Provably-Correct Fault Tolerant Control with Delayed Information

Liren Yang and Necmiye Ozay

***Abstract*— In this paper, we study a class of hierarchical finite transition systems representing a set of fault configurations, and we consider synthesizing fault tolerant controllers for such systems that lead to a graceful degradation as faults occur. In previous work, the problem was solved under the assumptions that (i) the specification for each fault configuration is of "reach-avoid-stay" type, (ii) the knowledge of the fault occurrence is immediate. We extend the previous work in two aspects. First, we propose an algorithm that works for specifications given in a more general fragment of linear temporal logic. Secondly, we show how the proposed algorithm can be modified to synthesize controllers that guarantee satisfaction of the specification even in the presence of fault detection delays.**

I. Introduction

Designing resilient systems that can operate in the presence of failures is crucial in many application domains. One key aspect of resiliency is graceful degradation. That is, we expect the system to satisfy certain (possibly relaxed) requirements even when failures occur. In this paper, we address this problem when the systems are modeled as a hierarchy of non-deterministic transition systems, and the requirements for different failure modes are given in linear temporal logic.

In recent years, there has been significant amount of research on synthesizing controllers that can guarantee that the closed-loop system satisfies given logic specifications [12], [4]. The problem we study in this paper and the solution approach follows this line of work. In particular, when faults can be detected and isolated instantaneously, the problem can be reduced to a control synthesis problem on a finite transition system, obtained by flattening the hierarchical structure we start with, subject to temporal logic constraints. By exploiting the fact that the occurrences of faults can be represented by a directed acyclic graph leading to a partial order, we propose an algorithm that avoids this flattening step and obtain a more efficient solution. With further assumptions on the specifications, soundness of the proposed algorithm is proved.

In the second part of the paper, we focus on the case where fault detection and isolation occur with an unknown but bounded delay. This results in controller not having full information of the state at decision time. Synthesis with partial information is in general quite harder as it requires an exponential power set construction to keep track of belief states [5], [10], [15]. Delayed information is a special case of partial information. We show how the algorithm proposed in the first part can be modified to handle detection delays.

The main focus of the paper is to 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 provide a toy example to demonstrate the ideas and leave larger scale implementations on realistic system models (e.g., using abstraction-based methods) for future work.

II. Preliminaries

A. Notation

Let $\mathbb{N}$ be the set of nonnegative integers. For a finite set Σ, a finite word w over Σ is a finite sequence of elements from Σ, i.e., $w = w(0)w(1)w(2)\cdots w(n)$ with $w(t) \in \Sigma$ for all $0 \leq t \leq n$. An ω-word $w = w(0)w(1)w(2)\cdots$ over Σ is an infinite sequence of elements $w(t) \in \Sigma$. Let Σ^* denote the set of all the finite words over Σ, and Σ^ω denote the set of all ω-words over Σ. For an ω-word $w = w(0)w(1)w(2)\cdots$ and any $n \in \mathbb{N}$, $w(0)\cdots w(n)$ is called a prefix of w and $w(n)w(n+1)w(n+2)\cdots$ is called a suffix of w. For a set $S \subseteq (\Sigma^\omega)$, we use $pref(S)$ (resp. $suff(S)$) to denote the set of prefixes (resp. suffixes) of ω-words from S.

B. Finite Transition System with Fault Configurations

Finite transition systems are the basic building blocks of the overall system considered in this paper. A (timed) finite transition system, denoted by TS, is a tuple $(Q, A, \rightarrow, AP, L)$, 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 set of atomic propositions, and $L : Q \rightarrow 2^{AP}$ 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 $t \in \mathbb{N}$. An *execution* ρ of system TS is an infinite sequence of pairs $\big(q(0), a(0)\big)\big(q(1), a(1)\big)\cdots$, where $\big(q(t), a(t), q(t+1)\big) \in \rightarrow$. The *$\omega$-word w_ρ generated by execution ρ* is defined as $w_\rho = L\big(q(0)\big)L\big(q(1)\big), \cdots$.

In this paper we are interested in ***finite transition systems with fault configurations*** (faulty systems for short), which is the discrete analogue of the systems studied in [14]. 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. In what follows, we formally define a finite transition system with fault configurations.

Let $F = \{f_1, \ldots, f_M\}$ be a finite set, each element f_i is called of a *fault configuration*, 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, $f_i \preceq f_j$ means that fault f_j is more severe than or equal to fault f_i, and $f_i \prec f_j$ means that fault

Authors are with the Dept. of EECS, Univ. of Michigan, Ann Arbor, MI 48109, USA yliren,necmiye@umich.edu. This work is supported in part by Ford Motor Co., DARPA and a NASA ECF award.

f_j is strictly more severe. We define the set of minimum elements of $E \subseteq F$ to be

$$\min(E) := \{f_j \in E \mid \nexists f_i \in E \text{ s.t. } f_i \prec f_j\}, \tag{1}$$

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 f_1. Finally we define the successors of a fault $f_i \in F$ to be

$$\text{succ}(f_i) := \min\big(\{f_j \in F \mid f_i \prec f_j\}\big). \tag{2}$$

By definition, fault f_j is a successor of fault f_i if f_j is more sever than f_i and there are no other faults in between.

Let $F = \{f_1, \dots, f_M\}$ be a partially ordered set of fault configurations, a finite transition system with the given fault configurations, denoted by TS^{F}, is a tuple $(Q, F, A, \rightarrow_{TS}, \rightarrow_F, AP, L)$, where

- Q, A, AP have the same meanings as the ones in a regular finite transition systems, F is the given set of fault configurations;
- $\rightarrow_{TS} \subseteq Q \times F \times A \times Q$ 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 f_1, and that $(f_i, f_j) \in \rightarrow_F$ if and only if (iff) $f_j \in \{f_i\} \cup \text{succ}(f_i)$, a fault transition (f_i, f_j) is called *nontrivial* if $f_i \neq f_j$;
- $L : Q \times F \to AP$ is the labeling function.

Similarly to regular systems, we define an execution ρ of system TS^{F} to be an infinite sequence of 3-tuples $\big(q(0), f(0), a(0)\big)\big(q(1), f(1), a(1)\big) \cdots$, where $\big(q(t), f(t), a(t), q(t+1)\big) \in \rightarrow_{TS}$, and $\big(f(t), f(t+1)\big) \in \rightarrow_F$ for all $t \in \mathbb{N}$. The ω-word w_ρ generated by execution ρ is $w_\rho = L\big(q(0), f(0)\big) L\big(q(1), f(1)\big), \cdots$.

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 $f_i \in F$ of the same subscript. Every TS_i has a distinct transition relations $\rightarrow_i := \{(q, a, q') \mid (q, f_i, a, q') \in \rightarrow_{TS}\}$ and different labeling functions $L_i := L(\cdot, f_i)$. 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. Note that, by definition of $\rightarrow_F$, a subsystem TS_i 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.

Secondly, by definition, TS^{F} has a common state set Q, atomic proposition set AP 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_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 $a \in A$.

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 in section IV, 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.

C. Fault Detection with Delay

In this paper, we consider both fault detection with and without delay. In reality, the faults may not be detected immediately after their occurrence. Instead, a detector will collect data from the system in an online manner and make diagnosis based on these data. The detector reports a fault whenever the collected data suggests that the current system behavior is different enough from that of the healthy system, and the fault is isolated whenever the system behavior differs from those of other possible faulty models.

There is research to design such detectors based on model invalidation approaches [8], [7]. In particular, such detectors are guaranteed to detect and isolate the fault within time T after its occurrence. T is a constant depending on the difference between the considered faulty system and other system models, and its value can be computed offline. Note that the actual online fault detection may not take as long as time T. Instead, T is only an upper bound for the actual detection delay.

Also note that these fault detection techniques are developed for continuous state space models governed by difference equations. In this paper we consider finite transition systems, which can be viewed as abstractions that simulate some underlying continuous systems [12]. The delay upper bounds for such abstractions can be obtained from offline analysis of the underlying continuous systems. To be specific, given a faulty system TS^{F} with fault configuration set $F = \{f_1, \cdots, f_M\}$, let f_i be arbitrary current fault, and let $f_j \in \text{succ}(f_i)$ be a possible succeeding fault. We assume a fault detection delay of length T_j is required to isolate fault f_j from other faults in $\{f_i\} \cup \text{succ}(f_i) \setminus \{f_j\}$.

D. Linear Temporal Logic

We use linear temporal logic without next operator (LTL$_{\setminus \bigcirc}$) [3], [9] to specify the correct closed-loop system behavior.

1) Syntax of $\mathrm{LTL}_{\setminus\bigcirc}$*:* Let AP be an atomic propositions set, the syntax of $\mathrm{LTL}_{\setminus\bigcirc}$ formulas over AP is given by

$$\varphi ::= \pi \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \,\mathcal{U}\, \varphi_2, \tag{3}$$

where $\pi \in AP$. With the grammar given in Eq. (3), we define the other propositional and temporal logic operations as follows: $\varphi_1 \wedge \varphi_2 := \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\varphi_1 \to \varphi_2 := \neg\varphi_1 \vee \varphi_2$, $\Diamond\varphi := True \,\mathcal{U}\, \varphi$, $\Box\varphi := \neg\Diamond\neg\varphi$.

2) Semantics of $\mathrm{LTL}_{\setminus\bigcirc}$*:* Given an infinite word $w = w(0)w(1)w(2)\cdots \in (2^{AP})^\omega$ and an $\mathrm{LTL}_{\setminus\bigcirc}$ formula φ, we say φ holds for w at time t (or the word satisfies φ at time t), denoted by $w, t \models \varphi$, if and only if φ holds for $w(t)w(t+1)w(t+2)\cdots$. By this we mean:

- $w, t \models \pi$ iff $\pi \in w(t)$,
- $w, t \models \neg\varphi$ iff $w, t \not\models \varphi$,
- $w, t \models \varphi_1 \vee \varphi_2$ iff $w, t \models \varphi_1$ or $w, t \models \varphi_2$,
- $w, t \models \varphi_1 \,\mathcal{U}\, \varphi_2$ iff $\exists s \geq t : w, s \models \varphi_2$ and $\forall t \leq s' < s : w, s' \models \varphi_1$.

Finally we say $w \models \varphi$ iff $w, 0 \models \varphi$.

3) Linear Time Property: 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 w belonging to P is equivalent to the following: for all $p \in pref(\{w\})$ there exists $s \in (2^{AP})^\omega$ such that $ps \in P$. An LT property P is called a liveness property if for all $p \in (2^{AP})^*$ there exists $s \in (2^{AP})^\omega$ such that $ps \in P$. It is well known that any LT property can be written as the conjunction of a safety property and a liveness property [3]. In particular, such decomposition is not unique but a canonical sharp one exists [3]. That is, there exists a decomposition $P = P^\star_{\text{safe}} \cap P^\star_{\text{liveness}}$, such that for any other decomposition $P = P_{\text{safe}} \cap P_{\text{liveness}}$, one has $P^\star_{\text{safe}} \subseteq P_{\text{safe}}$ and $P_{\text{liveness}} \subseteq P^\star_{\text{liveness}}$.

Given an LTL formula φ (not necessarily excluding next operator) over atomic proposition set AP, the words satisfying φ, i.e., $Word(\varphi) := \{w \in (2^{AP})^\omega \mid w \models \varphi\}$, is a linear time property over the same AP. In particular, this LT property is ω-regular, which can be equivalently described by a nondeterministic Buchi automaton (NBA) [3]. By LT property decomposition,

$$Word(\varphi) = Word_{\text{safe}}(\varphi) \cap Word_{\text{liveness}}(\varphi). \tag{4}$$

By [2], two new NBA's can be constructed (from the automaton that generates $Word(\varphi)$), one generates $Word_{\text{safe}}(\varphi)$, and the other generates $Word_{\text{liveness}}(\varphi)$. Converting these two new NBA's into LTL formulas φ_{safe}, $\varphi_{\text{liveness}}$, we have $\varphi = \varphi_{\text{safe}} \wedge \varphi_{\text{liveness}}$.

In this paper, we consider a special type of $\mathrm{LTL}_{\setminus\bigcirc}$ formula that specifies an absolutely decomposable property, which is defined as follows.

Definition 1: Let $P \subseteq (2^{AP})^\omega$ be a property over set AP. Property P is called *absolutely decomposable* if there exists a decomposition $P = P_{\text{safe}} \cap P_{\text{liveness}}$, such that

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

Note that $pref(P_{\text{liveness}}) = (2^{AP})^*$, thus the definition of absolute liveness coincides with the one given by [1]. Some useful results are listed and proven in Appendix A.

4) Two-player Temporal Logic Game: The systems (both regular and faulty ones) studied in this paper are nondeterministic, the actual evolution of the system can be viewed as the outcome of a two player game between the controller and the environment. In each round of the game, the controller picks an action first, then the environment picks a transition that is available under the current state and action [13]. More formally, given a finite transition system $TS = (Q, A, \to, AP, L)$, a *control strategy* is a partial function $\mu : (Q \times A)^* \times Q \to A$ that assigns a next-step action based on execution history, and an *environment strategy* $\eta : (Q \times A)^* \to Q$ defines the next-step state. The μ-η-controlled execution starting from initial state q_0, denoted by $\rho^{\mu\text{-}\eta}(q_0)$ is an execution $\big(q(0), a(0)\big)\big(q(1), a(1)\big)\big(q(2), a(2)\big)$ of TS such that (i) $q(0) = q_0$, (ii) $a(t+1) = \mu\big(q(0), a(0), \cdots, q(t), a(t), q(t+1)\big)$, and (iii) $q(t+1) = \eta\big(q(0), a(0), \cdots q(t), a(t)\big)$. The objective of the controller is to give a strategy μ so that the words $w_{\rho^{\mu\text{-}\eta}(q_0)}$ generated by all the μ-η-controlled executions starting from q_0 satisfy some given $\mathrm{LTL}_{\setminus\bigcirc}$ formula φ, regardless of the move η of the environment. Such a strategy is called *winning* for initial state q_0. We define the *maximum winning set* $Win(\varphi, TS)$ to be the set of all states that can have a winning strategy, i.e., $Win(\varphi, TS) := \{q_0 \in Q \mid \exists\mu : \forall\eta : w_{\rho^{\mu\text{-}\eta}(q_0)} \models \varphi\}$. We say W is a *winning set* if $W \subseteq Win(\varphi, TS)$.

III. Problem Statement

In this paper we consider synthesizing correct-by-construction controller that leads to a graceful degradation for an finite transition system with fault configuration set $F = \{f_1, \ldots, f_M\}$. The correct behavior of the closed-loop system, i.e., the so called graceful degradation, is specified by the following $\mathrm{LTL}_{\setminus\bigcirc}$ formula:

$$\Phi = \bigwedge_{f_i \in F} \left(\Diamond\Box f_i \to \varphi_i\right), \tag{5}$$

where φ_i is an $\mathrm{LTL}_{\setminus\bigcirc}$ formula specifying the system's desired behavior when the final fault configuration is $f_i \in F$.

Eq. (5) says: if the fault configuration eventually stays at f_i, the specification φ_i associated with this fault is achieved. Note that the fault status of a faulty system is guaranteed to reach a specific configuration $f_i \in F$ 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.

We now formally define the two problems considered in this paper.

Problem 1: [**Synthesis with Immediate Fault Detection**] Given a fault configuration F, let TS^{F} be finite transition system with fault configuration F, and let Φ be an LTL formula (over the same AP) in the form of Eq. (5). Assuming that a fault is detected immediately after it occurs, find a winning set $W \subseteq Win(TS^{\mathrm{F}}, \Phi)$ and the winning strategies associated with each state in the set W.

Problem 2: [**Synthesis with Delayed Fault Detection**] The problem is stated the same as Problem 1, except that any fault $f_i \in F$ requires at most time T_j to detect and isolate.

IV. Solution Approach

A. Solution to Problem 1

This part gives an algorithm that solves Problem 1 in a recursive manner. We start with introducing the idea of the proposed algorithm in an intuitive way. We then explain the meaning of the returned values of the algorithm, and comment on how these returned values can be used to solve Problem 1. Finally the correctness of the algorithm under certain assumptions is stated.

First, notice that a major challenge in solving Problem 1 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 f_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 f_j. In particular, this requires the following to hold in case the system degrades to fault configuration f_j:

(I1) (**bad prefix issue**) the finite word generated by the old strategy does not violate the new specification φ_j,

(I2) (**succeeding strategy issue**) there is a new strategy to achieve specification φ_j starting from the current state.

Finally note that the argument also applies to the new fault f_j and its succeeding configurations, if any. This hence suggests a recursive algorithm.

The above intuition is formalized by Algorithm 1.

1) Inputs: Algorithm 1 takes a system TS^{F}, an $\text{LTL}_{\setminus\bigcirc}$ formula Φ in form of Eq, (5), and a fault configuration f_i as inputs. Fault f_i can be seen as the initial configuration. Note that a faulty system always starts from being healthy by definition, here f_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 f_i. $\mathcal{K}_i := \{K_j\}_{f_j \succeq f_i}$ is a collection of maps K_j, each map K_j relates a state to a strategy. It might be helpful to think $K_j(q)$ as a strategy that achieves specification φ_j if the system starts from state q and stays at fault configuration f_j forever. The fault-tolerant strategy, with initial fault f_i and initial state q_0, 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:

$$\mu\Big(\big(q(0)=q_0, f(0), a(0)\big)\cdots\big(q(t), f(t)\big)\Big)$$
$$=K_n\big(q(s)\big)\Big(\big(q(s), a(s)\big), \cdots\big(q(t-1), a(t-1)\big)q(t)\Big), \tag{6}$$

where n in "K_n" is the subscript of latest fault $f(t)$, and

$$s = \min_{\substack{0 \,\leq\, \tau \,\leq\, t \\ f(\tau) \,=\, f(t)}} \tau. \tag{7}$$

Finally the function also returns an LTL formula ψ_i called *strengthened formula*, which is obtained by strengthening φ_i by additional safety specifications. ψ_i can be seen as the specification of an overall system that captures all possible degradations from current fault f_i.

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 two oracles. In what follows we always assume the two oracles are sound.

(1) $[\psi_j^{\text{safe}}, \psi_j^{\text{liveness}}] =$ **Decomp**(ψ_j) does the sharpest LT property decomposition;

(2) $[W_i, K_i] =$ **Win**(ψ_i, TS_i) 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 $\text{LTL}_{\setminus\bigcirc}$ formula ψ_i.

If a worst fault is reached, function **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 ψ_j^{safe}; to guarantee the existence of succeeding strategies, 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 ψ_i and this finishes the round of the recursion.

Algorithm 1 $[W_i, \mathcal{K}_i, \psi_i] =$ **Win**$^{\mathrm{F}}(\Phi, TS^{\mathrm{F}}, f_i)$

```
Initialize $W_i = \emptyset$, $\mathcal{K}_i = \emptyset$, $\psi_i = \varphi_i$
if $f_i \in \max(F)$ then
    $[W_i, K_i] = \mathbf{Win}(\varphi_i, TS_i)$
    $\mathcal{K}_i = \{K_i\}$
else
    for $f_j \in \mathrm{succ}(f_i)$ do
        $[W_j, \mathcal{K}_j, \psi_j] = \mathrm{Win}^{\mathrm{F}}(\Phi, TS^{\mathrm{F}}, f_j)$
        $[\psi_j^{\text{safe}}, \psi_j^{\text{liveness}}] = \mathbf{Decomp}(\psi_j)$
        $\psi_i = \varphi_i \wedge (\Box W_j) \wedge \psi_j^{\text{safe}}$
    $[W_i, K_i] = \mathbf{Win}(\psi_i, TS_i)$
    $\mathcal{K}_i = \mathcal{K}_j \cup \{K_i\}$
return $W_i$, $\mathcal{K}_i$, $\psi_i$
```

Theorem 1: Assume that each φ_i specifies an absolutely decomposable property, then Algorithm 1 is sound in the sense that every state in W_1 is a winning set w.r.t. overall specification Φ, with each state in W_1 equipped with a winning strategy defined by Eq. (6).

Proof of Theorem 1 can be found in Appendix B.

As mentioned earlier at the end of section II-B, faulty system TS^{F} can be modeled by a regular transition system with state space $Q \times F$. Problem 1 hence can be solved theoretically by solving a Rabin game [4], whose complexity is given by

$$\mathcal{O}\left(\left(|A|\,|Q|\,|F|\,2^{(2^{|\Phi|}|\Phi)|}\right)^{2k}\right), \tag{8}$$

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. (5), and k is the number of accepting pairs in associated Rabin automaton, which is a small number that is usually equal to 1 [6].

The complexity of Algorithm 1, ignoring the complexity for LTL formula decomposition, is given by

$$\mathcal{O}\left(|F|\left(|A|\,|Q|\,2^{(2^{|\varphi|}|\varphi)|}\right)^{2k}\right), \tag{9}$$

where $|\varphi| = \max_{i:f_i \in F} |\varphi_i|$. The complexity of our approach is linear in $|F|$, the number of faults, while the complexity in Eq. (8) contains term $\mathcal{O}(|F|\,2^{(2^{|\Phi|}|\Phi)|})^{2k})$ where $|\Phi|$ is linear in $|F|$.

B. Solution to Problem 2

In this part, we modify Algorithm 1 to solve a special class of instances of Problem 2. In what follows, we first briefly discuss the challenges when there are delays for fault detection. Then we restrict ourselves to the problems in which the considered fault configuration set is a chain. Such problems allow a solution by a simple modification to Algorithm 1.

We first address the challenges caused by detection delay. Assume that the system degrades from configuration f_i to f_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 (f_i, f_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 φ_j for some possible succeeding fault configuration $f_j \succ f_i$;

(I4) (**succeeding strategy issue**) the execution may be led to parts of the state space where no strategies are available to achieve specification φ_j for some succeeding faults.

Note that the bad prefix issue (I3) cannot be solved by simply applying Algorithm 1. In Algorithm 1 (line 9), the specifications under succeeding faults are taken into consideration when synthesizing controller for current fault f_i. But the synthesis is done for system TS_i, not 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 ψ_i, hence violate ψ_j^{safe} 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.

As a preliminary step to solving Problem 2, in what follows in the paper, we consider a special class of faulty systems whose fault configuration set F is a chain. That is, for any f_i, $f_j \in F$, f_i and f_j are comparable. Without loss of generality, we assume that $f_1 \preceq f_2 \preceq \cdots \preceq f_M$. In order to solve the two challenges raised above for such special systems, we modify Algorithm 1 in two aspects. In line 10 of Algorithm 1, (i) pair (ψ_i, TS_i) is replaced by a pair $(\psi_i{}', TS')$ with so called one-step-margin, (ii) oracle **Win** is strengthened by some invariance restrictions. Next we explain these two modifications in details.

1) One-Step-Margin by Positive Normal Form: To solve issue (I3), we define a pair (ψ_i', TS_i') with *one-step-margin* for a given pair (ψ_i, TS_i). System TS_i is the subsystem associated with fault f_i of a faulty system TS^{F}, whose fault configuration set $F = \{f_1, \cdots, f_M\}$ is a chain. We assume that f_{i+1} is the only successor of fault f_i.

We first introduce the following notations for the given regular finite transition system $TS_i = (Q, A, \rightarrow_i, AP, L_i)$. For all $\pi \in AP$, define $[\pi]_i := \{q \in Q \mid \pi \in L_i(q)\}$ and $[\neg\pi]_i := AP \setminus [\pi]_i$. For a state set $S \subseteq Q$, define its one-step-out set to be

$$\begin{aligned} \textbf{Out}_i(S) := \{q \in S \mid \exists a \in A, q' \notin S : \\ (q, a, q') \in \rightarrow_i \cup \rightarrow_{i+1}\}, \end{aligned} \tag{10}$$

where $\rightarrow_{i+1}$ is the transition relation for the only succeeding system TS_{i+1}.

Now the pair (ψ_i', TS_i') is defined as follows.

- $AP' = AP \cup \{\pi' \mid \pi \in AP\}$, with π' representing the "negation" of proposition π.
- $TS_i' = (Q, A, \rightarrow_i, AP', L_i')$ is a finite transition system, where Q, A, $\rightarrow$ are the same as the underlying transition system TS_i above, and $L_i' : Q \rightarrow 2^{AP'}$ is defined to be such that

$$\begin{aligned} \forall \pi \in AP : \pi \in L_i'(q) \Leftrightarrow q \in [\pi]_i \setminus \textbf{Out}([\pi]_i) \\ \pi' \in L_i'(q) \Leftrightarrow q \in [\neg\pi]_i \setminus \textbf{Out}([\neg\pi]_i) \end{aligned} \tag{11}$$

- For the given LTL$_{\setminus\bigcirc}$ formula ψ_i, write ψ_i in positive normal form [3], and replace every $\neg\pi$ in the formula by $\pi' \in AP'$. We denote the obtained formula by ψ_i'.

The usefulness of one-step-margin pair (ψ_i', TS_i') is as follows. First, if a strategy achieves ψ_i' on system TS_i', it also achieves ψ_i on system TS_i. Secondly, if the system degrades from TS_i to TS_{i+1}, and ψ_i' is violated at some point by the old strategy, ψ_i will not be violated for at least one more step. Note that this one step margin is enough for the detector to identify the fault. This is because formula ψ_i' is already violated, while we know ψ_i' should have been achieved if the system evolved with TS_i. Based on the above argument, one can synthesize a strategy that achieves ψ_i' on TS_i', then apply this strategy to TS_i, the prefix generated is guaranteed not to violate ψ_i before the end of uninformed execution horizon. This prefix, by the assumption of absolute decomposability of φ_i (and hence ψ_i), is a good prefix in terms of the specification of the final fault (see proof of Theorem 1, observation (b)). This hence solves issue (I3) stated at the beginning of this section.

Also note that the one-step-margin modification only works when the faults form a chain. The reason is that, one-step-margin is only enough for fault detection, but not necessarily enough for the fault isolation whenever there are multiple possible succeeding faults. As will be discussed in section V, however, such one-step-margin modification is not necessary for some special cases.

2) Synthesis with Invariance Restrictions: We now consider solving the succeeding strategy issue (I4). For this purpose, we modify oracle **Win**. Before line 10 of Algorithm 1 where **Win** is called, a line is inserted, in which 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 [11]. Then oracle **Win** is called as before, except that (i) the input is replaced by the one-step-margin pair (ψ_i', TS_i') , and (ii) the synthesis is restricted within state set C_j, while the actions available at each state $q \in C_j$ are restricted within $I_j(q)$.

By the above modification, the states will always stay within $C_j \subseteq W_j$ even after the system degrades from fault configuration f_i to f_j. This hence guarantees a succeeding strategy after an uninformed execution horizon of any length. Also note that the chain structure assumption on the fault set is not essential for this modification to work. When there are multiple possible faults f_j succeeding current fault f_i, an intersection set C can be computed as $\bigcap_{j: f_j \in \mathrm{succ}(f_i)} C_j$, and a map I can be defined to be such that $I(q) = \bigcap_{j: f_j \in \mathrm{succ}(f_i)} I_j(q)$. Then winning set W_i can be synthesized by **Win** restricted to state set C and control actions in $I(q)$.

We summarize the soundness of the above modifications by the following proposition, whose proof follows from the soundness of Algorithm 1 and the above arguments.

Proposition 1: In addition to the hypothesises in Theorem 1, assume that the considered fault set is a chain, and that a further fault does not occur during any uninformed execution horizon. Algorithm 1 is sound for Problem 2, if its 10th line is replaced with $[W_i, K_i] = \textbf{Win}(\psi_i', TS_i')$, where (ψ_i', TS_i') is the one-step-margin pair of (ψ_i, TS_i), and the synthesis in oracle **Win** is done with states restricted in controlled invariant set C_j, and with the actions at $q \in C_j$ restricted in invariant action set $I_j(q)$.

V. Discussion

Several discussions are presented below, regarding to the two modifications of Algorithm 1 in section IV-B, and the relation between this work and the previous one.

First, note that an invariance property is absolute safety property (Proposition 3 in Appendix A). Also note that if ψ_j^{safe} (line 9 of Algorithm 1) specifies an invariance property, then $Word(\Box W_j) \subseteq Word(\psi_j^{\mathrm{safe}})$. This suggests that the bad prefix issue (I3) is automatically solved if the synthesis in oracle **Win** is restricted within controlled invariant set $C_j \subseteq W_j$. Hence there is no need to create the one-step-margin pair, which leads to a potentially more conservative solution. In this case, since no additional time margin is needed, it is not required to assume that the faults form a chain.

Secondly, if strengthened formula ψ_j in Algorithm 1 specifies a so called suffix-closed property[1], it is not hard to show that any winning set W_j w.r.t. ψ_j is controlled invariant under the corresponding winning strategies. This suggests that C_j, the largest controlled invariant set contained by winning set W_j, is equal to W_j.

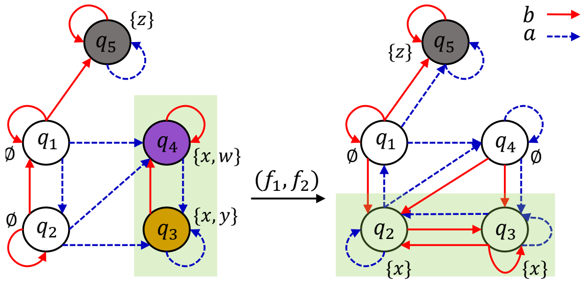

Fig. 1: Faulty system TS^{F}. Left: regular system TS_1 associated with fault f_1, right: TS_2 associated with fault f_2. Different colors marked different propositions predicate: w (purple), x (green box), y (orange), z (grey).

In previous work [14], the fault structure is not a chain, but we only consider reach-avoid-stay type of requirements for each fault configuration. That is, φ_i's in Eq. (5) are in the form $(\Box s_i) \wedge (\Diamond \Box g_i)$, where s_i is an atomic proposition marking states that are considered to be safe at fault configuration f_i, and g_i is an atomic proposition that marks the goal state. Note that $\Box s_i$ specifies an invariant property, while $\Diamond \Box g_i$ specifies a suffix-closed, absolute liveness property. Moreover, it is not hard to show by induction that ψ_i's in Algorithm 1 can also be written as a conjunction of an invariance formula and $\Diamond \Box g_i$. By the discussions in the above paragraphs, the controller synthesis with detection delay can be simply solved by Algorithm 1 with invariant restrictions proposed in section IV-B.2

Finally, a notable property of our solution approach to Problem 2 is that it does not rely on the value of T, the upper bound for the detection delay. Instead, it is sufficient to know that T exists, i.e., is finite. This is a result of the following facts. First, one-step-margin is enough for fault detection in a chain. Secondly, we synthesize controllers restricted to invariant actions of $C_j \subseteq W_j$, hence the states stay inside the winning set W_j of the next fault for arbitrarily long time.

VI. Example

In this section we present a toy example that illustrates the proposed solution approach. Consider a faulty system $TS^{\mathrm{F}} = (Q, F, A, \to_{TS}, \to_F, AP, L)$, where $Q = \{q_1, q_2, q_3, q_4, q_5\}$, $F = \{f_1, f_2\}$, $A = \{a, b\}$, $AP = \{w, x, y, z\}$. The only fault transition in $\to_F$ is (f_1, f_2). The system transitions $\to_{TS}$ and the labeling function are defined in Fig. 1. The graceful degradation of system TS^{F} is specified by an LTL$_{\backslash \bigcirc}$ formula in form of Eq. 5, where

$$\varphi_1 = (\Box \neg z) \wedge (\Diamond \Box x) \wedge (\Box \Diamond w) \wedge (\Box \Diamond y), \tag{12}$$

$$\varphi_2 = (\Box \neg z) \wedge (\Diamond \Box x) \tag{13}$$

In what follows we compute the fault tolerant winning set $\mathrm{Win}^{\mathrm{F}}(\Phi, TS^{\mathrm{F}})$ for both cases with and without detection delay, to demonstrate the difference introduced by the delay.

1) Winning Set without Detection Delay: To find the fault tolerant winning set, Algorithm 1 starts from fault f_2 as base case. Winning set W_2 can be found as $\{q_2, q_3, q_4\}$. We then

[1] A property P over set AP is called suffix-closed, iff any suffix of a word in P also belongs to P, i.e., $uv \in P \Rightarrow v \in P$.

go to fault f_1, and strengthen φ_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 φ_2. Finally we compute $W_1 = \textbf{Win}(\psi, TS_1) = \{q_2, q_3, q_4\}$, 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 φ_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 φ_2.

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. First note that the safety part of φ_2 specifies an invariance property. As discussed in section V, the one-step-margin is not needed in this example. Then, similar as before, Algorithm 1 starts from fault f_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 f_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 φ_2 can not be achieved.

VII. Appendix

A. Properties of Absolutely Decomposable Property

In this part, some useful results regarding to absolutely decomposable properties are presented.

First, we give a lemma about general safety properties that is used in the later proofs.

Lemma 1: Let P_1 and P_2 be two safety property over AP, $\mathit{pref}(P_1) = \mathit{pref}(P_2)$ implies that $P_1 = P_2$.

Proof: Assume for a contradiction that $\mathit{pref}(P_1) = \mathit{pref}(P_2)$ but $P_1 \neq P_2$. Without loss of generality, this means there exists $w \in P_1$ such that $w \notin P_2$. Since $w \notin P_2$ and P_2 is a safety property, we immediately know that w has a bad prefix $w_t := w(0)w(1)\cdots w(t) \notin \mathit{pref}(P_2)$. But on the other hand, $w \in P_1$ and this implies that $w_t \in \mathit{pref}(P_1) = \mathit{pref}(P_2)$, which is a contradiction. ■

The following propositions are used in proving soundness of Algorithm 1.

Proposition 2: Let P be an absolutely decomposable property, then for all $p \in \mathit{pref}(P)$, $w \in P$, $pw \in P$.

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

Proposition 3: An invariance property P is an absolute safety property.

Proposition 4: Let P_1, P_2 be two absolute safety properties, $P = P_1 \cap P_2$ is also absolute safety property.

Proof: Proposition 4, 3 easily follow from the definition of absolute safety properties. ■

Proposition 5: Let P be an absolutely decomposable property with the specific decomposition $P = P_{\text{safe}} \cap P_{\text{liveness}}$, then $\mathit{pref}(P) = \mathit{pref}(P_{\text{safe}})$.

Proof: From the proof of Proposition 2, we already know that $\mathit{pref}(P) \subseteq \mathit{pref}(P_{\text{safe}})$. Next we show the other direction. For this purpose, let $p \in \mathit{pref}(P_{\text{safe}})$ and $w \in P$ be arbitrary. Next we show $pw \in P$ and conclude $p \in \mathit{pref}(P)$.

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

(b) Secondly, also note that $p \in \mathit{pref}(P_{\text{liveness}}) = (2^{AP})^*$, and $w \in P \subseteq P_{\text{liveness}}$. By bullet 2 in Definition 1, we have $pw \in P_{\text{liveness}}$.

Combining (a) and (b), we have $pw \in P_{\text{safe}} \cap P_{\text{liveness}} = P$. Therefore $p \in \mathit{pref}(P)$ and this finishes the proof. ■

Proposition 6: Let P be an absolutely decomposable property with a specific decomposition $P = P_{\text{safe}} \cap P_{\text{liveness}}$, and let $P = P^\star_{\text{safe}} \cap P^\star_{\text{liveness}}$ be the sharpest decomposition, then $P_{\text{safe}} = P^\star_{\text{safe}}$.

Proof: By $P = P^\star_{\text{safe}} \cap P^\star_{\text{liveness}}$, we have $P \subseteq P^\star_{\text{safe}}$. This hence gives

$$\mathit{pref}(P) \subseteq \mathit{pref}(P^\star_{\text{safe}}). \tag{14}$$

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

$$\mathit{pref}(P^\star_{\text{safe}}) \subseteq \mathit{pref}(P_{\text{safe}}). \tag{15}$$

Combine (14), (15), we have

$$\mathit{pref}(P) \subseteq \mathit{pref}(P^\star_{\text{safe}}) \subseteq \mathit{pref}(P_{\text{safe}}). \tag{16}$$

But by Proposition 5, we know that $\mathit{pref}(P) = \mathit{pref}(P_{\text{safe}})$, which forces all "$\subseteq$" in Eq. (16) to be "$=$". Thus we have $\mathit{pref}(P^\star_{\text{safe}}) = \mathit{pref}(P_{\text{safe}})$. Applying Lemma 1, we have $P^\star_{\text{safe}} = P_{\text{safe}}$. ■

Proposition 7: Let P_1 be an absolutely decomposable property under decomposition $P_1 = P_{1,\text{safe}} \cap P_{1,\text{liveness}}$, and let $P_{2,\text{safe}}$ be an absolute safety property, then $P = P_1 \cap P_{2,\text{safe}}$ is absolutely decomposable under $P = P_{\text{safe}} \cap P_{\text{liveness}}$, where $P_{\text{safe}} = P_{1,\text{safe}} \cap P_{2,\text{safe}}$ and $P_{\text{liveness}} = P_{1,\text{liveness}}$.

Proof: First note that P_{safe} is indeed a safety property and $P_{\text{safe}} \cap P_{\text{liveness}} = (P_{1,\text{safe}} \cap P_{2,\text{safe}}) \cap P_{\text{liveness}} = P_{2,\text{safe}} \cap (P_{1,\text{safe}} \cap P_{1,\text{liveness}}) = P$ is a valid decomposition. Moreover, by Proposition 4, P_{safe} is a absolute safety property. By definition P is absolutely decomposable, and $P_{\text{safe}} = P_{1,\text{safe}} \cap P_{2,\text{safe}}$ is the unique absolute safety property involved in the decomposition by Proposition 6. ■

B. Proof of Theorem 1

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

Let W_{i_k}'s be the winning sets and ψ_{i_k}'s be the strengthened formulas returned in each round of recursion. Regarding to 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 1), 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 1, we have

$$\begin{aligned}&Word(\psi_{i_{k-1}})\\ =&Word(\varphi_{i_{k-1}}) \cap Word(\Box W_{i_k}) \cap Word(\psi^{\text{safe}}_{i_k}),\end{aligned} \tag{17}$$

where $Word(\varphi_{i_{k-1}})$ is absolutely decomposable by assumption, $Word(\Box W_{i_k})$ is an absolute safety property by Proposition 3, and $Word(\psi^{\text{safe}}_{i_k})$ is absolute safety property presuming that ψ_{i_k} is absolutely decomposable. One can easily verify by induction that $Word(\psi_{i_k})$ is absolutely decomposable, using Proposition 3, 4, 7. Next applying Proposition 5, this implies

$$pref\big(Word(\psi_{i_k})\big) = pref\big(Word(\psi^{\text{safe}}_{i_k})\big) \tag{18}$$

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

$$pref\big(Word(\psi_{i_{k-1}})\big) \subseteq pref\big(Word(\psi^{\text{safe}}_{i_k})\big). \tag{19}$$

Combining Eq. (18) (19), we have

$$pref\big(Word(\psi_{i_{k-1}})\big) \subseteq pref\big(Word(\psi_{i_k})\big). \tag{20}$$

With the same argument used to obtain Eq. (19),

$$pref\big(Word(\psi_{i_n})\big) \subseteq pref\big(Word(\varphi_{i_n})\big). \tag{21}$$

Now to prove the soundness, consider an execution starting from $q_0 \in W_{i_1}$ under control strategy μ constructed by Eq. (6) and arbitrary environment strategy η

$$\rho^{\mu\text{-}\eta}(q_0) = \big(q(0) = q_0, f(0), a(0)\big)\big(q(1), f(1), a(1)\big)\cdots, \tag{22}$$

and the word generated by this execution

$$w_{\rho^{\mu\text{-}\eta}(q_0)} = w(0)w(1)w(2), \cdots. \tag{23}$$

First, let t_{i_k} denote the time instant when fault transition (f_{i_k-1}, f_{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 ψ_{i_1}. Hence $\Box W_{i_2}$, which is part of ψ_{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 $w_{i_k} := w(t_{i_k})\cdots w(t_{i_{k+1}} - 1)$ be the word segment that is generated under fault f_{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 ψ_{i_k}. Therefore $w_{i_k} \in pref\big(Word(\psi_{i_k})\big)$. By observation (b), this means $w := w_{i_1}w_{i_2}\cdots w_{i_n} \in pref\big(Word(\varphi_{i_n})\big)$.

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 $w = w(0)\cdots w(t_{i_n} - 1)$ generated so far belongs to $pref\big(Word(\varphi_{i_n})\big)$. Note that the succeeding strategy will focus on achieving φ_{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 $v = v(t_{i_n})v(t_{i_n} + 1)\cdots \in Word(\varphi_{i_n})$. Recall that φ_{i_n} is absolutely decomposable. By Proposition 2, the overall word $wv \models \varphi_{i_n}$. This proves the soundness of Algorithm 1 under the given assumptions.

References

[1] B. Alpern and F. B. Schneider. Defining liveness. *Information processing letters*, 21(4):181–185, 1985.

[2] B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed computing*, 2(3):117–126, 1987.

[3] C. Baier and J. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[4] C. Belta, B. Yordanov, and E. A. Gol. *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017.

[5] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games with imperfect information. In *International Workshop on Computer Science Logic*, pages 287–302. Springer, 2006.

[6] S. Coogan, M. Arcak, and C. Belta. Formal methods for control of traffic flow. 2016.

[7] F. Harirchi, Z. Luo, and N. Ozay. Model (in) validation and fault detection for systems with polynomial state-space models. In *American Control Conference (ACC), 2016*, pages 1017–1023. IEEE, 2016.

[8] F. Harirchi and N. Ozay. Model invalidation for switched affine systems with applications to fault and anomaly detection. *IFAC-PapersOnLine*, 48(27):260–266, 2015.

[9] J. Liu and N. Ozay. Finite abstractions with robustness margins for temporal logic-based control synthesis. *Nonlinear Analysis: Hybrid Systems*, 22:1–15, 2016.

[10] G. 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.

[11] P. Nilsson and N. Ozay. Incremental synthesis of switching protocols via abstraction refinement. In *Proc. of IEEE CDC*, pages 6246–6253, 2014.

[12] P. Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer, 2009.

[13] W. Thomas, T. Wilke, et al. *Automata, logics, and infinite games: a guide to current research*, volume 2500. Springer Science & Business Media, 2002.

[14] L. Yang, N. Ozay, and A. Karnik. Synthesis of fault tolerant switching protocols for vehicle engine thermal management. In *American Control Conference (ACC), 2016*, pages 4213–4220. IEEE, 2016.

[15] X. Yin and S. Lafortune. Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, 61(5):1239–1254, 2016.