction

The abstraction process returns a finite transition system for a given plant model and specifications. The transitions capture the flow of the continuous plant dynamics, and the (discrete) states of the finite transition system are properly labeled according to the given specifications. In this work, the finite transition systems serving as abstractions are nondeterministic. That is, given the current state and the control action applied at that state, there might be multiple succeeding states. Specifically, we also reinforce such transition systems with so called progress groups, which encode additional transience properties of the underlying concrete system. Such transition systems are already well-established in the literature. We refer the reader to [85] for a formal definition of such abstractions and detailed algorithms generating them.

The abstraction process is decomposed into three steps, that is, discretization, labeling and transition computation. We now describe each step, incorporated with the fuel cell system properties for computational efficiency.

Discretization

We first partition the state space of a given concrete system into finitely many regions. Each region is mapped to a discrete state in the finite transition system. In the rest of this paper we will call a "discrete state" as "state" for short, when the context is unambiguous. We use a manually constructed non-uniform rectangular grid partition in the state space. A rectangular partition reduces the abstraction computation effort significantly when the system flow/vector field are (mixed) monotone [26] or multi-

affine [133].

Notice that both continuous-valued control inputs $(i, w_{\text{cool}}, P_{\text{H}})$ and boolean control input (u_{HR}) are present in the system. We discretize control space U by creating a grid on the continuous-valued control space, which leads to a finite set of control actions. It should be noted that the discretization of the continuous control variables leads to more than 70 control actions in total in this application.

Labeling

After the state space partition, each region in the partition needs to be labeled as "target", "recurrence target", "safe" and "unsafe" according to the specifications.

Consider "reach-stay" specification *Spec7*. The regions contained by set $\{x \in X \mid T_1 \text{ and } T_2 \in [340, 350]\}$ are labeled as "target". For "recurrence" specification *Spec4*, the regions contained by set $\{x \in X \mid SOC_{\text{B}} \in [SOC_{\text{B,target}} - \delta, SOC_{\text{B,target}} + \delta]\}$ are labeled as "recurrence target". The remainder of the specifications are of the "avoid" type. A region is labeled "safe" if the "avoid" specification is satisfied everywhere in that region for all operating conditions; it is labeled "unsafe" if the specification is violated somewhere in the region for some operating conditions.

The challenge is that some "avoid" specifications are implicitly related to states and operating conditions. For example, requirement *Spec1* requires fuel cell output power $P_{\text{FC,output}} \geq 0$ (or equivalently $E_{\text{FC,stack}} \geq 0$ by Eq. (6.1)), $P_{\text{FC,output}}$ is a function of both system state (fuel cell temperature T_{avg}) and operating condition (membrane water content λ, hydrogen-oxygen partial pressure $p_{\text{H}_2}, p_{\text{O}_2}$). Therefore, to label a region safe or unsafe in terms of *Spec1*, we need to check the worst case in that region. That is, if the minimum fuel cell output power $P_{\text{FC,output}}$ (or equivalently $E_{\text{FC,stack}}$) in the region is negative under some operating conditions (which violate specification *Spec1*), the region is labeled unsafe.

As described in Section 6.2.1, $E_{\text{FC,stack}}$ is a nonlinear function in state x and operating condition d. Therefore, determining the exact minimum value of $E_{\text{FC,stack}}$ requires solving a nonlinear optimization problem over x and d, which might be intractable. However, function $E_{\text{FC,stack}}(x, d)$ allows an efficient and reasonable approximation by Theorem 2 in Section 2.1 if the considered regions are rectangles. Theorem 2 applies to function $E_{\text{FC,stack}}(x, d)$ because $E_{\text{FC,stack}}(x, d)$ is continuously differentiable w.r.t. x and d on compact set $X \times D$. This means that all the continuous partial derivatives $\frac{\partial E_{\text{FC,stack}}}{\partial x}$ $\frac{\partial E_{\text{FC,stack}}}{\partial d}$ are bounded on $X \times D$; thus, $E_{\text{FC,stack}}(x, d)$ satisfies the hypothesis of Theorem 2.

By Theorem 2, under-(over-)approximating the minimum (or maximum) value of $E_{\text{FC,stack}}$ reduces to evaluating $E_{\text{FC,stack}}$ at the two extreme points of the considered rectangular region. The result of the approximation is illustrated using Fig. 6.4: the dashed line is the maximum and minimum value when x_1, x_2, or T_1, T_2 varies in $[273, 360]$K. Fig. 6.4 shows a gap between the approximated minimum value of

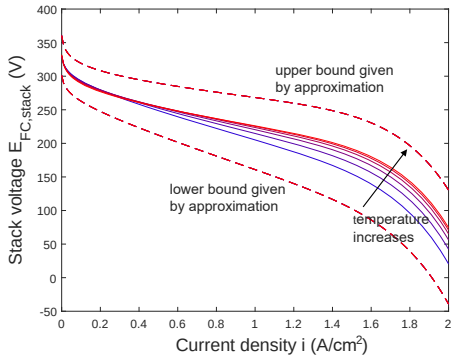

Figure 6.4: Approximation of polarization, fuel cell average temperature varies in [273, 360]K.

$E_{\text{FC,cell}}$ and the true values. This gap indicates that the approximation is conservative. However, when the size of the region to be labeled is smaller, the approximation gets tighter.

Computing Transitions

We compute transitions in the abstraction by arguing about the vector field directions of the concrete system, over a region in state space, and also over all operating conditions. In this part we provide an efficient way to compute these transitions using Theorems 2 from Section 2.1.

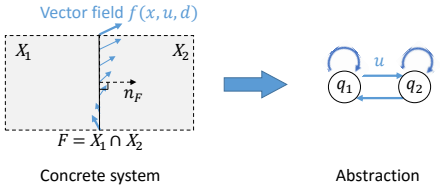

Figure 6.5: Computing transitions by arguing direction of vector field.

As shown in the left half of Fig. 6.5, X_1 and X_2 are two adjacent regions in the state space of concrete system, $F = X_1 \cap X_2$ is the adjacent facet between two regions, dashed arrow n_F is the normal vector of facet F (pointing from X_1 to X_2), and the solid arrows on F are the vector field $f(x, u, d)$ under some given u and operating conditions d. The right half of the figure shows the discrete states in the abstraction, in particular, discrete state q_1 (q_2) corresponds to region X_1 (X_2), and the transitions between q_1 and q_2 are defined as follows

$$q_1 \xrightarrow{u} q_2 \text{ if } \max_{\substack{x \in F, \\ d \in D}} n_F^T f(x, u, d) > 0. \tag{6.11}$$

Finally, since any trajectory starting from region X_i will stay in that region after a small amount of time, we need to add a self-transition on the corresponding discrete state q_i.

Assume a rectangular partition of the state space, the adjacent facets are all rectangular facets, i.e., $F = \{x \in X \mid x_j \in [\underline{x}_j, \overline{x}_j]\}$, and their normal vectors are natural basis vectors e_i (a vector whose i^{th} entry is one, and the other entries are zeros). Also note that allowable operating condition set D is a rectangular set by

definition, i.e., $D = \{d \mid d_k \in [\underline{d}_k, \overline{d}_k]\}$. Eq. (6.11) is thus equivalent to

$$q_1 \xrightarrow{u} q_2 \text{ if } \max_{\substack{x_j \in [\underline{x}_j, \overline{x}_j] \\ d \in [\underline{d}_k, \overline{d}_k]}} f_i(x, u, d) > 0. \tag{6.12}$$

The optimum values in (6.12) can be over approximated using Theorem 2. Fixing control u, and letting ϕ^u be the decomposition function of $f(\cdot, u, \cdot)$ defined by Eq. (2.15), we have

$$\max_{\substack{x_j \in [\underline{x}_j, \overline{x}_j] \\ d \in [\underline{d}_k, \overline{d}_k]}} f_i(x, u, d) \leq \phi_i^u([\overline{x}, \overline{d}], [\underline{x}, \underline{d}]), \tag{6.13}$$

where $\underline{x} = [\underline{x}_1, \ldots, \underline{x}_n]^T$ and $\overline{x} = [\overline{x}_1, \ldots, \overline{x}_n]^T$ (similar for $\underline{d}$, $\overline{d}$). We hence replace Eq. (6.13) by the following to compute the transitions:

$$q_1 \xrightarrow{u} q_2 \text{ if } \phi_i^u([\overline{x}, \overline{d}], [\underline{x}, \underline{d}]) > 0. \tag{6.14}$$

Recall the discussion following Theorem 2, if the partial derivative $\frac{\partial f_i}{\partial x_j}$ is not sign-stable, the approximations in Eq. (6.13) are not tight. In other words we may have $\phi_i^u([\overline{x}, \overline{d}], [\underline{x}, \underline{d}]) > 0$ but $\max f_i(x, u, d) \leq 0$. In that case, Eq. (6.13) and Eq. 6.14 are not equivalent. In fact, we create more transitions when using Eq. (6.14), and hence introduce more spurious behavior in the abstraction, thus leading to a more conservative solution but conserving the correctness.

Note that the partial derivative $\frac{\partial f_3}{\partial v}$ is not sign-stable, where $f_3 = \frac{dT_{\mathrm{R}}}{dt}$ is defined by Eq. (6.7). The sign of $\frac{\partial f_3}{\partial v}$ depends on which one of T_{R} and T_{amb} is larger. In this case, the conservatism can be reduced. We show f_3 is affine in state x and multi-affine in $[T_{\mathrm{amb}}, w]$ where $w := \varepsilon(v)v$. Hence to maximize (minimize, respectively) vector field component f_3, one need only evaluate f_3 at both upper and lower bounds of w, and pick the maximum (minimum, respectively) f_3 value. This practice is equivalent to evaluating f_3 at the upper and lower bounds of vehicle speed v, because $w = \varepsilon(v)v$

is monotone increasing in v. With this modification, Eq. (6.13) becomes

$$\max_{\substack{x_j \in [\underline{x}_j, \overline{x}_j] \\ d \in [\underline{d}_k, \overline{d}_k]}} f_i(x, u, d) \leq \max\left\{\phi_i^u([\overline{x}, \overline{\overline{d}}], [\underline{x}, \underline{d}]),\ \phi_i^u([\overline{x}, \overline{d}], [\underline{x}, \underline{\underline{d}}])\right\}, \tag{6.15}$$

where $\underline{\underline{d}}$ is the same as $\underline{d}$ except that its fifth entry (representing vehicle speed v) takes upper bound value; and $\overline{\overline{d}}$ is the same as $\overline{d}$ except that its fifth entry takes lower bound value.

Note that to compute the transition from q_2 to q_1, the only change is to choose the normal vector in Eq. (6.11) to be $n_F = -e_i$, and the above approximation process still applies to this case.

6.4.2 Synthesis

To synthesize a controller, we solve a temporal logic game on the obtained abstraction using graph search algorithms. As mentioned at the beginning of Section 6.4.1, the abstraction used in this work is nondeterministic. The actual evolution of such an abstraction can be viewed as the outcome of a two player game between the controller and the environment [112]. In each round of the game, the controller selects an action first, and then the environment selects a transition that is available under the current state and action. The goal of the controller is to win the game, that is, to satisfy the specification regardless of the moves of the environment. The algorithms for solving such games are fairly standard [112, 15, 82, 118, 85, 11], and refer the readers to these references for details. Fig. 6.6 shows an illustration of the synthesis process, and we only briefly describe the process with the following three steps:

1. We first solve the "stay" part of the game, by searching for the maximal controlled invariant set C_1 within the discrete states labeled as both "target" and

"safe". Each discrete state in C_1 will be assigned a set of control actions. Under the assigned actions, the closed-loop path starting from C_1 will stay in C_1 forever. Such a maximal controlled invariant set can be found by a fixed-point algorithm given in [82].

2. Next we solve the "recurrence" part of the game, restricted within set C_1. In particular, at each state in C_1, we can use only the actions assigned to that state in step 1 that render C_1 to be invariant. The recurrence game can be solved with the algorithm given in [118]. This gives a set of states contained by C_1, starting from where the "recurrence target" states (again, in C_1) can be reached infinitely often. This set of states is denoted as C_2. As shown in Fig. 6.6, $C_2 \subseteq C_1$. It should be noted that C_2 is also a controlled invariant set when the recurrence specification is achieved.
3. Then we solve the "reach-avoid" part of the game, using the algorithm given in [82]. The solution contains a set of states B, called the backwards reachable set of C_2, together with a set of actions assigned to each state in B. If these actions are applied accordingly, a path starting from B will reach C_2 in finite time, while avoiding "unsafe" states[2]. The obtained backwards reachable set B together with set C_2 form a winning set of the overall game.

Finally, once the winning set and the associated control actions are determined on the abstraction, we can extract a switching controller for the concrete system, that is, we map the actions assigned to each "winning" state to its corresponding region in the concrete system's state space. This leads to a look-up table controller defined in the concrete system state space.

To solve the reachability part, both in the reach-avoid-stay game and the recur-

[2]As noted in Footnote 1, there is a difference between specification $\Diamond\Box(x \in \text{target set})$ and "reaching the target set in finite time and staying there once arrived". Here we are achieving the latter one, which implies the satisfaction of the former one.

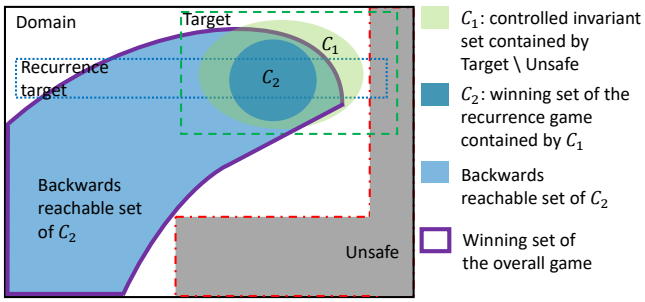

Figure 6.6: Illustration of synthesis procedure.

rence game, the existence of spurious loops in abstraction may prevent the target from being reached, therefore reducing the winning set. Hence, it is crucial to eliminate the spurious behavior in the abstraction as much as possible. To this end we encode in abstraction some transient properties of the underlining continuous system by progress groups. A set of states (each state assigned a set of control actions) forms a progress group if these discrete states correspond to a transient region in the original concrete system, under the assigned actions. A region is transient under some control actions if all trajectories starting from that region eventually leave the region in finite time under assigned control actions. Fig. 6.7 illustrates different notions of progress groups. In all three illustrations, the vector field in region $X_1 \cup X_2$ is pointing upwards under the assigned actions. Thus, all trajectories starting from region $X_1 \cup X_2$ will eventually leave the region. We hence group the corresponding states $\{q_1, q_2\}$ as a progress group, and forbid any infinite path from staying within states $\{q_1, q_2\}$. Notice that infinite paths within $\{q_1, q_2\}$ exist otherwise due to the self-loops and cycling between two states. The novel concept introduced here of multi-action state-dependent progress group will be motivated and explained in more details in the following section.

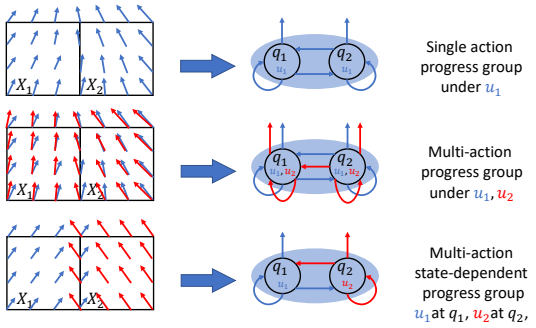

Figure 6.7: Different notions of progress groups.

6.4.3 Multi-action State-dependent Progress Group

In previous works [89], [109], progress groups were defined only for a single action. In a more recent paper [85], the notion of multi-action progress groups was introduced to encode richer transience properties of the underlying concrete system. In all of these works, progress groups were pre-computed before synthesis and are stored as part of the abstraction. These notions of progress groups, however, are not sufficient for the specific application considered in this paper. First, we find that single-action progress groups cannot accommodate the battery SOC requirement and some reachability requirements at the same time. Moreover, pre-computing and storing multi-action progress groups requires an unacceptable computation load because the number of actions in our application is relatively large. Hence, we develop a procedure here to construct multi-action progress groups on-the-fly during synthesis. This procedure leads to a more general notion called multi-action state-dependent progress groups.

We use Fig. 6.8 to illustrate why single-action progress groups are not sufficient for this application, that is, why the battery SOC requirement *Spec3* and the reachability part of requirement *Spec7* cannot be satisfied at the same time with single-action progress groups. On the left side of the figure, we plot the rectangular partition and the system's vector field projected onto SOC_{B}-T_1 space. By requirement *Spec3*,

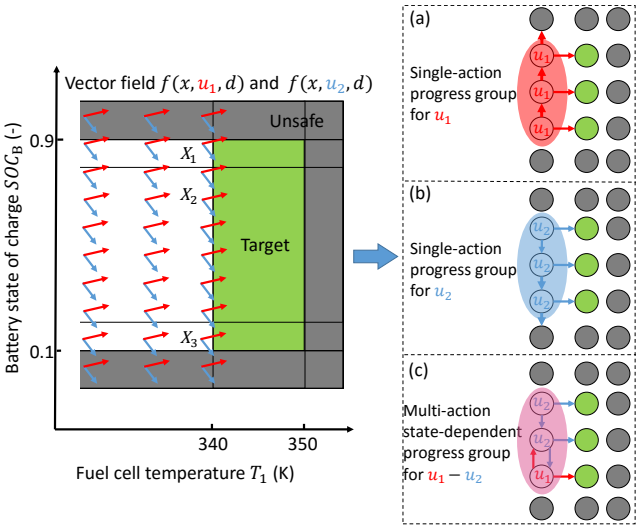

Figure 6.8: Illustration of system flow projected onto SOC_B-T_1 subspace (left), abstraction with single-action progress group (right (a), (b)) and abstraction with multi-action state-dependent progress group (right (c)). The self-loops on the discrete states are removed because they are part of some progress groups and hence correspond to transient regions.

the regions where battery SOC falls below 0.1 or exceeding 0.9 are labeled as unsafe (gray). By requirement *Spec7*, the fuel cell temperature should reach and stay between 340K to 350K, and the corresponding regions are labeled as target (green). To reach the target region, we can either (i) let the fuel cell do self-heat-up, meanwhile using the excess power generated to charge the battery (corresponding to action u_1), or (ii) use the heater to warm up the fuel cell, by drawing power from the battery (corresponding to action u_2)[3]. Note that no action can keep battery SOC constant while steering the fuel cell temperature towards the target. This is true due to the variation in motor requested power P_M, which is an operating condition (uncontrolled variable).

The right section of the Fig. 6.8 shows the abstraction, where the colored ellipses mark some progress groups and the arrows show the paths when the corresponding actions are applied. By choosing a single action (i.e., case (a) or (b)), the paths either

[3]In practice there are more than 70 actions in total, but we only plot the system's flow under two typical actions u_1 and u_2 to simplify the illustration.

go all the way up (case (a)) or down (case(b)) and lead to unsafe discrete states. Therefore battery SOC requirement *Spec3* is violated on the abstraction. Note that such paths are spurious because they do not represent any real trajectories of the concrete system (e.g., choosing u_1 at low battery SOC actually leads the trajectory into the target region before saturating the battery). Such spurious behaviors exist in abstraction due to the conservatism introduced by partitioning. On the other hand, the battery SOC requirement can be satisfied by applying multiple actions. As shown in Fig. 6.8 (c), all the paths stay safe when u_1 is applied at the bottom discrete state and u_2 is applied at the upper two discrete states. However, the reachability requirement is violated by an infinite loop caused by alternatively choosing u_1 and u_2. Again, this loop is spurious, as there is a constant flow towards the left no matter what action is chosen. We thus need multi-action (not necessarily state-dependent) progress groups to eliminate such loops when they are spurious.

Since the total number of multi-action progress groups grows exponentially in the number of available control actions (70 for this application), it easily exhausts time and memory to pre-compute these progress groups and encode them in the abstraction before synthesis. Therefore, instead of doing computation and storage before synthesis, we compute progress groups with multiple actions during the synthesis process, and we restrict the control actions on-the-fly based on the synthesis. As will be shown later, under such restrictions, the control actions assigned to different discrete states in the progress group may vary from one to the other, that is, the action is state-dependent. Algorithm 4 shows how to construct such progress groups and how to use them to compute a backwards reachable set of the target set.

Algorithm 4 is summarized with the following steps:

1. We start from an initial set B and compute its one-step-predecessors $P =$

Algorithm 4 $[B]$ = BackwardsReach(C, S, α). Compute the safe backwards reachable set of C, with multi-action state dependent progress groups constructed on-the-fly during the computation.

Input: the set C of discrete states to reach, the set S of all discrete states labeled as safe, abstraction α that maps a region in concrete system's state space to a discrete state.

Output: Set B, a backwards reachable set of set C.

1: $B = C$
2: $P = \mathbf{Pre}^1(B)$
3: $i = 0$
4: **while** B is not satisfactory **and** $i \leq$max_iter **do**
5: $(B', K) = \mathbf{CInv}\big(B \cup (P \cap S)\big)$
6: **if** $\exists v : \forall b \in B' \setminus B : \exists u_b \in K(b) : \forall x \in \alpha^{-1}(b), d \in D : v^T f(x, u, d) > 0$ **then**
7: $B' \setminus B$ is a multi-action state-dependent progress group whenever u_b is applied at $b \in B' \setminus B$
8: $B = B'$
9: $P = \mathbf{Pre}^1(B)$
10: **else**
11: Replace P by a proper subset of P that has not been used before
12: **end if**
13: $i = i + 1$
14: **end while**
15: **return** B

$\mathbf{Pre}^1(B)$. A discrete state p is called a one-step-predecessor of a set B, if there is a transition (under some actions assigned to p) leading p to some discrete state $b \in B$. Here, since we want to remain in the target set after arriving at it, set B is initialized as a controlled invariant set contained by the target.

2. Next, sub-procedure $(B', K) = \mathbf{CInv}\big(B \cup (P \cap S)\big)$ computes the largest controlled invariant set B' that is contained by set $B \cup (P \cap S)$, together with an invariance controller $K : B' \to 2^U$ that maps every discrete state in B' to a set of control actions that renders B' invariant.
3. If the discrete states in $B' \setminus B$ correspond to a transient region under some actions restricted by invariance controller K (this is checked by a sufficient condition in line 6), set $B' \setminus B$ form a multi-action state-dependent progress group, and is added to the backwards reachable set.
4. We repeat steps 1, 2, 3 until the winning set reaches a satisfactory size or a maximum number of iterations has been reached.

We explain the intuition behind Algorithm 4 as follows. For simplicity, we will

temporarily omit the "avoid" requirements. Suppose that B is indeed a backward reachable set of some controlled invariant set C, the following facts must hold:

- Set B is computed in a backwards expanding manner starting from and containing set C; hence, the paths satisfying the reach-stay requirement will always stay within set B. Thus, set B is controlled invariant under the actions that achieve the reach-stay requirement.
- For C to be reachable in finite time, no path stays in $B \setminus C$ forever. That is, set $B \setminus C$ will form a progress group under the actions that achieve the reach-stay requirement.

Therefore, in order to find a progress group that helps expanding the backwards reachable set of C, one need only explore the controlled invariant sets B' that contain C, and restrict to the action assignments that render B' invariant. These restrictions avoid exploring all subsets of discrete states with all possible action combination when computing multi-action state-dependent progress groups.

The remainder of this part explains the condition in line 6, Algorithm 4 and shows how to check this condition. Take Fig. 6.8 as an example. Shaded discrete states form a multi-action state-dependent progress group when u_2 is assigned to the two discrete states on the top and u_1 is assigned to the bottom discrete state. This is because the region represented by these discrete states is transient under corresponding actions. The transience can be checked efficiently by arguing the direction of the vector field of the underlying concrete system. As shown on left side of Fig. 6.8, the union of three regions $X_1 \cup X_2 \cup X_3$ is transient because the horizontal component of the vector field is always positive (i.e., pointing rightwards) when control u_2 is applied in X_1 X_2, and u_1 is applied in X_3. More generally, given a set of regions $\{X_k\}_{k=1}^m$ in n dimensional state space, each region equipped with one

control action u_k, $X = \bigcup_{k=1}^{m} X_k$ is transient under assigned actions if there exists $v \in \mathbb{R}^n$,

$$\forall k = 1 \ldots m : \min_{x \in X_k, d \in D} v^T f(x, u_k, d) > 0. \tag{6.16}$$

If vector $v = \pm e_i$ (the vector with the i^{th} entry being 1 and the rest being 0), and if X_k's are rectangles, the optimization problem in (6.16) can be approximated efficiently by the approach developed in section 6.4.1.

6.5 Results and Discussion

Using the solution approach described in Section 6.4, a switching controller is synthesized. The controller is in the form of a look-up table (see Step 3 in Section 6.4.2). At each time instant, the current state locates in one of the regions of the look-up table and the control action in that region is applied accordingly. Although multiple actions might be available in the region, selecting an arbitrary one is sufficient to guarantee the correctness. We implement a "lazy switching" heuristic for action selection to reduce the change in the control inputs, allowing us to always maintain the previously used control action, where possible. Whenever we enter a new region in the look-up table and the previous action is no longer available there, we change the action but try to maintain the position of as many actuators as possible. In particular, we can assign different priority to different actuators in terms of maintaining their current positions. If there is no such priority, we always tend to fix the actuator whose position has changed most recently. This practice helps to balance the position change of different actuators, avoiding frequent switches. The action selecting heuristics can be also designed to incorporate with specification *Spec2*, which is excluded from the correct-by-construction synthesis. Since *Spec2* requires that the

fuel cell current density i is smaller than i^*, we can always select the control actions with the smallest current density i to avoid violating *Spec2*.

The controller is shown to be able to achieve the specifications on the entire state domain X. That is, the computed winning set is equal to the domain X. We illustrate the closed-loop behaviors by simulating the system at 285K. We modify FTP-72 vehicle speed data [115] to obtain motor power P_{M} and vehicle speed v profile in a driving cycle. First, since the controller allows the vehicle speed v to vary only in [5,25]m/s, we saturate the v whenever its value falls below 5m/s. Second, P_{M}, the power requested by the motor, is assumed to be proportional to acceleration $\frac{dv}{dt}$ whenever the acceleration is positive, and is assumed to be 0 whenever the acceleration is negative. P_{M} is also scaled so that the value lies in the allowable range, that is, $[0, 17]$kW.

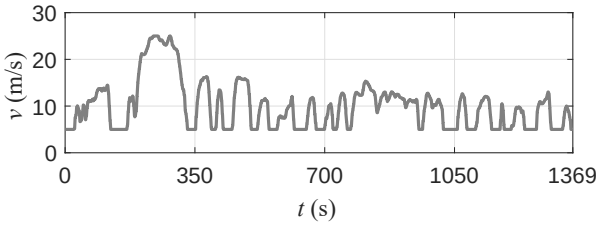

Figure 6.9: FTP-72 vehicle speed data, saturated to fit the operating condition constraints of the synthesized controller.

The simulation results are plotted in Figure 6.10, from which we make the following observations:

1. By plots (1-1) (1-2) (5-2), all states stay in the domain, and the battery SOC never exceeds upper or lower bounds.
2. By plot (1-1) fuel cell temperature reaches and stays in the target range (marked by a dashed green line).
3. By plot (5-2), the battery SOC recurrently visits the reference interval marked by the dashed blue lines. Here we assume the reference can vary over time, and is

determined by a higher level power management module. Note that the battery SOC need not stay in the interval after arriving, even if the reference has not yet changed: it simply maintains the capability to recover to the desired level. This can be seen from the simulation before 300s. Such behavior is desired because it gives more freedom to the controller, while guaranteeing that the battery SOC is able to recover. For example, while the fuel cell stack does self-heat-up at the warm-up stage, it generates more power than requested with the extra power stored in the battery. This explains why the battery SOC increases and exceeds the reference interval at the beginning of the simulation. However, once the target temperature is reached, the battery SOC begins to drop towards the reference interval.

4. By plot (2-2), it can be seen that the heater is not turned on at the warm-up stage. The fuel cell does self-heat-up instead, indicating that while the controller is not optimal in terms of warm-up speed, it does not harm the correctness of the controller.

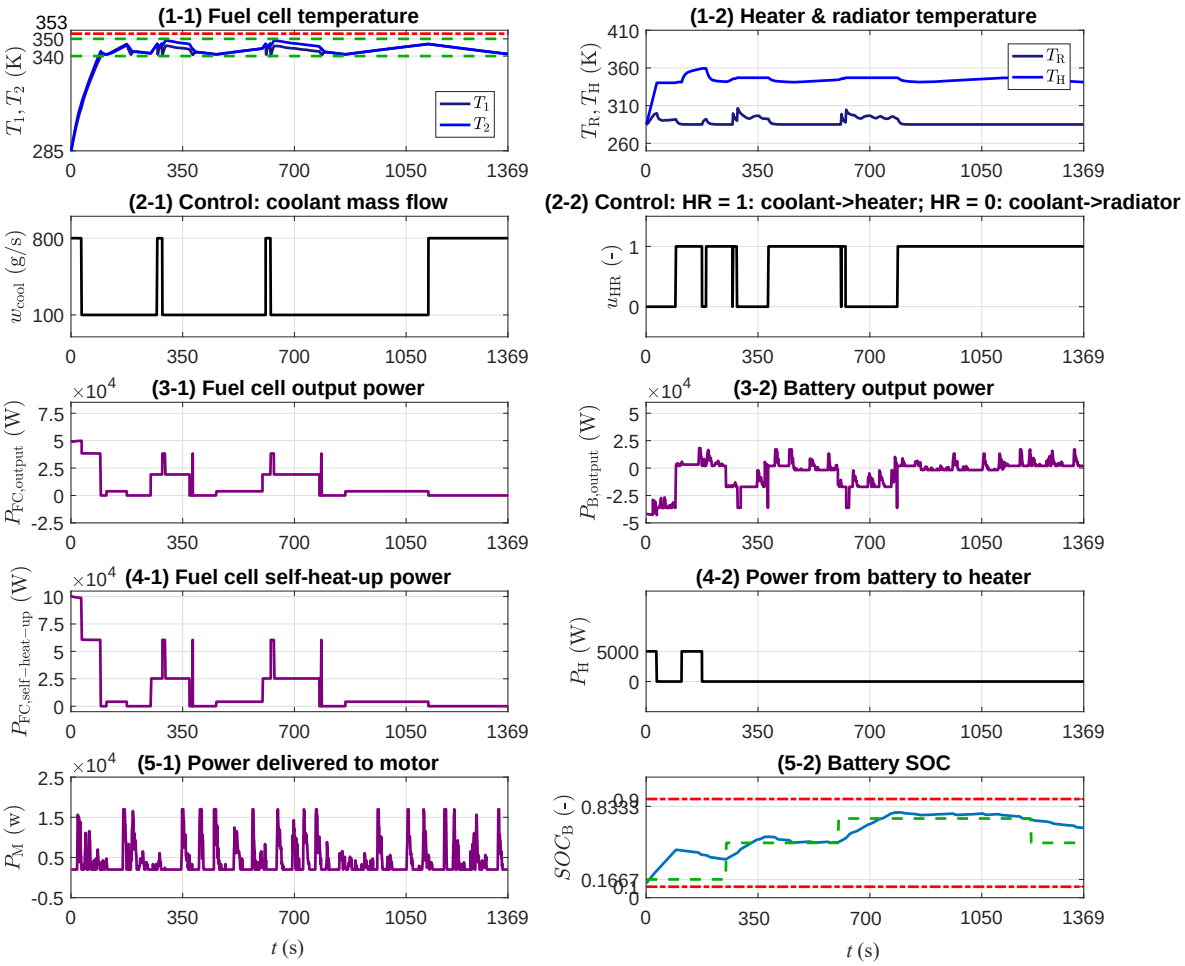

Figure 6.10: Simulations results: states, powers and selected controls. Temperature states start from 285K and battery SOC starts from 0.105.

CHAPTER VII

Fault-tolerant Path Planning

In this chapter, we consider the fault tolerant path planning problem for linear systems to satisfy some high level requirements specified by LTL. Unlike the problems solved in Chapter V where a winning set (i.e., a set of initial conditions starting from where the specification is achievable) is to be searched, an initial condition is given as part of the problem setup. This allows one to leverage Mixed Integer Linear Programming (MILP) based LTL encoding to search for an open-loop strategy that achieves the specification, and the fault-tolerant path planning problem is solved under this framework. We first show how open-loop fault tolerant strategies (associated with each initial state) can be synthesized by solving an MILP. These open-loop strategies, however, are not robust to the disturbances because of two reasons. First, since the disturbed system cannot be predicted precisely, the fault will be detected with a delay. Secondly, even if the faulty status is known, the true system trajectory may still deviate from the planned trajectory as the impact of the disturbance accumulates. To solve the two problems, we present a MILP formulation of the problem that incorporates finite detection delays, the open-loop strategy defined by the MILP's solution is then robustified with additional linear regulation.

Chapter overview. This chapter is organized as follows. Section 7.1 defines the fault-tolerant path planning problem. Section 7.2 introduces preliminaries on MILP encoding of LTL. Section 7.3 presents the proposed MILP based approach for fault-tolerant path planning problem. Section 7.4 presents the low level regulator design approach via optimization. Finally the proposed algorithm is illustrated with a motion planning on 2D plane.

Related work. The MILP-based approach considered in this chapter follows [58, 95, 119] and is also closely related to the constraint control of mixed-logical systems [12]. As mentioned in Chapter I, it aligns with [96, 100, 39, 104, 99] that try to make the obtained strategy more robust and reactive to environment changing. The use of similar hierarchical control architectures has become a common practice in many application domains [19]. For simple tasks like reaching a goal while avoiding obstacles that may appear on the fly, approaches based on LQR trees [111] or contraction theory [105] have been proposed along these lines.

7.1 Simplified Problem Setup

In this section, we define the fault tolerant path planning problem. The problem has two ingredients: (i) a system whose dynamics can degrade suddenly due to a fault, and (ii) an LTL formula that specifies the system's "graceful degradation".

System Model

The system model considered in this paper is defined by

$$\Sigma: \ x_{t+1} = A^{\sigma_t} x_t + B^{\sigma_t} u_t + F^{\sigma_t} + w_t, \tag{7.1}$$

$$\sigma_{t+1} \begin{cases} \in \{\mathrm{h}, \mathrm{f}\} & \text{if } \sigma_t = \mathrm{h} \\ = \mathrm{f} & \text{if } \sigma_t = \mathrm{f} \end{cases}. \tag{7.2}$$

where $x_t \in X \ \subseteq \mathbb{R}^n$ is the state, $u_t \in U \subseteq \mathbb{R}^m$ is the control input, $w_t \in W \subseteq \mathbb{R}^n$ is the disturbance, and $\sigma_t \in \{\mathrm{h}, \mathrm{f}\}$ is the fault status of the system. If $\sigma_t = \mathrm{h}$, the system is healthy and evolves with the dynamics defined by $(A^{\mathrm{h}}, B^{\mathrm{h}}, F^{\mathrm{h}})$; if $\sigma_t = \mathrm{f}$, this indicates that the fault has occurred and the system evolves with the dynamics defined by $(A^{\mathrm{f}}, B^{\mathrm{f}}, F^{\mathrm{f}})$. By Eq. (7.2), the fault is permanent because σ_t never recovers to h after it becomes f. In addition, we make the following assumption on the faults.

Assumption 3. We assume that the fault is T-detectable ([47]), that is, if the fault occurs at time step t_{o}, it will be detected at t_{d} where $t_{\mathrm{o}} + 1 \leq t_{\mathrm{d}} \leq t_{\mathrm{o}} + T$.

For Assumption 3, it should be noticed that T is only an upper bound on the detection delay, and the actual online detection can be earlier than $t_{\mathrm{o}} + T$. However, this fact cannot be incorporated in the offline path planning phase because the actual detection depends on the realization of w_t.

7.2 Mixed Integer Encoding of LTL

Given an LTL formula φ, it is well-known from the literature [119] that $\mathbb{x} \models \varphi$ can be encoded with mixed integer linear constraints in the following sense. Instead of imposing constraints on the infinite sequence $\mathbb{x}$, we search for a finite sequence $\mathbf{x} = x_1 x_2 \ldots x_k, x_{k+1 \ldots} x_N$ that satisfies the following linear inequality constraint:

$$H_{\varphi,k,N}(\mathbf{x}, \mathbf{b}) \leq 0, \tag{7.3}$$

where $\mathbf{b}$ is an N-sequence of auxiliary binary (hence integer) vectors[1], and $H_{\varphi,k,N}$ is an affine function in $(\mathbf{x}, \mathbf{b})$. In particular, $H_{\varphi,k,N}$ is constructed in such a way that the infinite sequence $\mathbb{x} := (x_1 x_2 \dots x_k)(x_{k+1} \dots x_N)^\omega$, obtained by unfolding $\mathbf{x} = x_1 x_2 \dots x_N$ at point k, is guaranteed to satisfy φ. In this case, we say that finite sequence $\mathbf{x}$ satisfy φ with a slight abuse of terminology.

With such mixed integer encoding technique, the path planning problem of linear systems can be formulated as a MILP. Let the system model be $x_{t+1} = Ax_t + Bu_t + F$ with state $x \in X$ and control $u \in U$, where $X \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^m$ are polytopes, also let x_{init} be the initial state, the MILP formulation is given by

$$
\begin{aligned}
\text{find} \quad & \mathbf{x}, \mathbf{u}, \mathbf{b} \\
\text{s.t.} \quad & \exists 1 \le k \le N-1 : H_{\varphi,k,N}(\mathbf{x}, \mathbf{b}) \le 0 \text{ and} \\
& A\mathbf{x}(N) + B\mathbf{u}(N) + F = \mathbf{x}(k+1), \\
& \mathbf{x}(t+1) = A\mathbf{x}(t) + B\mathbf{u}(t) + F, \ t = 1, \dots, N-1, \\
& \mathbf{u}(t) \in U, \, \mathbf{b}(t) \in \{0,1\}, \ t = 1, \dots, N, \\
& \mathbf{x}(t) \in X, \ t = 1, \dots, N, \\
& \mathbf{x}(1) = x_{\text{init}}.
\end{aligned}
\tag{7.4}
$$

Suppose that the above optimization problem is feasible and let $(\mathbf{x}, \mathbf{u}, \mathbf{b})$ be one of its solutions, an infinite control sequence $\mathbb{u}$ can be extracted from the finite sequence $\mathbf{u}$, by unfolding $\mathbf{u}$ at some point $1 \le k \le N-1$. This control sequence leads to an infinite state sequence $\mathbb{x}$ that satisfies the linear dynamics and LTL specification φ.

While formulating the MILP in Eq. (7.4), it is a common practice to modify the state space labeling with an extra Δ-margin so that the specification is satisfied robustly [36, 74, 64]. Such modification provides robustness against uncertainties like disturbances or errors due to the sampling of a continuous-time system. For example,

[1]In practice, some variables in $\mathbf{b}$ need not be restricted as binary in the formulation. Instead, they can be specified as real and will be binary automatically as a result of the encoding.

if an unsafe region is required to be avoided (i.e., $\varphi = \Box\neg\pi_{\text{unsafe}}$) under uncertainties, one can expand the unsafe set $X_{\text{unsafe}} := \{x \in \mathbb{R}^n : \pi_{\text{unsafe}} \in L(x)\}$ by Δ, i.e., define $\overline{X}_{\text{unsafe}} := X_{\text{unsafe}} \oplus \{x : \|x\| \leq \Delta\}$, where We $X \oplus Y := \{x+y : x \in X, y \in Y\}$ is the Minkowski sum of set $X, Y \subseteq \mathbb{R}^n$. Then avoiding $\overline{X}_{\text{unsafe}}$ means that X_{unsafe} is avoided with Δ-margin. Similarly, if some target region is required to be reached (i.e, $\varphi = \Diamond\pi_{\text{target}}$) robustly, one can shrink the target set $X_{\text{target}} := \{x \in \mathbb{R}^n : \pi_{\text{target}} \in L(x)\}$ into $\underline{X}_{\text{target}} := X_{\text{target}} \ominus \Delta$, where ,$X \ominus Y := \{x : \{x\} \oplus Y \subseteq X\}$ is the Minkowski difference of set X and Y. Reaching the shrunk set $\underline{X}_{\text{target}}$ guarantees that set X_{target} is reached with Δ-margin. To do such expansion and shrinking systematically for arbitrary LTL formulas, one needs to rewrite the specification φ in positive normal form [36]. If atomic proposition π has no negation in the front, we shrink the set $X_\pi := \{x \in \mathbb{R}^n : \pi \in L(x)\}$ by Δ; and if π has a negation in the front, we expand the set X_π by Δ.

7.3 MILP Formulation of Fault Tolerant Path Planning

In this section, we formulate the fault tolerant path planning problem as a MILP. The system is assumed to be undisturbed (an assumption to be relaxed in the next subsection) and have a fault that is T-detectable. In addition, the labeling of the state space is robustified with a Δ-margin. We will also assume that there is a way (to be presented in the next subsection) to keep the true, disturbed trajectories Δ-close to a nominal trajectory as long as the system's fault status does not change.

We begin by sketching the strategy that achieves specification Φ in Eq. (4.8). To satisfy Φ, the system can either stay in the healthy mode forever and satisfy φ^{h}, or enter faulty mode at some time t_{o} and start to satisfy φ^{f} from then on. However, since the fault is beyond our control, we can only respond to the fault occurrence

passively. In particular, as long as the fault has not been detected yet, there is a chance that the the system is healthy and will be healthy forever. Hence we need to achieve specification φ^{h} for the healthy mode in this case. On the other hand, once the fault is detected, the first half of Φ (i.e., $(\Box\neg\pi^{\mathrm{f}}) \wedge \varphi^{\mathrm{h}}$) can no more be satisfied. Hence the strategy needs to respond to the fault by rendering the system to satisfy φ^{f}.

The above analysis leads to a strategy visualized in Fig. 7.1 (upper part). Roughly speaking, strategy $\overline{\mathbf{u}}$ should contain two pieces: a sequence $\overline{\mathbf{u}}^{\mathrm{h}}$ (black) that achieves φ^{h} under the healthy dynamics, and a sequence $\overline{\mathbf{u}}^{\mathrm{f}}$ (gray) that achieves φ^{f} under the faulty dynamics. The two sequences $\overline{\mathbf{u}}^{\mathrm{h}}$ and $\overline{\mathbf{u}}^{\mathrm{f}}$ can have different length (N^{h}, N^{f} respectively), and both of them can be unfolded to obtain infinite sequences (the loops that are to be unfolded are marked with dashed line arrows in Fig. 7.1). In addition, there should be different control sequences $\overline{\mathbf{u}}^{\mathrm{f}}$ associated with different time instants of detection because our strategy should respond to the fault detected at anytime. We denote each control sequence associated with detection time t_{d} by

$$\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}]. \tag{7.5}$$

Several important remarks on $\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}]$ are made in what follows.

First, it should be noticed that sequence $\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}]$ starts from time t_{d}, but it may correspond to any fault that occurs at $\min\{1, t_{\mathrm{d}} - T\} \leq t_{\mathrm{o}} \leq t_{\mathrm{d}} - 1$, where $\min\{1, t_{\mathrm{d}} - T\}$ is the earliest possible fault occurrence time that associates with t_{d} given T-detectability assumption. All of these fault occurrences associated with the same t_{d} cannot be treated separately because the exact fault occurrence time t_{o} is not known in general. Instead, these different fault occurrences are all controlled with $\overline{\mathbf{u}}^{\mathrm{h}}$ and $\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}]$ in the following way:

- Within the so called "uninformed execution horizon" (i.e., $\min\{1, t_{\mathrm{d}} - T\} \leq t \leq$

$t_{\mathrm{d}} - 1$), we do not know the system is already faulty and have to apply $\overline{\mathbf{u}}^{\mathrm{h}}$ until time $t_{\mathrm{d}} - 1$.

- Starting from time t_{d}, the fault is known and $\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}]$ is applied.

With the above control strategy, each occurrence time t_{o} corresponds to a different trajectory generated under the faulty dynamics, denote by

$$\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}], \tag{7.6}$$

and our goal is to ensure that $\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ can be unfolded into an infinite sequence that satisfies φ^{f} for all t_{d} and $\min\{1, t_{\mathrm{d}} - T\} \leq t_{\mathrm{o}} \leq t_{\mathrm{d}} - 1$.

The family of sequence $\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ associated with a fixed t_{d} has a notable property, that is, they all behave approximately the same as one sequence $\overline{\mathbf{x}}^{\mathrm{h}}$ within the uninformed execution horizon, where $\overline{\mathbf{x}}^{\mathrm{h}}$ denotes the sequence generated by $\overline{\mathbf{u}}^{\mathrm{h}}$ under the healthy dynamics. In particular, $\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ will be Δ-close to $\overline{\mathbf{x}}^{\mathrm{h}}$ at least until $t_{\mathrm{d}} - 1$. This has to be true given the assumption that the disturbed healthy trajectories is always Δ-close to the nominal trajectory $\overline{\mathbf{x}}^{\mathrm{h}}$ as long as the system is healthy. Consequently, the fault should be detected as long as the real trajectory is outside the Δ-tube of $\overline{\mathbf{x}}^{\mathrm{h}}$. The fact that $\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ is close to $\overline{\mathbf{x}}^{\mathrm{h}}$ within the uninformed execution horizon is highlighted with the blue box in the lower part of Fig. 7.1.

The second remark is about the faults that are detected on the loop. These situation needs to be handled with extra care because it may correspond to multiple fault detections after unfolding the loop. In Fig. 7.1, for example, if the fault is detected at the black node denoting $\overline{\mathbf{x}}^{\mathrm{h}}(6)$, the detection time $t_{\mathrm{d}} = 6 + ml$ where l is the length of the loop and $m = 0, 1, 2, \ldots$. These cases need to be handled separately. In particular, all the cases with $ml \geq T$ can be treated as one. The number of cases is hence reduced to finitely many and we only need to find $\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}]$ for $1 \leq t_{\mathrm{d}} \leq N^{\mathrm{h}} + T$.

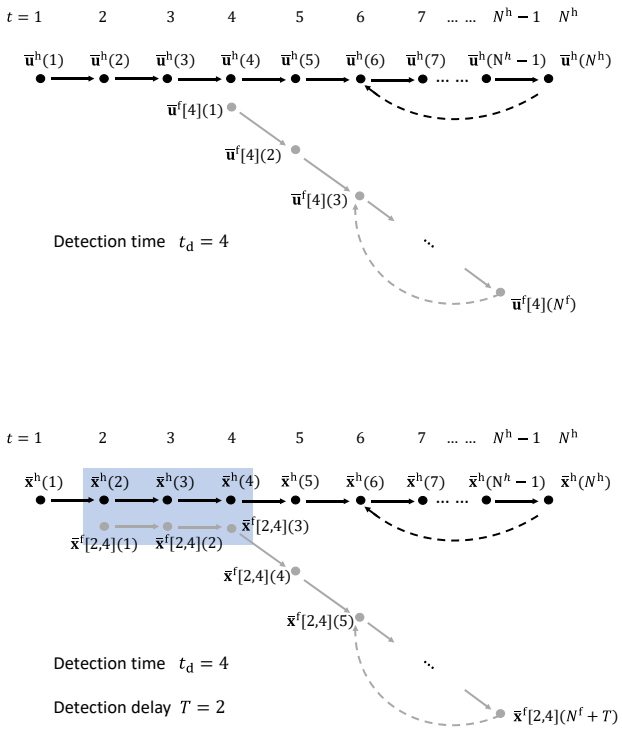

Figure 7.1: An Illustration of the fault tolerant path planning strategy (upper) and associated trajectories (lower).

In what follows, we transform the above descriptions of the strategy into MILP formulations.

Healthy Mode Path Planning

For the healthy mode control sequence $\overline{\mathbf{u}}^{\mathrm{h}}$ to achieve healthy specification φ^{h} under

the healthy dynamics defined by $(A^{\mathrm{h}}, B^{\mathrm{h}}, F^{\mathrm{h}})$, one needs the following constraints:

$$
\begin{aligned}
&\exists 1 \leq k^{\mathrm{h}} \leq N^{\mathrm{h}} - 1 : \\
&\quad H_{\varphi^{\mathrm{h}}, k^{\mathrm{h}}, N^{\mathrm{h}}}\left(\overline{\mathbf{x}}^{\mathrm{h}}, \mathbf{b}^{\mathrm{h}}\right) \leq 0 \text{ and} \\
&\quad A^{\mathrm{h}}\overline{\mathbf{x}}^{\mathrm{h}}(N^{\mathrm{h}}) + B^{\mathrm{h}}\overline{\mathbf{u}}^{\mathrm{h}}(N^{\mathrm{h}}) + F^{\mathrm{h}} = \overline{\mathbf{x}}^{\mathrm{h}}(k+1), \\
&\quad \overline{\mathbf{x}}^{\mathrm{h}}(t) = \overline{\mathbf{x}}^{\mathrm{h}}(t - N^{\mathrm{h}} + k), \overline{\mathbf{u}}^{\mathrm{h}}(t) = \overline{\mathbf{u}}^{\mathrm{h}}(t - N^{\mathrm{h}} + k), \\
&\quad t = N^{\mathrm{h}}, \ldots, N^{\mathrm{h}} + T, \qquad (7.7) \\
&\overline{\mathbf{x}}^{\mathrm{h}}(t+1) = A^{\mathrm{h}}\overline{\mathbf{x}}^{\mathrm{h}}(t) + B^{\mathrm{h}}\overline{\mathbf{u}}^{\mathrm{h}}(t) + F^{\mathrm{h}}, \\
&\quad t = 1, \ldots, N^{\mathrm{h}} - 1, \qquad (7.8) \\
&\overline{\mathbf{x}}^{\mathrm{h}}(1) = x_{\mathrm{init}}, \qquad (7.9)
\end{aligned}
$$

where $\overline{\mathbf{x}}^{\mathrm{h}}$ is the trajectory generated by the healthy dynamics under $\overline{\mathbf{u}}^{\mathrm{h}}$, $H_{\varphi^{\mathrm{h}}, k^{\mathrm{h}}, N^{\mathrm{h}}}$ is a function, encoding the specification φ^{h} within a horizon of length N^{h}, that is affine in both $\overline{\mathbf{x}}^{\mathrm{h}}$ and auxiliary variables $\mathbf{b}^{\mathrm{h}}$, and x_{init} is the initial state. Note that we extend $\overline{\mathbf{x}}^{\mathrm{h}}$ and $\overline{\mathbf{u}}^{\mathrm{h}}$ by T in Eq. (7.7) to handle the detection on the loop.

Faulty Mode Path Planning

We also require all the possible faulty trajectories $\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ to satisfy φ^{f} when they are controlled by the faulty mode control $\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}]$, which leads to the following constraints

for $1 \leq t_{\mathrm{d}} \leq N^{\mathrm{h}} + T$ and $\min\{1, t_{\mathrm{d}} - T\} \leq t_{\mathrm{o}} \leq t_{\mathrm{d}} - 1$:

$$\exists 1 \leq k^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}] \leq N^{\mathrm{f}} :$$

$$H_{\varphi^{\mathrm{f}}, k^{\mathrm{f}}, N^{\mathrm{f}}} \left(\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}], \mathbf{b}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]\right) \leq 0 \text{ and}$$

$$A^{\mathrm{f}}\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}](N^{\mathrm{f}}) + B^{\mathrm{f}}\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}](N^{\mathrm{f}}) + F^{\mathrm{f}} = \overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}](k^{\mathrm{f}} + 1), \tag{7.10}$$

$$\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}](t+1) = A^{\mathrm{f}}\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}](t) + B^{\mathrm{f}}\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}](t) + F^{\mathrm{f}},$$

$$t = T, \ldots, N^{\mathrm{f}} - 1, \tag{7.11}$$

$$\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}](t) = \overline{\mathbf{x}}^{\mathrm{h}}(t + t_{\mathrm{o}} - 1),$$

$$t = 1, \ldots, T, . \tag{7.12}$$

where $H_{\varphi^{\mathrm{f}}, k^{\mathrm{f}}, N^{\mathrm{f}}}$ encodes specification φ^{f} within horizon of length N^{f}. In particular, Eq. (7.12) requires the first T points in sequence $\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ overlaps with the corresponding points in healthy sequence $\overline{\mathbf{x}}^{\mathrm{h}}$. This constraint captures the fact that $\overline{\mathbf{x}}^{\mathrm{h}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ stays close to $\overline{\mathbf{x}}^{\mathrm{h}}$ within the uninformed execution horizon. This constraint hence couples constraints (7.7)-(7.9) with constraints (7.10)-(7.11).

In summary, let $\overline{\mathbf{u}}^{\mathrm{f}}$ ($\overline{\mathbf{x}}^{\mathrm{f}}$, respectively) be the vector obtained by stacking $\overline{\mathbf{u}}^{\mathrm{f}}[t_{\mathrm{d}}]$ ($\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$, respectively) for all t_{d} ($t_{\mathrm{o}}, t_{\mathrm{d}}$, respectively), the fault tolerant path planning for nominal system with detection delay can be formulated as the following MILP:

$$\begin{aligned}
\text{find} \quad & \overline{\mathbf{x}}^{\mathrm{h}}, \overline{\mathbf{x}}^{\mathrm{f}}, \overline{\mathbf{u}}^{\mathrm{h}}, \overline{\mathbf{u}}^{\mathrm{f}}, \mathbf{b}^{\mathrm{h}}, \mathbf{b}^{\mathrm{f}} \\
\text{s.t.} \quad & \text{Eq. (7.7)-(7.12)}, \\
& \overline{\mathbf{x}}^{\mathrm{h}}(t), \overline{\mathbf{x}}^{\mathrm{f}}(t) \in X, \quad \forall t, \\
& \overline{\mathbf{u}}^{\mathrm{h}}(t), \overline{\mathbf{u}}^{\mathrm{f}}(t) \in U, \quad \forall t, \\
& \mathbf{b}^{\mathrm{h}}(t), \mathbf{b}^{\mathrm{f}}(t) \in \{0, 1\}, \quad \forall t.
\end{aligned} \tag{7.13}$$

Comparing to regular path planning MILP formulation, the fault-tolerant path planning MILP has more constraints and variables. Let $n_{\mathrm{C}}^{\mathrm{f}}$, $n_{\mathrm{B}}^{\mathrm{f}}$ and m^{f} be the number of continuous and binary variables and constraints respectively, which are required

to encode the faulty mode LTL specification of a path of length N^{f}, the extra number of continuous and binary variables, and the number of constraints added on top of regular path planning MILP is $\mathcal{O}(N^{\mathrm{h}} n_{\mathrm{C}}^{\mathrm{f}})$, $\mathcal{O}(N^{\mathrm{h}} n_{\mathrm{B}}^{\mathrm{f}})$ and $\mathcal{O}(N^{\mathrm{h}} m^{\mathrm{f}})$ respectively. Particularly, these three quantities do not depend on the detection delay T.

7.4 Robustification of MILP Solution via Regulation

From the previous section, we formulate an MILP whose solution leads to a nominal trajectory $\overline{\mathbf{x}}$ that satisfies the specification Φ. In particular, we allow the real trajectory $\mathbf{x}$ to deviate at most Δ from the nominal (i.e., $\|\overline{\mathbf{x}}(t) - \mathbf{x}(t)\| \leq \Delta$) while still satisfying Φ. In this section, we show how to find the minimum margin Δ that can be achieved by a linear regulator and the corresponding regulation gain by solving a quasi-convex optimization problem.

In what follows we consider system

$$x_{t+1} = Ax_t + Bu_t + w_t, \tag{7.14}$$

where w_t is disturbance satisfying $\|w_t\| \leq d$ for all t, and A, B can refer to the system matrices for the healthy system or the faulty system. Note that the constant offset term F in Eq. (7.1) is dropped because it only shifts the equilibrium of the system and makes no difference when the regulation is of our concern. We call a system to be nominal if $w_t = 0$ for all t. Given an initial state x_{init} and an open-loop strategy $\overline{\mathbf{u}} = \overline{u}_1 \overline{u}_2 \dots \overline{u}_N$, the trajectory $\overline{\mathbf{x}} = \overline{x}_1 \overline{x}_2 \dots \overline{x}_N$ generated by nominal system is governed by

$$\overline{x}_{t+1} = A\overline{x}_t + B\overline{u}_t, \tag{7.15}$$

$$\overline{x}_1 = x_{\mathrm{init}}. \tag{7.16}$$

Under the given open-loop strategy, the actual trajectory may deviate from the planned nominal trajectory in the presence of nonzero disturbance w_t. Moreover, such deviation may accumulate with time because there is no feedback. In this work, we introduce feedback to keep the actual trajectory close to the planned nominal trajectory $\overline{\mathbf{x}}$ as time evolves. The block-diagram of the overall hierarchical closed-loop system is shown in Fig. 7.2. Instead of applying nominal control $\overline{u}_t$ directly to the system, we use

$$u_t = \overline{u}_t + K(\hat{x}_t - \overline{x}_t) \tag{7.17}$$

where K is the state feedback gain, $\hat{x}_t$ is the estimated state that is assumed to satisfy

$$\|\hat{x}_t - x_t\| \leq E. \tag{7.18}$$

Our goal is to design feedback gain K, so that the difference between the actual trajectory $\mathbf{x}$ and the planned nominal trajectory $\overline{\mathbf{x}}$ is bounded by a constant Δ over time.

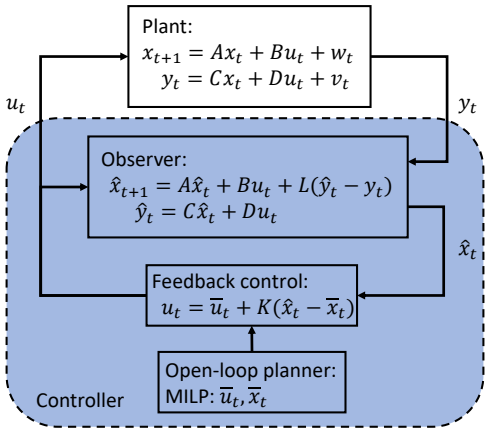

Figure 7.2: Block-diagram of the closed-loop system (the extension with output feedback can be found in the Appendix).

The rest of this section focuses on designing K such that the uniform bound on

$\|\overline{x}_t - x_t\|$ is minimized. Combining Eq. (7.14), (7.15), (7.17), we have

$$x_{t+1} - \overline{x}_{t+1} = (A + BK)(x_t - \overline{x}_t) + BK(\hat{x}_t - x_t) + w_t, \tag{7.19}$$

which implies

$$\begin{aligned}
\|x_{t+1} - \overline{x}_{t+1}\| \leq & \|(A + BK)(x_t - \overline{x}_t)\| \\
& + \|BK(\hat{x}_t - x_t)\| + \|w_t\| \\
\leq & \|A + BK\|_* \|x_t - \overline{x}_t\| \\
& + \|BK\|_* \|\hat{x}_t - x_t\| + \|w_t\| \\
\leq & \|A + BK\|_* \|x_t - \overline{x}_t\| \\
& + \|BK\|_* E + d,
\end{aligned} \tag{7.20}$$

where d is the bound of $\|w_t\|$ and E is the error bound on the state estimation from Eq. (7.18). Let Δ be the desired bound on $\|x_t - \overline{x}_t\|$, we require the following recurrence relation:

$$\|x_t - \overline{x}_t\| \leq \Delta \Rightarrow \|x_{t+1} - \overline{x}_{t+1}\| \leq \Delta. \tag{7.21}$$

Eq. (7.20), (7.21) hence suggests that

$$\|A + BK\|_* \leq \frac{\Delta - d - \|BK\|_* E}{\Delta}. \tag{7.22}$$

To minimize Δ, we formulate an optimization problem,

$$\begin{aligned}
\text{minimize}_{\delta, K} \quad & \delta \\
\text{s.t.} \quad & \|A + BK\|_* \leq \tfrac{\delta - d - \|BK\|_* E}{\delta} \\
& \delta \geq 0
\end{aligned} \quad . \tag{P1}$$

Propoition 15. The optimization problem (P1) is equivalent to the following quasi-convex optimization problem:

$$\begin{aligned}
\text{minimize}_K \quad & \tfrac{d + \|BK\|_* E}{1 - \|A+BK\|_*} \\
\text{s.t.} \quad & \|A + BK\|_* \leq 1
\end{aligned} \quad . \tag{P2}$$

Proof. We first prove the equivalence between the optimization problems (P1) and (P2). Then, we prove that the objective function of the second problem is quasi-convex.

First note that d and E are nonnegative, we hence have

$$\begin{array}{ll} \|A+BK\|_* \leq \frac{\delta-d-\|BK\|_*E}{\delta} & \Leftrightarrow \quad \delta \geq \frac{d+\|BK\|_*E}{1-\|A+BK\|_*} \\ \delta \geq 0 & \quad\quad\quad\quad\quad\quad\quad\quad \|A+BK\|_* \leq 1. \end{array} \tag{7.23}$$

Now let $K^\star, \delta^\star$ be an optimal solution of (P1). By Eq. (7.23), we know that $K^{\circ\circ} = K^\star$ is feasible for Problem (P2) and leads to an objective value $\frac{d+\|BK^{\circ\circ}\|_*E}{1-\|A+BK^{\circ\circ}\|_*} \leq \delta^\star$. Similarly, let $K^{\star\star}$ be an optimal to Problem (P2), we know that $K^\circ = K^{\star\star}$ and $\delta^\circ = \frac{d+\|BK^\star\|_*E}{1-\|A+BK^\star\|_*}$ are feasible for (P1) and they lead to the same objective value. This hence proves the equivalence between the two problems.

Next, we show that $\frac{d+\|BK\|_*E}{1-\|A+BK\|_*}$ is quasi-convex in K when $\|A+BK\|_* \leq 1$, i.e., $S_s := \left\{K \;\middle|\; \frac{d+\|BK\|_*E}{1-\|A+BK\|_*} \leq s, \|A+BK\|_* \leq 1\right\}$ is a convex set for any s. Without loss of generality, we only need to consider $s \geq 0$ as otherwise $S_s = \emptyset$. In that case,

$$S_s = \left\{ K \;\middle|\; \begin{array}{l} \|BK\|_*E + s\|A+BK\|_* \leq s-d, \\ \\ \|A+BK\|_* \leq 1 \end{array} \right\}. \tag{7.24}$$

Since constants $s, E \geq 0$, and $\|BK\|_*$, $\|A+BK\|_*$ are convex functions in K, it follows that S_s is a convex set, and this finishes the proof. □

We highlight the following three points regarding the above optimization problem. First, a quasi-convex optimization problem can be solved by solving a sequence of convex feasibility problems. The idea is to do a line search on the objective value $f(x)$ and check if $S_s := \{x \text{ feasible} \mid f(x) = s\}$ is empty or not. Since S_s is a convex set by quasi-convexity of f, this can be done relatively efficiently. The detailed algorithm can be found in [16]. Secondly, for the optimization problem (P2) to be feasible, it

is necessary (but not sufficient) that pair (A, K) is stabilizable. To see this, consider the equivalent problem in (P1), in which we require $\|A + BK\|_* \leq \frac{\delta - d - \|BK\|_* E}{\delta} < 1$. If (A, B) is not controllable, this constraint can not be satisfied with any gain K. Finally, if the system in Eq. (7.14) has an output function and the estimated state $\hat{x}_t$ is given by an observer, a similar quasi-convex problem can be derived to minimize the estimation error bound E in Eq. (7.18).

Several remarks are provided below, regarding the issues when combining the path planning with the regulation.

(i) First, note that we need to design K^{h} for regulating the health dynamics and K^{f} for the faulty dynamics, which leads to Δ^{h} and Δ^{f} margin respectively. The state labeling in the MILP formulation is hence modified with Δ^{h} or Δ^{f} correspondingly.

(ii) The second remark is on splitting the control authority. Let K^{h} (K^{f}, respectively) be the solution of the problem in Eq. (P2) formulated with the healthy (faulty, respectively) dynamics. The control authority required by linear regulation is $U^{\mathrm{h}}_{\mathrm{reg}} = \{u \in \mathbb{R}^m : \|u\| \leq \|K^{\mathrm{h}}\|_* \Delta\}$. Therefore $U^{\mathrm{h}}_{\mathrm{plan}}$, the control authority reserved for path planning, need to be shrunk by $\|K^{\mathrm{h}}\|_* \Delta^{\mathrm{h}}$, i.e., $U^{\mathrm{h}}_{\mathrm{plan}} = U \ominus U^{\mathrm{h}}_{\mathrm{reg}}$. The procedure of splitting U for the faulty mode follows similarly.

(iii) The third remark is about determining the faulty nominal trajectory $\overline{\mathbf{x}}^{\mathrm{f}}$ used in the regulation. Recall that the regulator is in the form of $u_t = \overline{\mathbf{u}}^{\mathrm{f}}(t) + K^{\mathrm{f}}(\mathbf{x}(t) - \overline{\mathbf{x}}^{\mathrm{f}}(r))$ where $\overline{x}_t$ is define by $\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ under the faulty mode, but t_{o} is not known. However, we will only switch the regulator gain from K^{h} to K^{f} at time t_{d}, after which $\overline{\mathbf{x}}^{\mathrm{f}}[t_{\mathrm{o}}, t_{\mathrm{d}}]$ are the same for all t_{o}.

(iv) Finally, note that the true trajectory $\mathbf{x}$ may not be Δ^{h}-close to the healthy nominal trajectory $\overline{\mathbf{x}}^{\mathrm{h}}$ at detection time t_{d}, although this is true for all $t \leq t_{\mathrm{d}} - 1$.

Hence extra error is introduced in this last step and need to be added on top of Δ^{h}. This extra error is bounded by

$$\Delta^{\mathrm{h}} + \|A^{\mathrm{h}} - A^{\mathrm{f}}\|_*\|x\| + \|B^{\mathrm{h}} - B^{\mathrm{f}}\|_*\|u\| + \|F^{\mathrm{h}} - F^{\mathrm{f}}\|. \tag{7.25}$$

However, depending on the criticality of the application, this extra error can be neglected if the real-time detection has high enough sampling rate, and thus can report the fault as soon as the Δ margin is violated by a tiny amount.

We now summarize the above construction in Section IV-A and Section IV-B with the following proposition:

Propoition 16. Suppose that the healthy system (and the faulty system, respectively) are regulated by K^{h} (K^{f}, respectively) found by solving problem (P2), around the trajectory obtained by solving the MILP in Eq. (7.13), then the true trajectory robustly satisfies specification Φ in Eq. (4.8). This hence solves Problem 1.

Example 3. We present an example on robot path planning in this section. For simplicity, we assume state feedback in this example.

The considered system is modeled with a double integrator on the plane. The healthy discrete-time system matrices $A^{\mathrm{h}}, B^{\mathrm{h}}, F^{\mathrm{h}}$ in Eq. (7.2) are obtained by sampling the following continuous-time system with a sampling rate $\tau = 2$s.

$$A_{\mathrm{c}}^{\mathrm{h}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -20 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -20 \end{bmatrix}, B_{\mathrm{c}}^{\mathrm{h}} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, F_{\mathrm{c}}^{\mathrm{h}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{7.26}$$

For the faulty system dynamics, we assume that $A^{\mathrm{f}} = A^{\mathrm{h}}$, $B^{\mathrm{f}} = B^{\mathrm{h}}$, but there is a non-zero constant offset term $F^{\mathrm{h}} = [0,\ 1.5,\ 0,\ 0]^T$, resulting in an undesirable

drift. Let $x = [x_1, x_2, x_3, x_4]^T$ be the state and $u = [u_1, u_2]$ be the control input, we restrict that $x \in X = [-10, 10] \times [-2, 8] \times [-15, 15] \times [-15, 15]$, and that $u \in U = [-15, 15] \times [-15, 15]$. We also assume that there is an additive disturbance $w \in \mathbb{R}^4$ as in Eq. (7.14). In particular, disturbance w satisfy $\|w\| \leq 0.25$. We assume that the fault detection delay is bounded by 3 samples (i.e., $T = 3$).

The specification is defined by the LTL formula in Eq. (4.8) with

$$\varphi^{\mathrm{h}} = (\Box\neg\pi_{\mathrm{r}}) \wedge (\Diamond\Box\pi_{\mathrm{g}}) \wedge (\Box\Diamond\pi_{\mathrm{b1}}) \wedge (\Box\Diamond\pi_{\mathrm{b2}}), \tag{7.27}$$

$$\varphi^{\mathrm{f}} = (\Box\neg\pi_{\mathrm{r}}) \wedge (\Diamond\Box\pi_{\mathrm{g}}). \tag{7.28}$$

The regions (in x_1–x_3 space) in which each atomic proposition holds are marked in Fig. 7.3. In particular, the regions for π_{r} and π_{g} are the rectangles with solid boundaries, and the regions associated with π_{b1} (π_{b2}, respectively) are to the left (right, respectively) of the bold blue dashed line.

We first design a linear regulator by solving the quasi-convex optimization problem in (P2). The quasi-convex problem is solved with a standard line-search algorithm [16] that reduces to solving a sequence of convex optimization problems. These convex problems are then solved using CVX [43]. In this example, since $A^{\mathrm{f}} = A^{\mathrm{h}}$, $B^{\mathrm{f}} = B^{\mathrm{h}}$, we only need one regulator gain K, and the extra error introduced by detection delay in Eq. (7.25) can be bounded by $\|F^{\mathrm{h}} - F^{\mathrm{f}}\|$ (here we assume that the real-time detection has high enough sampling rate so that this extra error can be neglected). The obtained optimal regulator gain leads to a margin $\Delta^{\mathrm{h}} = \Delta^{\mathrm{f}} =: \Delta = 0.4604$. We hence modify the labeling by Δ. In Fig. 7.3, this modification corresponds to the transparent margin surrounding the rectangles and the thinner dashed lines close to the bold ones. Finally, we shrink the control set U by $\|K\|_* \Delta$, as discussed at the end of Section IV-B in remark (ii).

The fault tolerant path planning is then solved with the MILP formulated in Section IV-A. We solve the MILP with Gurobi [44]. Fig. 7.3 shows (i) the scenario when the system is always healthy, and (ii) a faulty scenario where the fault happens at time instant $t_o = 1$ and is detected at $t_d = 3$. The black dotted line represents the nominal healthy trajectory $\overline{\mathbf{x}}^h$ and the red dotted line represents the nominal faulty trajectory $\overline{\mathbf{x}}^f[t_o, t_d]$. The dark gray solid curve is the disturbed trajectory assuming that the system remains healthy forever, and the purple solid curve is the disturbed trajectory under the considered faulty scenario. The following observations can be made based on these simulations:

(i) It can be seen that both the disturbed trajectories satisfy the specification corresponding to their mode.

(ii) The two curves stay close until the fault detection, where the nominal trajectory starts to deviate from the healthy trajectory.

(iii) The healthy trajectory (gray) detours to the left more than it requires to satisfy φ^h, as it needs to preserve extra "room" for the faulty trajectory (purple) to avoid the red obstacle.

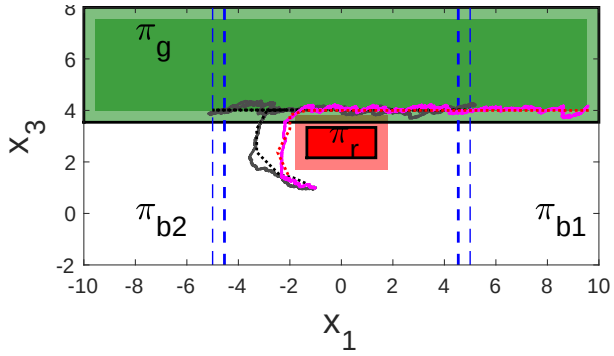

Figure 7.3: The planned trajectories (dotted) and the disturbed trajectories (solid) for health mode (black, gray) and faulty mode (red, purple).

CHAPTER VIII

Conclusion and Outlook

8.1 Conclusion

In this thesis, the problem of correct-by-construction fault-tolerant control was studied. As the first step towards correct-by-construction fault-tolerance, we studied model-based fault detection and the associated detectability analysis problems to provide a guarantee on finite fault detection. By combing LTL monitoring and model invalidation, an MILP-based detectability analysis was proposed to incorporate both the knowledge of the dynamical model and certain correct behavior the healthy system must fulfill. It was shown that the proposed detectability analysis method is less conservative.

We formalized the fault-tolerant control synthesis problem by defining a hierarchical system with fault configurations and capturing its graceful degradation requirement by two slightly different LTL formulas. The fault-tolerant control synthesis problem was then solved with two approaches, namely abstraction-based control synthesis via game solving and MILP-based path planning.

For abstraction-based synthesis, we approached the problem by first considering it on finite transition systems. A bottom up recursive algorithm was proposed to

decompose the overall synthesis problem into multiple smaller ones using the partial order induced by the fault configuration. This led to more scalable algorithms. A notable feature of the considered synthesis problem is that the worst case detection delay, if finite, is also incorporated in the synthesis. The problem with detection delay falls into the category of partial information game, and is hard to solve in general. However, we showed that, for a class of specifications with suffix-closeness and a novel property called absolute decomposability, the problem can be solved without constructing the belief space, and hence the exponential state space explosion can be avoided. A widely studied LTL fragment specified by GR(1) formula was shown to satisfy those structural properties.

To demonstrate that the methods developed for discrete systems can be used for continuous state systems as well, a fuel cell thermal management problem was solved via abstraction-based approach. To achieve compute a less conservative abstraction more efficiently, we i) studied the structural property of the fuel cell thermal dynamics and ii) extended the existing synthesis tool. In particular, a structural property called weak sign-stability was introduced and analyzed from the view of mixed monotonicity.

For the MILP-based path planning, we proposed a control hierarchy that combines high level path planning and low level regulation. The bounded error property of the low level regulation provided a way to handle detection delay in the fault tolerant path planning. Comparing to the regular MILP-based LTL path planning,The obtained control strategy was shown to be able to react to environment change and to be robust agianst disturbance.

8.2 Future Work

First, a potential extension of the fault-tolerant control synthesis problem definition itself is that we wish to specify graceful degradation with priority. That is, at a faulty mode, we desire to achieve the associated faulty specification only when achieving the healthy specification is not possible. Our conjecture is that LTL is not sufficient to express such priority and we plan to explore lattice-LTL [1] for the extension.

For the abstraction-based fault-tolerant control synthesis techniques presented in Chapter V, we wish to find a way to algorithmically verify if an LTL formula specifies an absolutely decomposable property, which is required to guarantee the proposed algorithm's soundness if achieving Φ is of our concern (see Section 4.2).

Regarding the abstraction-based synthesis for continuous state systems, one challenge is that the winning set for each faulty mode may be far from being maximal, due to the conservativeness of the abstraction. Sometimes in practice, with a small perturbation in some of the problem setup parameters (e.g., the bounds on disturbance), the winning set may change from the entire domain to empty set. This phenomenon may reduce the value of the proposed approach which requires the winning sets of different faulty modes to be nested according to the faults severity. To tackle this problem, one possible solution is to explore another type of abstraction known as the l-complete approximations [77], which are less conservative as it precisely encodes the long-term behavior of the underlying continuous-state system. Such abstractions may be obtained through long-term reachability analysis of the continuous-state system.

The work regarding fault-tolerant path planning currently presented in Chapter VII assumes a simplified fault configuration, which consists only one faulty mode other than the healthy mode. The extension to the system with any fault configura-

tions can be done naively by applying the same formulation techniques, but unlike the abstraction-based approach developed in Chapter 4 where the synthesis can be done in a decomposed manner, this leads to a MILP that can only be solved monolithically. Future work may address this issue, if possible. To extend the proposed approach to a broader class of systems other than linear ones, we plan to combine the MILP-based path planner with existing path planners that are more suitable for general nonlinear dynamics. While the MILP-based planner can work on a simplified linear model and focus on the high level logic constraints, the existing path planners will be used to handle the complicated dynamic constraints. Finally, to support the developed approach with some experiments, we plan to implement the MILP-based path planning on small drones.

Appendix A

Long Proofs

1.1 Proof of Theorem 3

Soundness

We prove the soundness of Algorithm 1 by a bottom up induction. To this end, define the level of a fault $\phi^{[i]}$ to be the longest distance from $\phi^{[i]}$ to a leaf mode of F. Clearly, the level of a fault $\phi^{[i]}$ is strictly larger than that of its successor $\phi^{[j]} \in \text{succ}(\phi^{[i]})$. Let k be the level of certain faulty modes in F, consider the following statement of k:

- Statement(k): Suppose at some time t, we have

  (A.1) $$\phi(t) = \phi^{[i]},$$

  (A.2) $$q(t) \in W^{[i]},$$

  (A.3) $$\phi(t+1) = \phi^{[j]} \in \text{succ}(\phi^{[i]}) \text{ where } \phi^{[j]} \text{ is of level } k,$$

  then Ψ is satisfied under strategy μ defined by Eq. (5.2).

If we can prove Statement(k) is true for all k, then the soundness of Algorithm 1 is immediately proved by picking k to be the level of mode $\phi^{[j]} \in \text{succ}(\phi^{[1]})$. This covers the case where the system will degrade from healthy mode at time $t+1$, and the soundness of Algorithm 1 under the always-healthy case is trivial. In what follows, we prove Statement(k) is true for all k by induction over k.

1° Base case: $k = 0$, i.e., $\phi^{[j]}$ is a leaf mode of F.

First, assuming Eq. (A.1)-(A.3), we have $q(t) \in W^{[i]}$, $\phi(t) = \phi^{[i]}$ and hence the dynamics is governed by $TS^{[i]}$. By definition of the overall strategy μ in Eq. (5.2), we know that sub-strategy $K^{[i]}$ is applied at time t, which guarantees $\psi^{[i]}$ (and hence $\Box W^{[j]}$ by line 8) under $TS^{[i]}$ dynamics (line 10). Hence $q(t+1) \in W^{[j]}$.

Moreover, Eq. (A.3) tells us that, at time $t+1$, the dynamics is governed by $TS^{[j]}$. Hence μ will start using sub-strategy $K^{[j]}\big(q(t+1)\big)$, which is well defined as $q(t+1) \in W^{[j]}$. Since $\phi^{[j]}$ is a leaf faulty mode, there will be no further degradation from there and thus

(i) $\phi(s) = \phi^{[j]},\ \forall s \geq t+1 \quad \Leftrightarrow \quad \pi^{[j]} \in L_Q^{\mathrm{F}}\big(q(s), \phi(s)\big),\ \forall s \geq t+1,$

(ii) $\phi(s) \neq \phi^{[j]},\ \forall s \leq t \quad \Leftrightarrow \quad \pi^{[j]} \notin L_Q^{\mathrm{F}}\big(q(s), \phi(s)\big),\ \forall s \leq t,$

(iii) $L_Q\big(q(t+1)\big)L_Q\big(q(t+2)\big)\ldots \models \varphi^{[j]} \quad \Rightarrow \quad L_Q^{\mathrm{F}}\big(q(t+1), \phi(t+1)\big)L_Q^{\mathrm{F}}\big(q(t+2), \phi(t+2)\big)\ldots \models \varphi^{[j]}.$

Combining bullets (i)-(iii) yields $L_Q^{\mathrm{F}}\big(q(1), \phi(1)\big)L_Q^{\mathrm{F}}\big(q(2), \phi(2)\big)\ldots \models \neg\pi^{[j]}\,\mathcal{U}\ \big(\Box\pi^{[j]} \wedge \varphi^{[j]}\big)$ and hence Ψ. That is, Statement(0) is proved.

2° Induction step: Assume that Statement(0), Statement(1), ..., Statement(k) are true, we prove Statement($k+1$) also holds in the sequel. There are two cases.

(i) Either $\phi(s) = \phi^{[j]}$ for all $s \geq t+1$ and the proof of Statement($k+1$) follows exactly the same as in the base case.

(ii) Or there exists $s > t+1$ such that $\phi(s) = \phi^{[\ell]} \in \mathrm{succ}(\phi^{[j]})$. In this case, the level of $\phi^{[\ell]} \in \mathrm{succ}(\phi^{[j]})$ is known to be strictly smaller than $k+1$ (the level of $\phi^{[j]}$). Then Statement(level of $\phi^{[\ell]}$), which holds by the hypothesis, verifies the satisfaction of Ψ.

Since specification Ψ is satisfied in both cases, the induction step is completed.

Completeness

Similar to the proof of the soundness, the completeness of Algorithm 1 can also be proved by a bottom up induction on k, the level of faulty modes in F. Consider the following statement of k:

- Statement(k): Suppose at some time t, we have

$$\phi(t) = \phi^{[i]}, \tag{A.4}$$

$$\phi(t+1) = \phi^{[j]} \in \mathrm{succ}(\phi^{[i]}) \text{ where } \phi^{[j]} \text{ is of level } k, \tag{A.5}$$

$$q(t+1) \notin W^{[j]}, \tag{A.6}$$

then there is an environment strategy that leads to the violation of Ψ.

Next we prove Statement(k) holds for all k by induction.

1° Base case: $k = 0$, i.e., $\phi^{[j]}$ is a leaf mode of F.

Since $\phi^{[j]}$ is a leaf, we know

(i) $\phi(s) = \phi^{[j]},\ \forall s \geq t+1 \quad \Leftrightarrow \quad \pi^{[j]} \in L_Q^{\mathrm{F}}\big(q(s), \phi(s)\big),\ \forall s \geq t+1,$

(ii) $\phi(s) \neq \phi^{[j]},\ \forall s \leq t \quad \Leftrightarrow \quad \pi^{[j]} \notin L_Q^{\mathrm{F}}\big(q(s), \phi(s)\big),\ \forall s \leq t,$

(iii) $\phi(s) \neq \phi^{[\ell]},\ \forall s \geq t+1 \quad \Leftrightarrow \quad \pi^{[\ell]} \notin L_Q^{\mathrm{F}}\big(q(s), \phi(s)\big),\ \forall s \geq t+1,\ \ell \neq j$
$\Rightarrow L_Q^{\mathrm{F}}\big(q(1), \phi(1)\big) L_Q^{\mathrm{F}}\big(q(2), \phi(2)\big) \cdots \not\models \neg\pi^{[\ell]}\, \mathcal{U}\ \big(\Box\pi^{[\ell]} \wedge \varphi^{[\ell]}\big),\ \forall \ell \neq j.$

Also, since it is assumed by Eq. (A.6) that $q(t+1) \notin W^{[j]}$, we know that

(iv) there exists an environment strategy under which any word $L_Q\big(q(t+1)\big) L_Q\big(q(t+2)\big) \cdots \not\models \varphi^{[j]} \quad \Rightarrow L_Q^{\mathrm{F}}\big(q(t+1), \phi(t+1)\big) L_Q^{\mathrm{F}}\big(q(t+2), \phi(t+2)\big) \cdots \not\models \varphi^{[j]}.$

Combining (i)(ii)(iv) yields $L_Q^{\mathrm{F}}\big(q(1), \phi(1)\big) L_Q^{\mathrm{F}}\big(q(2), \phi(2)\big) \cdots \not\models \neg\pi^{[j]}\, \mathcal{U}\ \big(\Box\pi^{[j]} \wedge \varphi^{[j]}\big)$, together with (iii) this establishes the violation of Ψ.

2° Induction step: Assume Statement(0), Statement(1), ..., Statement(k), we show that Statement($k+1$) also holds.

Assume the hypothesis of Statement($k+1$), we have $\phi(t+1) = \phi^{[j]}$ of level $k+1$ and $q(t+1) \notin W^{[j]}$. Consider the following environment strategy that determines the fault mode transition:

- If $\Box W^{[\ell]}$ is not yet violated for all $\phi^{[\ell]} \in \text{succ}(\phi^{[j]})$, the stay at the current mode $\phi^{[j]}$;
- otherwise $\Box W^{[\ell]}$ is violated for some $\phi^{[\ell]} \in \text{succ}(\phi^{[j]})$ at time s (i.e., $q(s) \notin W^{[\ell]}$), go to faulty mode $\phi^{[\ell]}$ at time s.

If the above environment strategy is adopted, we may have the following two possible cases:

(i) $\Box W^{[\ell]}$ is never violated for all $\phi^{[\ell]} \in \text{succ}(\phi^{[j]})$ and all time, but $\varphi^{[j]}$ cannot be achieved in the worst case. Suppose otherwise $\varphi^{[j]}$ is also satisfied, then $L_Q\big(q(t+1)\big)L_Q\big(q(t+2)\big)\ldots \models \psi^{[j]} = \varphi^{[j]} \wedge (\bigwedge_{\phi^{[\ell]} \in \text{succ}(\phi^{[j]})} \Box W^{[\ell]})$ and this implies $q(t+1)$ should be contained by the the maximal winning set $W^{[j]}$ synthesized against $\psi^{[j]}$ and dynamics $TS^{[j]}$, which contradicts with the assumption $q(t+1) \notin W^{[j]}$. Hence Ψ is violated in this case because the faulty mode stays at $\phi^{[j]}$ but $\varphi^{[j]}$ is violated.

(ii) $q(s) \notin W^{[\ell]}$ for some $\phi^{[\ell]} \in \text{succ}(\phi^{[j]})$ at time $s > t+1$. In this case, according to the faulty mode transition strategy described above, $\phi(s-1) = \phi^{[j]}$ and $\phi(s) = \phi^{[\ell]}$ with level $\leq k$. By Statement(level of $\phi^{[\ell]}$) we know that Ψ must be violated.

Since Ψ is violated in both cases if the above faulty mode selection strategy is adopted by the environment, Statement($k+1$) is verified, and this finishes the

induction step.

1.2 Proof of Theorem 4

Assume the actual transitions of faults are given by $(\phi^{[i_1]}, \phi^{[i_2]}), (\phi^{[i_2]}, \phi^{[i_3]}), \cdots, (\phi^{[i_{n-1}]}, \phi^{[i_n]})$, where $\phi^{[i_1]}$ is the initial fault and $\phi^{[i_n]}$ is the final fault, and $(\phi^{[i_{k-1}]}, \phi^{[i_k]}) \in \rightarrow_F$ are the nontrivial degradations.

Let $W^{[i_k]}$'s be the winning sets and $\psi^{[i_k]}$'s be the strengthened formulas returned in each round of recursion. Regarding these sets and formulas, we can make the following observations.

(a) $W^{[i_1]} \subseteq W^{[i_2]} \subseteq \cdots W^{[i_n]}$. By soundness of oracle **Win**, $W^{[i_{k-1}]}$ is the winning set w.r.t. specification $\psi^{[i_{k-1}]}$. But note that $\psi^{[i_{k-1}]}$ is a conjunction of $\Box W^{[i_k]}$ with other formulas (see line 9, Algorithm 2), thus $W^{[i_{k-1}]} \subseteq W^{[i_k]}$. This hence proves the nested relation of $W^{[i_k]}$'s because k is arbitrary in the above argument.

(b) $pref\big(Word(\psi^{[i_1]})\big) \subseteq pref\big(Word(\psi^{[i_2]})\big) \subseteq \cdots \subseteq pref\big(Word(\psi^{[i_n]})\big) \subseteq pref\big(Word(\varphi^{[i_n]})\big)$.

To see this, recall line 9 of Algorithm 2, we have

$$Word(\psi^{[i_{k-1}]}) = Word(\varphi^{[i_{k-1}]}) \cap Word(\Box W^{[i_k]}) \cap Word(\psi^{[i_k]}_{\text{safety}}),$$

where $Word(\varphi^{[i_{k-1}]})$ is absolutely decomposable by assumption, $Word(\Box W^{[i_k]})$ is an absolute safety property by Proposition 10, and $Word(\psi^{[i_k]}_{\text{safety}})$ is absolute safety property presuming that $\psi^{[i_k]}$ is absolutely decomposable. One can easily verify by induction that $Word(\psi^{[i_k]})$ is absolutely decomposable, using Proposition 10, 5, 8.

Next applying Proposition 6, this implies

$$pref\big(Word(\psi^{[i_k]})\big) = pref\big(Word(\psi^{[i_k]}_{\text{safety}})\big) \tag{A.7}$$

Also note that $\psi^{[i_{k-1}]}$ is obtained by conjunction of $\psi^{[i_k]}_{\text{safety}}$ and other formulas (line 9, Algorithm 2), hence

$$pref\big(Word(\psi^{[i_{k-1}]})\big) \subseteq pref\big(Word(\psi^{[i_k]}_{\text{safety}})\big). \tag{A.8}$$

Combining Eq. (A.7) (A.8), we have

$$pref\big(Word(\psi^{[i_{k-1}]})\big) \subseteq pref\big(Word(\psi^{[i_k]})\big). \tag{A.9}$$

With the same argument used to obtain Eq. (A.8),

$$pref\big(Word(\psi^{[i_n]})\big) \subseteq pref\big(Word(\varphi^{[i_n]})\big). \tag{A.10}$$

Now to prove the soundness, consider an execution starting from $q_0 \in W^{[i_1]}$ under control strategy μ constructed by Eq. (5.2) and arbitrary environment strategy η

$$\rho^{\mu\text{-}\eta}(q_0) = \big(q(0) = q_0, \phi(0), a(0)\big)\big(q(1), \phi(1), a(1)\big)\cdots, \tag{A.11}$$

and the word generated by this execution

$$\mathbb{w}_{\rho^{\mu\text{-}\eta}(q_0)} = w(0)w(1)w(2), \cdots. \tag{A.12}$$

First, let $t^{[i_k]}$ denote the time instant when fault transition $(\phi^{[i_k-1]}, \phi^{[i_k]})$ happens. By observation (a), it is not hard to show by induction that $q(t) \in W^{[i_k]}$ for $t \leq t^{[i_k]}$.

1° Base case: $k = 2$. The execution starts from $q(0) = q_0 \in W^{[i_1]}$, and the strategy enforces $\psi^{[i_1]}$. Hence $\Box W^{[i_2]}$, which is part of $\psi^{[i_1]}$ by construction, is true before the system degrades at time instant $t^{[i_2]}$.

2° As induction hypothesis, assume that for all $k \leq m$, we have $q(t) \in W^{[i_k]}$ for $t \leq t^{[i_k]}$ Now we move to $k = m+1$. First, by observation (a), $W^{[i_k]} \subseteq W^{[i_{m+1}]}$ for all $k \leq m$. The hypothesis immediately becomes $q(t) \in W^{[i_{m+1}]}$ for $t \leq t^{[i_m]}$, and what remains to be verify is when $t^{[i_m]} \leq t \leq t^{[i_{m+1}]}$. Again by hypothesis, we

know $q(t^{[i_m]}) \in W^{[i_{m+1}]}$. But by construction, strategy μ enforces the succeeding execution, which starts from $q(t^{[i_m]})$, to satisfy $\Box W^{[i_{m+1}]}$. In other words, for $t^{[i_m]} \leq t \leq t^{[i_{m+1}]}$, we have $q(t) \in W^{[i_{m+1}]}$. This hence finishes the induction step.

This immediately implies that $q(t^{[i_n]}) \in W^{[i_n]}$.

Next we show that the finite word generated until time $t^{[i_n]}$ belongs to $pref\big(Word(\varphi^{[i_n]})\big)$ using observation (b). Let $\mathbf{w}^{[i_k]} := w(t^{[i_k]}) \cdots w(t^{[i_{k+1}]} - 1)$ be the word segment that is generated under fault $\phi^{[i_k]}$. Note that this segment is generated starting from $q(t^{[i_k]}) \in W^{[i_k]}$ (by the result from the last paragraph), under the winning strategy designed to achieve $\psi^{[i_k]}$. Therefore $\mathbf{w}^{[i_k]} \in pref\big(Word(\psi^{[i_k]})\big)$. By observation (b), this means $\mathbf{w} := \mathbf{w}^{[i_1]}\mathbf{w}^{[i_2]} \cdots \mathbf{w}^{[i_n]} \in pref\big(Word(\varphi^{[i_n]})\big)$.

To this point, we have shown that when the final fault occurs at time $t^{[i_n]}$, the state $q(t^{[i_n]})$ is in the winning set $W^{[i_n]}$ for this final fault. We also know that finite word $\mathbf{w} = w(0) \cdots w(t^{[i_n]} - 1)$ generated so far belongs to $pref\big(Word(\varphi^{[i_n]})\big)$. Note that the succeeding strategy will focus on achieving $\varphi^{[i_n]}$ starting from state $q(t^{[i_n]})$, where the strategy is well defined because $q(t^{[i_n]}) \in W^{[i_n]}$. Moreover, this strategy generates an execution $\mathbb{v} = v(t^{[i_n]})v(t^{[i_n]} + 1) \cdots \in Word(\varphi^{[i_n]})$. Recall that $\varphi^{[i_n]}$ is absolutely decomposable. By Proposition 3, the overall word $\mathbf{w}\mathbb{v} \models \varphi^{[i_n]}$. This proves the soundness of Algorithm 2 under the given assumptions.

1.3 Proof of Theorem 5

Soundness

The soundness of Algorithm 3 can be proved using a bottom up induction exactly the same as that in Appendix 1.1. Consider the following statement of faulty mode level k.

- Statement(k): Suppose at some time t, we have

  (A.13) $$\phi(t) = \phi^{[i]},$$

  (A.14) $$q(t) \in W^{[i]},$$

  (A.15) $$\phi(t+1) = \phi^{[j]} \in \mathrm{succ}(\phi^{[i]}) \text{ where } \phi^{[j]} \text{ is of level } k,$$

  then Ψ is satisfied under strategy $\widehat{\mu}$ defined by Eq. (5.6).

1° Base case: $k = 0$, i.e., $\phi^{[j]}$ is a leaf mode of F.

First, by Assumption 2, no fault is allowed to happen within the uninformed horizon. Hence $\phi(t+1) = \phi^{[j]} \neq \phi^{[i]} = \phi(t)$ implies that

(A.16) $$\widehat{\phi}(t+1) = \phi^{[i]}.$$

Moreover, since the detection delay is assumed to be finite, there is $r > t+1$ such that

(A.17) $$\widehat{\phi}(s) = \phi^{[i]} \text{ and } \phi(s) = \phi^{[j]},\ \forall s \in [\![t+1, r]\!],$$

(A.18) $$\widehat{\phi}(r+1) = \phi^{[j]}.$$

That is, $r+1$ is the time instant fault $\phi^{[j]}$ is detected.

By Eq. (A.16) and the definition of $\widehat{\mu}$ (see Eq. (5.2)), we know that sub-strategy $K^{[i]}$ is used with in time interval $[\![t+1, r]\!]$, Since $K^{[i]}$ is synthesized against $TS^{[i]} \odot I$, where the actions in I guarantee invariance in $C^{[j]}$ when the dynamics is governed by $TS^{[j]}$, hence we have

(A.19) $$q(s) \in C^{[j]} \subseteq W^{[j]},\ \forall s \in [\![t+1, r+1]\!],$$

By Eq. (A.18) and the definition of $\widehat{\mu}$, sub-strategy $K^{[j]}$ is used starting from time $r+1$, which is valid because $q(r+1) \in W^{[j]}$ by Eq. (A.19). Since $\phi^{[j]}$ is a

leaf mode, the suffix word starting from time $r+1$ satisfied $\varphi^{[j]}$, i.e.,

$$L_Q^{[j]}\big(q(r+1)\big)L_Q^{[j]}\big(q(r+2)\big)\ldots \models \varphi^{[j]} \tag{A.20}$$

which implies that

(i) $L_Q^{[j]}\big(q(r+1)\big)L_Q^{[j]}\big(q(r+2)\big)\ldots \models \varphi^{[j]}_{\text{liveness}}$

$\Rightarrow \quad L_Q^{[j]}\big(q(t+1)\big)L_Q^{[j]}\big(q(t+2)\big)\ldots \models \varphi^{[j]}_{\text{liveness}}$ because $\varphi^{[j]}_{\text{liveness}}$ specifies an absolute liveness property.

(ii) $L_Q^{[j]}\big(q(r+1)\big)L_Q^{[j]}\big(q(r+2)\big)\ldots \models \varphi^{[j]}_{\text{safety}}$.

Note that $\varphi^{[j]}$ is both absolutely decomposable and suffix-closed, hence $\varphi^{[i]}_{\text{safety}}$ is invariance by Proposition 11. This means $\varphi^{[j]}_{\text{safety}} = \Box\psi$ for some propositional formula ψ, and $L_Q^{[j]}(q) \models \psi$ for any state $q \in W^{[j]}$ as $W^{[j]}$ is the winning set synthesized against $\varphi^{[j]} = \varphi^{[j]}_{\text{safety}} \wedge \varphi^{[j]}_{\text{liveness}} = (\Box\psi) \wedge \varphi^{[j]}_{\text{liveness}}$. Hence Eq. (A.19) implies that

(iii) $L_Q^{[j]}\big(q(t+1)\big)\ldots L_Q^{[j]}\big(q(r)\big) \in pref\big(Word(\varphi^{[j]}_{\text{safety}})\big)$.

Also since $\varphi^{[j]}_{\text{safety}}$ is absolute safety, bullets (ii) (iii) implies that

(iv) $L_Q^{[j]}\big(q(t+1)\big)L_Q^{[j]}\big(q(t+2)\big)\ldots \models \varphi^{[j]}_{\text{safety}}$.

Combining bullets (i) and (iv) hence yields $L_Q^{[j]}\big(q(t+1)\big)L_Q^{[j]}\big(q(t+2)\big)\ldots \models \varphi^{[j]}$. But also note that $\phi(t)$ changes from $\phi^{[i]}$ to $\phi^{[j]}$ at time $t+1$ and will stay at leaf mode $\phi^{[j]}$ ever since, therefore we have $L_Q^{\mathrm{F}}\big(q(t+1),\phi(t+1)\big)L_Q^{\mathrm{F}}\big(q(t+2),\phi(t+2)\big)\ldots \models \neg\pi^{[j]}\,\mathcal{U}\,\big(\Box\pi^{[j]} \wedge \varphi^{[j]}\big)$. Hence Ψ is achieved and Statement(0) is proved.

2° Induction step: assume that Statement(0), Statement(1), …, Statement(k) are true, we prove Statement($k+1$) also holds in the sequel. There are two cases.

(i) Either $\phi(s) = \phi^{[j]}$ for all $s \geq t+1$ and the proof of Statement(k) fol-

lows almost the same as in the base case, except that $\varphi^{[j]}$, $\varphi^{[j]}_{\text{safety}}$, $\varphi^{[j]}_{\text{liveness}}$ should be replaced by $\psi^{[j]}$, $\psi^{[j]}_{\text{safety}}$, $\psi^{[j]}_{\text{liveness}}$ respectively. Since $\psi^{[j]}_{\text{safety}} = \varphi^{[j]}_{\text{safety}} \wedge (\bigwedge_{j:\phi^{[j]} \in \text{succ}(\phi^{[i]})})(\Box W^{[j]})$ is still an invariance property and $\varphi^{[j]}_{\text{liveness}} = \psi^{[j]}_{\text{liveness}}$, we know that $\psi^{[j]}$ is also both absolute decomposable and suffix-closed, and the satisfaction of Ψ can be established in the same way.

(ii) Or there exists $s > t+1$ such that $\phi(s) = \phi^{[\ell]} \in \text{succ}(\phi^{[j]})$, in which case Ψ must be also satisfied by Statement(level of $\phi^{[\ell]}$), which holds because the level of $\phi^{[\ell]} \in \text{succ}(\phi^{[j]})$ is known to be strictly smaller than $k+1$ (i.e., the level of $\phi^{[j]}$).

Since specification Ψ is satisfied in both cases, the induction step is completed.

With Statement(k) proved for all k, we immediately obtain the soundness of Algorithm 3. This is because either the faulty mode always stays at healthy and $\psi^{[1]}$ (hence $\varphi^{[1]}$ and Ψ) is satisfied, or a degradation occurs at time $t+1$, in which case Ψ is satisfied by Statement(level of $\phi^{[1]}$).

Completeness

To establish the completeness of Algorithm 3, we need the following Lemmas.

Lemma 4. In Algorithm 3, $C^{[j]} = W^{[j]}$ if $\varphi^{[j]}$ is suffix-closed.

Proof. By construction, we have $C^{[j]} \subseteq W^{[j]}$. Whenever $\varphi^{[j]}$ is suffix-closed, we also have $W^{[j]} \subseteq C^{[j]}$ because 1) $C^{[j]}$ is the largest controlled invariant set contained by $W^{[j]}$ under $TS^{[j]}$ and 2) $W^{[j]}$ is invariant (i.e., $q(t) \in W^{[j]}$ for all t) under strategy $K^{[j]}$ whenever the dynamics is governed by $TS^{[j]}$.

To prove that $q(t) \in W^{[j]}$ for all t under dynamics $TS^{[j]}$ and strategy $K^{[j]}$, assume for a contradiction that $q(1) \in W^{[j]}$ but $q(t) \notin W^{[j]}$ for some t under strategy $K^{[j]}$, and let $\mathbb{w} := w(1)w(2)\ldots$, where $w(s) = L_Q^{[j]}\big(q(s)\big)$ for all s, be a generated

word. We have $\mathbb{w} \models \psi^{[j]}$ by construction. Since $\varphi^{[j]}$ is suffix-closed, so is $\psi^{[j]} = \varphi^{[j]} \wedge (\bigwedge_{\ell:\phi^{[\ell]} \in \text{succ}(\phi^{[j]})} \square C^{[\ell]})$. Therefore $\mathbb{w}_t = w(t)w(t+1)\ldots \models \psi^{[j]}$. But this means it is possible to achieve $\psi^{[j]}$ starting from state $q(t)$ under some strategy, which contradicts with $W^{[j]}$ being maximal. □

Lemma 5. Let $W(\Psi, TS^{\text{F}}, T)$ be the maximal winning set of Ψ on faulty system TS^{F} under Assumption 2, and let $W(\Psi, TS^{\text{F}}, T_1)$ be the winning set under the same settings and an extra assumption that detection delay $T^{[i]} = 1$ for all fault $\phi^{[i]}$. We have $W(\Psi, TS^{\text{F}}, T) \subseteq W(\Psi, TS^{\text{F}}, T_1)$.

Proof. This should be clear because when $T^{[i]} = 1$, the environment has more restriction in terms of hiding the true faulty mode from the controller, comparing to the case where $T^{[1]} \geq 1$. Formally, this means $\mathfrak{V}_{\text{delay}}(F, T_1) \subseteq \mathfrak{V}_{\text{delay}}(F, T)$. Let $q(1) \in W(\Psi, TS^{\text{F}}, T)$ and let μ be the associated winning strategy. By Definition 13, this means

$$\forall(\mathbb{f}, \widehat{\mathbb{f}}) \in \mathfrak{V}_{\text{delay}}(F, T_1) \subseteq \mathfrak{V}_{\text{delay}}(F, T) : \mathbb{w} \in \mathfrak{W}\big(\mathbb{f}, \widehat{\mathbb{f}}, \mu, q(1)\big) \models \Psi, \tag{A.21}$$

that is, $q(1) \in W(\Psi, TS^{\text{F}}, T_1)$. □

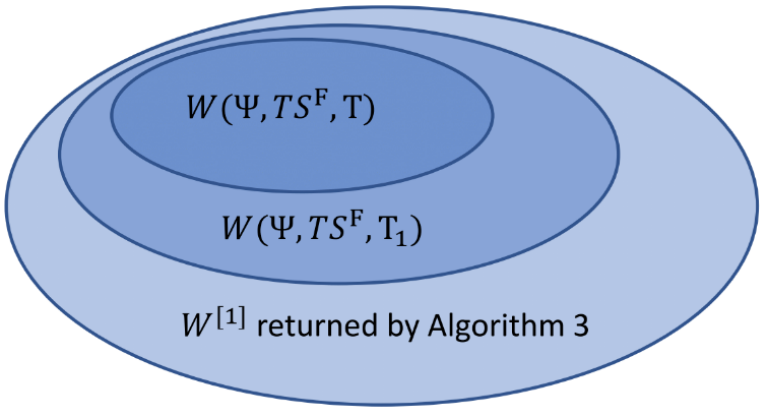

Figure A.1: Illustration of the completeness proof of Algorithm 3.

We now prove the completeness of Algorithm 3 under the assumption that $\varphi^{[j]}$ is both absolutely decomposable and suffix-closed. To this end, we will show by a bottom up induction that $W(\Psi, TS^{\mathrm{F}}, T_1) \subseteq W^{[1]}$, where $W^{[1]}$ is the winning set returned by Algorithm 3. Then we have $W(\Psi, TS^{\mathrm{F}}, T) \subseteq W(\Psi, TS^{\mathrm{F}}, T_1) \subseteq W^{[1]}$ by Lemma 5, and this verifies the completeness.

To prove the above statement, let $q(1) \in W(\Psi, TS^{\mathrm{F}}, T_1)$, and let μ_1 be the associated winning strategy. Note that μ_1 is not necessarily the same as $\widehat{\mu}$ defined in Eq. (5.6). Consider the following statement of faulty mode level k:

- Statement(k): Let k be the level of faulty mode $\phi^{[i]}$ and t be any time instant, under strategy μ_1, we have $\phi(t) = \phi^{[i]} \Rightarrow q(t) \in W^{[i]}$, where $q(t)$ is any possible state at time t of system TS^{F} initiated from $q(1) \in W(\Psi, TS^{\mathrm{F}}, T_1)$, and $W^{[i]}$ is defined in Algorithm 3.

Clearly, if Statement(k) can be verified for all k, then applying Statement(level of $\phi^{[1]}$) at time $t = 1$ immediately yields $q(1) \in W^{[1]}$ where $W^{[1]}$ is returned by Algorithm 3, and this proves the completeness as argued above.

In the sequel we show that Statement(k) is true for all k by induction.

1° Base case, $k = 0$, i.e., $\phi^{[i]}$ is a leaf mode of F.

First, since $\phi^{[i]}$ is already a leaf mode, this means $\mathbb{f} = \phi(1)\phi(2)\cdots\left(\phi(r) = \phi^{[i]}\right)^{\omega}$ for some $r \leq t$. Also since μ_1 achieves Ψ, this implies that the word $\mathbb{w}_r := w(r)w(r+1)\cdots$ generated starting from time r must satisfy $\varphi^{[i]}$. Note that $\varphi^{[i]}$ is assumed to be suffix-closed, applying Lemma 4 hence gives $q(s) \in W^{[i]}$ for all $s \geq r$. Therefore $q(t) \in W^{[i]}$ as $t \geq r$.

2° Induction step: assume that Statement(0), Statement(1), ..., Statement(k) are true, we will prove Statement($k+1$) also holds, i.e. $q(t) \in W^{[i]}$ if $\phi(t) = \phi^{[i]}$. To this end, first define

$\mathbf{q}_t = q(1)q(2)\cdots q(t)$,

$\widehat{\mathbf{f}}_t = \widehat{\phi}(1)\widehat{\phi}(2)\cdots\widehat{\phi}(t)$,

$\mathbf{f}_t = \phi(1)\phi(2)\cdots\phi(t)$.

Since μ_1 is the winning strategy assoicated with $W(\Psi, TS^{\mathrm{F}}, T_1)$, we know the following facts under $\phi(t) = \phi^{[i]}$:

(i) $q(t) \in \cap_{j:\phi^{[j]}\in \mathrm{succ}(\phi^{[i]})} W^{[j]}$.

To prove bullet (i), assume for a contradiction that $q(t) \notin W^{[j]}$ for some $j : \phi^{[j]} \in \mathrm{succ}(\phi^{[i]})$. Then we can construct $\mathbf{g}_t = \phi(1)\phi(2)\cdots\phi(t-1)\phi^{[j]}$, which is exactly the same as $\mathbf{f}_t$ except the last element is replaced by $\phi^{[j]}$. Note that $\mathbf{g}_t$ has exactly the same estimation sequence $\widehat{\mathbf{g}}_t = \widehat{\mathbf{f}}_t$ due to the one step detection delay, hence the decision of μ is not affected. But in that case, we have $\phi(t) = \phi^{[j]}$ but $q(t) \notin W^{[j]}$, which contradicts with Statement(ℓ) where $\ell \leq k$ is the level of mode $\phi^{[j]}$.

(ii) $\forall a(t) \in \mu_1(\mathbf{q}_t, \widehat{\mathbf{f}}_t), \forall \phi^{[j]} \in \mathrm{succ}(\phi^{[i]}), \forall\big(q(t), \phi^{[j]}, a(t), q(t+1)\big) \in \rightarrow_{TS}$: $q(t+1) \in W^{[j]}$.

To prove bullet (ii), assume for a contradiction that, funder some $\phi^{[j]} \in \mathrm{succ}(\phi^{[i]})$ and the associated dynamics $TS^{[j]}$, we have $q(t+1) \notin W^{[j]}$ under some $a(t) \in \mu_1(\mathbf{q}_t, \widehat{\mathbf{f}}_t)$. The the environment can pick $\phi(t+1) = \phi^{[j]}$ and this contradicts with Statement(level of $\phi^{[j]}$, which is $\leq k$) that is already established.

(iii) $\mu_1(\mathbf{q}_t, \widehat{\mathbf{f}}_t) \subseteq \cap_{j:\phi^{[j]}\in\mathrm{succ}(\phi^{[i]})} I^{[j]}\big(q(t)\big)$.

Note that by Lemma 4, $W^{[j]}$ is invariant under $TS^{[j]}$, whereas $I^{[j]}(q)$ is defined to be the set of all actions that assures the invariance of $W^{[j]}$. By bullet (i) and (ii), we know that $q(t) \in W^{[j]}$ and $q(t+1) \in W^{[j]}$ under any $a(t) \in \mu_1(\mathbf{q}_t, \widehat{\mathbf{f}}_t)$, hence $\mu_1(\mathbf{q}_t, \widehat{\mathbf{f}}_t) \subseteq I^{[j]}\big(q(t)\big)$ for all $j : \phi^{[j]} \in \mathrm{succ}(\phi^{[i]})$ and

this proves bullet (iii).

(iv) $\forall a(t) \in \mu_1(\mathbf{q}_t, \widehat{\mathbf{f}}_t), \forall \phi^{[j]} \in \text{succ}(\phi^{[i]}), \forall \big(q(t), \phi(t) = \phi^{[i]}, a(t), q(t+1)\big) \in \rightarrow_{TS}$: $q(t+1) \in W^{[j]}$.

The difference from bullet (ii) is that the evolution is under $TS^{[i]}$ rather than $TS^{[j]}$. To prove bullet (iv), assume otherwise $q(t+1) \notin W^{[j]}$ for some $a(t), \phi^{[j]}$ and transition under $TS^{[i]}$, then the environment can pick $\phi(t+1) = \phi^{[j]}$ and this contradicts with Statement(level of $\phi^{[j]}$, which is $\leq k$) that is already established.

(v) Suppose $\phi(t) = \phi^{[i]}$, then $q(t+1) \in \cap_{j:\phi^{[j]} \in \text{succ}(\phi^{[i]})} W^{[j]}$.

Applying bullet (iv) gives the desired statement.

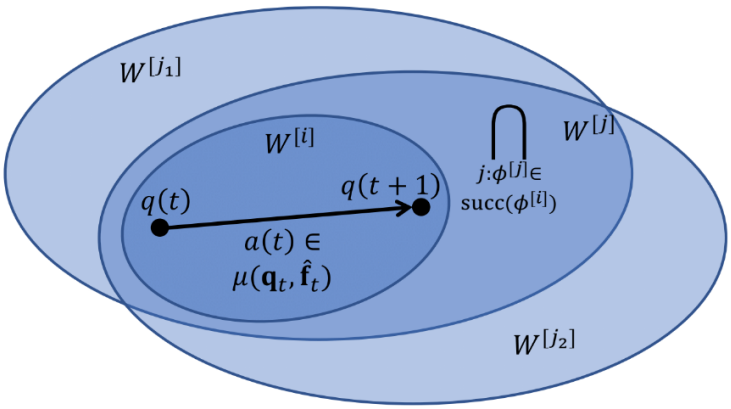

Figure A.2: Illustration of facts (i)-(v), induction step in the completeness proof of Algorithm 3

Let us consider the system's state sequence $\mathbb{p} = \mathbf{q}_t \mathbb{p}_{t+1}$, generated under μ_1 and a specific faulty sequence $\mathbf{f}_t(\phi^{[i]})^\omega$. Define

$$s = \min_{\tau:\phi(\tau)=\phi^{[i]}} \tau \tag{A.22}$$

Note that $\phi(t) = \phi^{[i]}$ by the hypothesis of Statement$(k+1)$, we have $s \leq t$. Also note that, since $\phi(s) = \phi^{[i]}$, bullets (i)-(v) developed above hold for $q(s)$ as well. Consider $\mathbb{p}_s = q(s)q(s+1)\cdots q(t)p(t+1)p(t+2)\cdots$. Since μ is the winning strategy that achieves Ψ while the faulty mode stays at $\phi^{[i]}$ eventually, the word

generated by $\mathbb{p}_s$ satisfies $\varphi^{[i]}$. Moreover, by applying bullets (i) and (v) inductively on time, we can establish that $p(r) \in \cap_{j:\phi^{[j]} \in \mathrm{succ}(\phi^{[i]})} W^{[j]}$ for all $r \geq s$. Hence the word generated by $\mathbb{p}_s$ models $\psi^{[i]} := \varphi^{[i]} \wedge (\bigwedge_{j:\phi^{[j]} \in \mathrm{succ}(\phi^{[i]})} \Box C^{[j]})$ because $C^{[j]} = W^{[j]}$ by Lemma 4. Also note that this is achieved under dynamics $TS^{[i]}$ with actions restricted in $I := \cap_{j:\phi^{[j]} \in \mathrm{succ}(\phi^{[i]})} I^{[j]}\big(q(t)\big)$ by bullet (ii). Hence we have $q(s) \in \mathbf{Win}(\psi^{[i]}, TS^{[i]} \odot I) =: W^{[i]}$. Finally, by Lemma 4, the suffix-closedness of $\psi^{[i]}$ suggests that a winning execution should never leave the winning set $W^{[i]}$, hence $q(t) \in W^{[i]}$.

Hence the induction step is completed.

Appendix B

Variables and Constants in the Fuel Cell Thermal Model

Control u

u_{HR}	$u_{\mathrm{HR}} = 1$	indicating that the coolant flow goes through the heater
	$u_{\mathrm{HR}} = 0$	indicating that the coolant flow goes through the radiator
i	[0,1.5]	(A cm^{-2}) Cell current density
P_{H}	[0, 35000]	(W) Power requested by heater
w_{cool}	[0,800]	(g s^{-1}) Coolant mass flow rate

State x

SOC_{B}	[0,1]	(-) Battery energy
T_1	[273, 360]	(K) Temperature of first control volumes
T_2	[273, 360]	(K) Temperature of second control volumes
T_{H}	[250, 400]	(K) Heater temperature
T_{R}	[250, 340]	(K) Radiator temperature

Operating Condition d

P_{M}	$[2, 17]$	(kW) Power requested by motor
$p_{\mathrm{O_2}}$	5×10^4	(Pa) Oxygen partial pressure
$p_{\mathrm{H_2}}$	1.5×10^5	(Pa) Hydrogen partial pressure
r_{v}	$[0, 10^{-7}]$	(mol cm^{-3} s^{-1}) Volumetric evaporating rate
v	[10,20]	(ms^{-1}) Vehicle speed
T_{amb}	[273,290]	(K) Ambient temperature
λ	[4,22]	(-) Membrane water content

Other Variables

$E_{\text{FC,stack}}$	(V) Fuel cell stack electrical potential
i_0	(A cm^{-2}) Exchange current density
$P_{\text{B,output}}$	(W) Battery output power
$P_{\text{FC,output}}$	(W) Fuel cell output power
$P_{\text{FC,self-heat-up}}$	(W) Power for fuel cell self-heat-up
R_Ω	(Ω cm^2) Cell resistivity
T_{avg}	(J mol^{-1} K^{-1}) Average fuel cell temperature
$T_{\text{FC,in,cool}}$	(K) Inlet coolant temperature (into fuel cell)
$T_{\text{FC,out,cool}}$	(K) Outlet coolant temperature (from fuel cell)
Δh_{rxn}	(J mol^{-1}) Reaction enthalpy
Δh_{v}	(J mol^{-1}) Evaporation enthalpy
Δs_{rxn}	(J mol^{-1} K^{-1}) Reaction entropy

Constants

c_{air}	(1.0 J g^{-1}K^{-1}) Air specific heat capacity
F	(96485 C mol^{-1}) Faraday constant
P_{ref}	(101325 Pa) Reference pressure
R	(8.314 J mol^{-1} K^{-1}) Universal gas constant

Parameters

a_{MT}	(V) Mass transfer potential loss coefficient
A_{FC}	(cm^2) Fuel cell cross section area
A_{G}	(cm^2) Fuel cell geometric area
b_{MT}	(-) Mass transfer potential loss exponent
c_{cool}	(J g^{-1}K^{-1}) Coolant specific heat capacity
c_{FC}	(J g^{-1}K^{-1}) Fuel cell specific heat capacity
C_{H}	(J K^{-1}) Heater heat capacity
C_{R}	(J K^{-1}) Radiator heat capacity
$E_{\text{B,cell}}$	(V) Battery cell open-circuit potential
$G_{\text{B,stack,total}}$	(Ws) Battery stack energy capacity
$i_{0,\text{ref}}$	(A cm^{-2}) Reference exchange current density
i_{MT}	(A cm^{-2}) Mass transfer current density
i_{x}	(A cm^{-1}) Crossover current density
$k_{\text{amb}\to\text{FC}}$	(W cm^{-3} K^{-1}) Heat transfer coefficient: ambient to stack
$k_{\text{amb}\to\text{H}}$	(W cm^{-3} K^{-1}) Heat transfer coefficient: ambient to heater
n_{p}	(-) Number of battery cells in parallel
n_{s}	(-) Number of battery cells in series
$n_{\text{FC,cell}}$	(-) Number of fuel cells in stack
$r_{\text{B,cell}}$	(Ω) Battery cell internal resistance
V_{FC}	(cm^3) Fuel cell volume
α	(-) Charge transfer coefficient
δ_{FC}	(cm) Channel or cell length
κ_T	(W cm^{-1} K^{-1}) thermal conductivity
ρ_{FC}	(g cm^{-3}) Fuel cell density

BIBLIOGRAPHY

[1] Shaull Almagor and Orna Kupferman. Latticed-ltl synthesis in the presence of noisy inputs. *Discrete Event Dynamic Systems*, 27(3):547–572, 2017.

[2] Bowen Alpern and Fred B Schneider. Defining liveness. *Information processing letters*, 21(4):181–185, 1985.

[3] Bowen Alpern and Fred B Schneider. Recognizing safety and liveness. *Distributed computing*, 2(3):117–126, 1987.

[4] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 6271–6278. IEEE, 2014.

[5] Sohel Anwar. *Fault tolerant drive by wire systems: Impact on vehicle safety and reliability*. Bentham Science Publishers, 2012.

[6] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *International Conference on Runtime Verification*, pages 147–160. Springer, 2011.

[7] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[8] Ayca Balkan, Moshe Vardi, and Paulo Tabuada. Controller Synthesis for Mode-Target Games. In *Proc. of IFAC ADHS*, 2015.

[9] Ayca Balkan, Moshe Vardi, and Paulo Tabuada. Mode-target games: Reactive synthesis for control applications. *IEEE Transactions on Automatic Control*, 63(1):196–202, 2017.

[10] Andreas Bauer, Martin Leucker, and Christian Schallhart. Monitoring of real-time properties. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 260–272. Springer, 2006.

[11] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017.

[12] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[13] Franco Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced lyapunov functions. *IEEE Transactions on automatic control*, 39(2):428–433, 1994.

[14] Mogens Blanke, Marcel Staroswiecki, and N Eva Wu. Concepts and methods in fault-tolerant control. In *American Control Conference, 2001. Proceedings of the 2001*, volume 4, pages 2606–2620. IEEE, 2001.

[15] Roderick Bloema, Barbara Jobstmannb, Nir Pitermanc, Amir Pnueli, and Yaniv Sa'ar. Synthesis of reactive (1) designs. *J. Comput. System Sci.*, 78:911–938, 2012.

[16] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[17] Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner. *Model-based testing of reactive systems - Advanced lectures, LNCS 3472*. Springer-Verlag, Berlin, 2005.

[18] Janusz Brzozowski, Galina Jirásková, and Chenglong Zou. Quotient complexity of closed languages. *Theory of Computing Systems*, 54(2):277–292, 2014.

[19] Joel W Burdick, Noel du Toit, Andrew Howard, Christian Looman, Jeremy Ma, Richard M Murray, and Tichakorn Wongpiromsarn. Sensing, navigation and reasoning technologies for the darpa urban challenge. Technical report, California Inst. of Technology, Pasadena, Jet Propulsion Laboratory, 2007.

[20] Christos G Cassandras and Stephane Lafortune. *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[21] Phillip R Chandler. Self-repairing flight control system reliability and maintainability program executive overview. In *Proc. Nat. Aero. & Electr. Conf*, pages 586–590, 1984.

[22] Krishnendu Chatterjee, Laurent Doyen, Thomas A Henzinger, and Jean-François Raskin. Algorithms for omega-regular games with imperfect information. In *International Workshop on Computer Science Logic*, pages 287–302. Springer, 2006.

[23] Yuxiao Chen, James Anderson, Karan Kalsi, Steven H Low, and Aaron D Ames. Compositional set invariance in network systems with assume-guarantee contracts. In *2019 American Control Conference (ACC)*, pages 1027–1034. IEEE, 2019.

[24] Glen Chou, Yunus Emre Sahin, Liren Yang, Kwesi J Rutledge, Petter Nilsson, and Necmiye Ozay. Using control synthesis to generate corner cases: A case study on autonomous driving. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2906–2917, 2018.

[25] Tianguang Chu and Lin Huang. Mixed monotone decomposition of dynamical systems with application. *Chinese science bulletin*, 43(14):1171–1175, 1998.

[26] Samuel Coogan and Murat Arcak. Efficient finite abstraction of mixed monotone systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 58–67. ACM, 2015.

[27] Samuel Coogan, Murat Arcak, and Calin Belta. Formal methods for control of traffic flow: Automated control synthesis from finite-state transition models. *IEEE Control Systems Magazine*, 37(2):109–128, 2017.

[28] Samuel Coogan, Murat Arcak, and Alexander A Kurzhanskiy. Mixed monotonicity of partial first-in-first-out traffic flow models. In *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pages 7611–7616. IEEE, 2016.

[29] Eric Dallal, Alessandro Colombo, Domitilla Del Vecchio, and Stéphane Lafortune. Supervisory control for collision avoidance in vehicular networks with imperfect measurements. In *Proc. of IEEE CDC*, pages 6298–6303, 2013.

[30] Marcelo d'Amorim and Grigore Roşu. Efficient monitoring of ω-languages. In *International Conference on Computer Aided Verification*, pages 364–378. Springer, 2005.

[31] Volker Diekert and Paul Gastin. First-order definable languages. *Logic and automata*, 2:261–306, 2008.

[32] Xu Chu Ding, Calin Belta, and Christos G Cassandras. Receding horizon surveillance with temporal logic specifications. In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 256–261. IEEE, 2010.

[33] Maxence Dutreix and Samuel Coogan. Specification-guided verification and abstraction refinement of mixed-monotone stochastic systems. *arXiv preprint arXiv:1903.02191*, 2019.

[34] Alina Eqtami and Antoine Girard. A quantitative approach on assume-guarantee contracts for safety of interconnected systems. In *European Control Conference*, 2019.

[35] John S Eterno, Jerold L Weiss, Douglas P Looze, and Alan Willsky. Design issues for fault tolerant-restructurable aircraft control. In *Decision and Control, 1985 24th IEEE Conference on*, volume 24, pages 900–905. IEEE, 1985.

[36] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

[37] Kevin Forsberg and Harold Mooz. The relationship of system engineering to the project cycle. In *INCOSE International Symposium*, volume 1, pages 57–65. Wiley Online Library, 1991.

[38] Paul M Frank and Xianchun Ding. Survey of robust residual generation and evaluation methods in observer-based fault detection systems. *Journal of process control*, 7(6):403–424, 1997.

[39] Damian Frick, Tony A Wood, Gian Ulli, and Maryam Kamgarpour. Robust control policies given formal specifications in uncertain environments. *IEEE control systems letters*, 1(1):20–25, 2017.

[40] Supratim Ghosh, Mustafa Kara, and Necmiye Ozay. Structural detectability of faults in discrete-time affine systems. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6359–6365. IEEE, 2018.

[41] Antoine Girard, Gregor Gössler, and Sebti Mouelhi. Safety controller synthesis for incrementally stable switched systems using multiscale symbolic models. *IEEE Transactions on Automatic Control*, 61(6):1537–1549, 2015.

[42] Antoine Girard, Giordano Pola, and Paulo Tabuada. Approximately bisimilar symbolic models for incrementally stable switched systems. *IEEE Transactions on Automatic Control*, 55(1):116–126, 2009.

[43] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx: Matlab software for disciplined convex programming, 2008.

[44] Inc Gurobi Optimization. Gurobi optimizer reference manual. *URL http://www. gurobi. com*, 2015.

[45] Farshad Harirchi, Zheng Luo, and Necmiye Ozay. Model (in)validation and fault detection for systems with polynomial state-space models. In *American Control Conference (ACC)*, 2016–preprint.

[46] Farshad Harirchi and Necmiye Ozay. Model invalidation for switched affine systems with applications to fault and anomaly detection. In *Proceedings of the Analysis and Design of Hybrid Systems*, pages 260–266, 2015.

[47] Farshad Harirchi and Necmiye Ozay. Guaranteed model-based fault detection in cyber-physical systems: a model invalidation approach. *Automatica*, 93:476 – 488, 2018.

[48] Farshad Harirchi, Sze Zheng Yong, and Necmiye Ozay. Guaranteed fault detection and isolation for switched affine models. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, pages 5161–5167. IEEE, 2017.

[49] Klaus Havelund and Grigore Roşu. An overview of the runtime verification tool java pathexplorer. *Formal methods in system design*, 24(2):189–215, 2004.

[50] Nilson Henao, Sousso Kelouwani, Kodjo Agbossou, and Yves Dubé. Proton exchange membrane fuel cells cold startup global strategy for fuel cell plug-in hybrid electric vehicle. *Journal of Power Sources*, 220:31–41, 2012.

[51] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Multi-layered abstraction-based controller synthesis for continuous-time systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 120–129. ACM, 2018.

[52] Ari Ingimundarson, Jose Manuel Bravo, Vicenç Puig, Teodoro Alamo, and Pedro Guerra. Robust fault detection using zonotope-based set-membership consistency test. *International journal of adaptive control and signal processing*, 23(4):311–330, 2009.

[53] R. Isermann. *Fault-diagnosis systems: an introduction from fault detection to fault tolerance*. Springer Science & Business Media, 2006.

[54] Ali Jadbabaie, Jie Lin, and A Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on automatic control*, 48(6):988–1001, 2003.

[55] Mrdjan Jankovic. Robust control barrier functions for constrained stabilization of nonlinear systems. *Automatica*, 96:359–367, 2018.

[56] Kui Jiao and Xianguo Li. Water transport in polymer electrolyte membrane fuel cells. *Progress in Energy and Combustion Science*, 37(3):221–291, 2011.

[57] Satish G Kandlikar, Zijie Lu, and Thomas A Trabold. Current status and fundamental research needs in thermal management within a pemfc stack,. *in ASME Journal of Fuel Cells Science and Technology*, 2008.

[58] Sertac Karaman, Ricardo G Sanfelice, and Emilio Frazzoli. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *2008 47th IEEE Conference on Decision and Control*, pages 2117–2122. IEEE, 2008.

[59] Parthasarathy Kesavan and Jay H Lee. A set based approach to detection and isolation of faults in multivariable systems. *Computers & Chemical Engineering*, 25(7-8):925–940, 2001.

[60] Marius Kloetzer and Calin Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

[61] Hui Kong, Ezio Bartocci, Sergiy Bogomolov, Radu Grosu, Thomas A Henzinger, Yu Jiang, and Christian Schilling. Discrete abstraction of multiaffine systems. In *International Workshop on Hybrid Systems Biology*, pages 128–144. Springer, 2016.

[62] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Trans. on Robotics*, 25:1370–1381, 2009.

[63] Yinan Li, Jun Liu, and Necmiye Ozay. Computing finite abstractions with robustness margins via local reachable set over-approximation. *arXiv preprint arXiv:1507.06248*, 2015.

[64] Jun Liu and Necmiye Ozay. Finite abstractions with robustness margins for temporal logic-based control synthesis. *Nonlinear Analysis: Hybrid Systems*, 22:1–15, 2016.

[65] Zexiang Liu and Necmiye Ozay. Safety control with preview automaton. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 280–281. ACM, 2019.

[66] Kaushik Mallik, Anne-Kathrin Schmuck, Sadegh Soudjani, and Rupak Majumdar. Compositional abstraction-based controller synthesis for continuous-time systems. *arXiv preprint arXiv:1612.08515*, 2016.

[67] Zohar Manna and Pierre Wolper. Synthesis of communicating processes from temporal logic specifications. In *Workshop on Logic of Programs*, pages 253–281. Springer, 1981.

[68] Mostafa G Mehrabi, A Galip Ulsoy, Yoram Koren, and Peter Heytler. Trends and perspectives in flexible and reconfigurable manufacturing systems. *Journal of Intelligent manufacturing*, 13(2):135–146, 2002.

[69] Pierre-Jean Meyer. *Invariance and symbolic control of cooperative systems for temperature regulation in intelligent buildings*. PhD thesis, Université Grenoble Alpes, 2015.

[70] Pierre-Jean Meyer, Samuel Coogan, and Murat Arcak. Sampled-data reachability analysis using sensitivity and mixed-monotonicity. *IEEE Control Systems Letters*, 2(4):761–766, Oct 2018.

[71] Pierre-Jean Meyer, Alex Devonport, and Murat Arcak. Tira: Toolbox for interval reachability analysis. *arXiv preprint arXiv:1902.05204*, 2019.

[72] Pierre-Jean Meyer and Dimos V Dimarogonas. Hierarchical decomposition of ltl synthesis problem for nonlinear control systems. *IEEE Transactions on Automatic Control*, 2019.

[73] Pierre-Jean Meyer, Antoine Girard, and Emmanuel Witrant. Compositional abstraction and safety synthesis using overlapping symbolic models. *IEEE Transactions on Automatic Control*, 63(6):1835–1841, 2017.

[74] Oscar Mickelin, Necmiye Ozay, and Richard M Murray. Synthesis of correct-by-construction control protocols for hybrid systems using partial state information. In *Proc. of ACC*, pages 2305–2311, 2014.

[75] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on automatic control*, 50(7):947–957, 2005.

[76] George E Monahan. State of the art–a survey of partially observable markov decision processes: theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[77] Thomas Moor, Jörg Raisch, and Siu O'young. Discrete supervisory control of hybrid systems based on l-complete approximations. *Discrete Event Dynamic Systems*, 12(1):83–107, 2002.

[78] Scott J Moura, Duncan S Callaway, Hosam K Fathy, and Jeffrey L Stein. Impact of battery sizing on stochastic optimal power management in plug-in hybrid electric vehicles. In *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, pages 96–102. IEEE, 2008.

[79] Eric A Muller, Anna G Stefanopoulou, and Lino Guzzella. Optimal power control of hybrid fuel cell systems for an accelerated system warm-up. *IEEE transactions on control systems technology*, 15(2):290–305, 2007.

[80] Petter Nilsson. *Correct-by-Construction Control Synthesis for High-Dimensional Systems*. PhD thesis, University of Michigan, December 2017.

[81] Petter Nilsson, Omar Hussien, Yuxiao Chen, Ayca Balkan, Matthias Rungger, Aaron Ames, Jessy Grizzle, Necmiye Ozay, Huei Peng, and Paulo Tabuada. Preliminary results on correct-by-construction control software synthesis for adaptive cruise control. In *Proc. of IEEE CDC*, pages 816–823, 2014.

[82] Petter Nilsson and Necmiye Ozay. Incremental synthesis of switching protocols via abstraction refinement. In *Proc. of IEEE CDC*, pages 6246–6253, 2014.

[83] Petter Nilsson and Necmiye Ozay. Control synthesis for large collections of systems with mode-counting constraints. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*, pages 205–214. ACM, 2016.

[84] Petter Nilsson and Necmiye Ozay. Synthesis of separable controlled invariant sets for modular local control design. In *2016 American Control Conference (ACC)*, pages 5656–5663. IEEE, 2016.

[85] Petter Nilsson, Necmiye Ozay, and Jun Liu. Augmented finite transition systems as abstractions for control synthesis. *Discrete Event Dynamic Systems*, 27(2):301–340, 2017.

[86] Amir Pnueli Nir Piterman and Yaniv Sa'ar. Synthesis of reactive (1) designs. In *Proceedings of the International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 364–380, 2006.

[87] Hassan Noura, Didier Theilliol, Jean-Christophe Ponsart, and Abbas Chamseddine. *Fault-tolerant control systems: Design and practical applications*. Springer Science & Business Media, 2009.

[88] Gurobi Optimization. Inc.,gurobi optimizer reference manual, 2015. *URL: http://www. gurobi. com*, 2014.

[89] Necmiye Ozay, Jun Liu, Pavithra Prabhakar, and Richard M Murray. Computing augmented finite transition systems to synthesize switching protocols for polynomial switched systems. In *Proc. of ACC*, 2013.

[90] Necmiye Ozay, Mario Sznaier, and Constantino Lagoa. Model (in) validation of switched arx systems with unknown switches and its application to activity monitoring. In *49th IEEE Conference on Decision and Control (CDC)*, pages 7624–7630. IEEE, 2010.

[91] Necmiye Ozay, Mario Sznaier, and Constantino Lagoa. Convex certificates for model (in) validation of switched affine systems with unknown switches. *IEEE Transactions on Automatic Control*, 59(11):2921–2932, 2014.

[92] Necmiye Ozay, Mario Sznaier, and Constantino Lagoa. Convex certificates for model (in)validation of switched affine systems with unknown switches. *IEEE Transactions on Automatic Control*, 59(11):2921–2932, 2014.

[93] Benjamin L Pence and Jixin Chen. A framework for control oriented modeling of pem fuel cells. In *ASME 2015 Dynamic Systems and Control Conference*, pages V002T26A002–V002T26A002. American Society of Mechanical Engineers, 2015.

[94] Jay T Pukrushpan, Anna G Stefanopoulou, and Huei Peng. *Control of fuel cell power systems: principles, modeling, analysis and feedback design*. Springer Science & Business Media, 2004.

[95] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 81–87. IEEE, 2014.

[96] Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M Murray, and Sanjit A Seshia. Reactive synthesis from signal temporal logic specifications. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 239–248. ACM, 2015.

[97] Thomas Reinbacher, Kristin Yvonne Rozier, and Johann Schumann. Temporal-logic based runtime observer pairs for system health management of real-time systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 357–372. Springer, 2014.

[98] Matthias Rungger and Paulo Tabuada. Computing robust controlled invariant sets of linear systems. *IEEE Transactions on Automatic Control*, 62(7):3665–3670, 2017.

[99] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. 2016.

[100] Sadra Sadraddini and Calin Belta. Robust temporal logic model predictive control. In *Communication, Control, and Computing (Allerton), 2015 53rd Annual Allerton Conference on*, pages 772–779. IEEE, 2015.

[101] Sadra Sadraddini and Calin Belta. Formal methods for adaptive control of dynamical systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1782–1787. IEEE, 2017.

[102] Meera Sampath, Raja Sengupta, Stephane Lafortune, Kasim Sinnamohideen, and Demosthenis C Teneketzis. Failure diagnosis using discrete-event models. *IEEE transactions on control systems technology*, 4(2):105–124, 1996.

[103] Adnane Saoud and Antoine Girard. Multirate symbolic models for incrementally stable switched systems. *IFAC-PapersOnLine*, 50(1):9278–9284, 2017.

[104] Pier Giuseppe Sessa, Damian Frick, Tony A Wood, and Maryam Kamgarpour. From uncertainty data to robust policies for temporal logic planning. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 157–166. ACM, 2018.

[105] Sumeet Singh, Anirudha Majumdar, Jean-Jacques Slotine, and Marco Pavone. Robust online motion planning via contraction theory and convex optimization. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 5883–5890. IEEE, 2017.

[106] A Prasad Sistla. On characterization of safety and liveness properties in temporal logic. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, pages 39–48. ACM, 1985.

[107] Hal L Smith. Global stability for mixed monotone systems. *Journal of Difference Equations and Applications*, 14(10-11):1159–1164, 2008.

[108] Roy S Smith and John C Doyle. Model validation: A connection between robust control and identification. *IEEE Transactions on automatic control*, 37(7):942–952, 1992.

[109] Fei Sun, Necmiye Ozay, Eric M Wolff, Jun Liu, and Richard M Murray. Efficient control synthesis for augmented finite transition systems with an application to switching protocols. In *Proc. of ACC*, pages 3273–3280, 2014.

[110] Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer, 2009.

[111] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[112] Wolfgang Thomas et al. *Automata, logics, and infinite games: a guide to current research*, volume 2500. Springer Science & Business Media, 2002.

[113] Jan Tretmans. *A formal approach to conformance testing*. Ph.D. Thesis, University of Twente, Netherlands, 1992.

[114] Jan Tretmans. Test generation with inputs, outputs, and quiescence. In *Proceedings of the International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pages 127–146, 1996.

[115] United States Environmental Protection Agency. Dynamometer drive schedules. Available at https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules.

[116] J Watkins and S Yurkovich. Fault detection using set-membership identification. *IFAC Proceedings Volumes*, 29(1):3993–3998, 1996.

[117] Eric M Wolff and Richard M Murray. Optimal control of mixed logical dynamical systems with long-term temporal logic specifications. 2013.

[118] Eric M Wolff, Ufuk Topcu, and Richard M Murray. Efficient reactive controller synthesis for a fragment of linear temporal logic. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 5033–5040. IEEE, 2013.

[119] Eric M Wolff, Ufuk Topcu, and Richard M Murray. Optimization-based trajectory generation with linear temporal logic specifications. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5319–5325. IEEE, 2014.

[120] Pierre Wolper, Moshe Y Vardi, and A Prasad Sistla. Reasoning about infinite computation paths. In *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pages 185–194. IEEE, 1983.

[121] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray. Receding horizon temporal logic planning. *IEEE Trans. Autom. Control*, 57(11):2817–2830, 2012.

[122] Bai Xue, Qiuye Wang, Naijun Zhan, and Martin Fränzle. Robust invariant sets generation for state-constrained perturbed polynomial systems. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 128–137. ACM, 2019.

[123] Liren Yang, Amey Karnik, and Necmiye Ozay. Quickly finding recursively feasible solutions for mpc with discrete variables. In *Control Technologies and Applications (CCTA), 2017 IEEE Conference on*. IEEE, 2018.

[124] Liren Yang, Amey Karnik, Benjamin Pence, Md Tawhid Bin Waez, and Necmiye Ozay. Fuel cell thermal management: Modeling, specications and correct-by-construction control synthesis. In *Proceedings of American Control Conference*, 2017.

[125] Liren Yang, Amey Karnik, Benjamin Pence, Md Tawhid Bin Waez, and Necmiye Ozay. Fuel cell thermal management: Modeling, specifications, and correct-by-construction control synthesis. *IEEE Transactions on Control Systems Technology*, 2019.

[126] Liren Yang, Oscar Mickelin, and Necmiye Ozay. On sufficient conditions for mixed monotonicity. *IEEE Transactions on Automatic Control*, 2019.

[127] Liren Yang and Necmiye Ozay. A note on some sufficient conditions for mixed monotone systems. Technical report, University of Michigan, Department of EECS, 2017. Available at `http://hdl.handle.net/2027.42/136122`.

[128] Liren Yang and Necmiye Ozay. Provably-correct fault tolerant control with delayed information. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 542–549. IEEE, 2017.

[129] Liren Yang and Necmiye Ozay. Robustification and parametrization of switching controllers for a class of set invariance problems. *IFAC-PapersOnLine*, 50(1):1470–1477, 2017.

[130] Liren Yang and Necmiye Ozay. Fault detectability analysis of switched affine systems with linear temporal logic constraints. In *Decision and Control (CDC), 2018 IEEE 58th Annual Conference on*. IEEE, 2018. (to appear).

[131] Liren Yang and Necmiye Ozay. Tight decomposition functions for mixed monotonicity. In *Decision and Control (CDC), 2018 IEEE 58th Annual Conference on*. IEEE, 2018. (to appear).

[132] Liren Yang and Necmiye Ozay. Fault-tolerant output-feedback path planning with temporal logic constraints. In *Decision and Control (CDC), 2018 IEEE 57th Annual Conference on*. IEEE, 2018 (to appear).

[133] Liren Yang, Necmiye Ozay, and Amey Karnik. Synthesis of fault tolerant switching protocols for vehicle engine thermal management. ACC, 2016.

[134] Xiang Yin and Stéphane Lafortune. Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, 61(5):1239–1254, 2016.

[135] Majid Zamani and Murat Arcak. Compositional abstraction for networks of control systems: A dissipativity approach. *IEEE Transactions on Control of Network Systems*, 5(3):1003–1015, 2017.

[136] Majid Zamani, Giordano Pola, Manuel Mazo, and Paulo Tabuada. Symbolic models for nonlinear control systems without stability assumptions. *IEEE Transactions on Automatic Control*, 57(7):1804–1809, 2012.