

Diagnosis of Repeated Failures For Discrete Event Systems With Linear-Time Temporal Logic Specifications *

Shengbing Jiang (shengbing.jiang@gm.com)
GM R&D and Planning, Warren, MI 48090-9055

Ratnesh Kumar (rkumar@iastate.edu)
Dept. of Elec. & Comp. Eng., Iowa State Univ., Ames, IA 50014

Abstract

In our earlier work we introduced a state-based approach for the diagnosis of repeatedly occurring failures in discrete event systems (DESs). Since temporal logic provides a simpler way of specifying system properties, in this paper a temporal logic based approach for diagnosing the occurrence of a repeated number of failures is developed. Linear-time temporal logic (LTL) formulae are used to represent the specifications of DESs. Notions of prediagnosability for failures and diagnosability for repeated failures are introduced in the setting of temporal logic. A polynomial algorithm for the test of prediagnosability for failures is provided. The diagnosis problem for repeated failures in the temporal logic setting is reduced to one in a state-based setting, and so the prior results of state-based repeated failure diagnosis can be applied. Finally, a simple example is given for illustration.

Keywords: Discrete event system, repeated failure diagnosis, linear-time temporal logic.

Note to Practitioners: By failure we mean execution of a behavior that violates the desired specifications. Certain types of failures can occur repeatedly. In a manufacturing facility different types of parts follow different paths, and routing violations in such a discrete flow network can occur repeatedly. Further, system specifications are expressed more naturally in a temporal logic. This paper presents a framework for diagnosing repeatedly occurring failures for specifications expressed in the linear-time temporal logic. The notions of failure-traces, indicator-traces, failure-prediagnosability, failure-points, and repeated failure diagnosability are introduced in this setting, and algorithms are presented to check for failure-prediagnosability as well as various notions of repeated failure diagnosability. The theory developed is illustrated through an example that abstractly represents a traffic monitoring example.

*The research was supported in part by the National Science Foundation under the grants NSF-ECS-9709796, NSF-ECS-0099851, NSF-ECS-0218207, NSF-ECS-0244732, NSF-EPNES-0323379, and NSF-0424048, and a DoD-EPSCoR grant through the Office of Naval Research under the grant N000140110621.

1 Introduction

A failure in a system is the execution of a behavior that violates a specification representing system’s nominal behavior. Examples of failures include execution of a faulty event [22], reaching a faulty state [28], or more generally violating a formal specification expressed, say, in a temporal logic [11]. The task of failure analysis is to monitor the system behavior, and determine the occurrence of any failures (called failure detection), and identify its type or origin (called failure isolation or diagnosis).

A notion of failure diagnosability of discrete event systems was introduced in [22]. A system is said to be diagnosable if execution of a failure can be detected within a bounded number of occurrences of state-transitions/events. A method for constructing a diagnoser was developed, and a necessary and sufficient condition of diagnosability was obtained in terms of certain properties of the constructed diagnoser. An algorithm of polynomial complexity for testing diagnosability without having to construct a diagnoser was obtained in [10, 27]. Also, by first applying such a test, the construction of a diagnoser for a system that is not diagnosable can be avoided. The test was further extended to systems containing cycles of unobservable events in [12].

The notion of diagnosability developed in [22] was applied to failure diagnosis in HVAC (heating, ventilating, and air-conditioning) systems in [23], and a distributed implementation of the diagnoser was reported in [7]. In [21], the notion of failure diagnosis was extended to a notion of active failure diagnosis, where one could control the behavior of the system to better diagnose it. A template based approach to failure detection in timed discrete event system was developed in [9, 6, 17]. Common to all these work is that they are “event-based”. “State-based” approaches and their extension to timed systems were considered in [16, 15, 28]. It should be noted that the two approaches (event-based and state-based) are equivalent and polynomially transformable to one another. Failure diagnosis has also been studied in the setting of Petri nets, see for example [1, 14, 18].

[7, 20, 24, 2, 25] studied distributed diagnosis in which diagnosis is performed by diagnosers communicating with each other either directly or through a coordinator, thereby pooling together the observations. It is shown in [24] that distributed diagnosis with unbounded communication delay is undecidable. The decentralized diagnosis problem studied in [19] on the other hand considers diagnosis by local diagnosers without having them gather information at a central location. It introduces the notion of *co-diagnosability* and provides polynomial algorithms for verification of co-diagnosability as well as for synthesis of diagnosers.

In all these prior work, the occurrence of a failure is specified either as the occurrence of a faulty event (“event-based” approach) or as the visit of a faulty state (“state-based” approach), or as execution of a trace outside a specification language. However, more generally a failure can be defined to be the execution of a state/event trace that violates a given formal specification, such as reaching a “deadlock” state, or reaching a “live-lock” set of states, or reaching a state from where no “fair execution” is possible in future. We say a trace to be a failure-trace if its execution implies that the given formal specification has already been violated, whereas we say a trace to be an indicator-trace if its execution implies that the given formal specification is violated by all its feasible infinite extensions (the specification may not be violated by the trace or any of its finite extensions). A failure-trace, on the other

hand, is a trace for which any of its infinite extension (feasible as well as infeasible) violates the given specification. A failure-trace is also an indicator-trace, but the converse need not hold.

In order to capture faults in a more natural way, we proposed in [11] the use of linear-time temporal logic (LTL) to express failure specifications. We reduce the problem of failure diagnosis to that of model-checking [3], and an algorithm of complexity polynomial in the number of system states and exponential in the length of the LTL specification formula was obtained for failure diagnosis. A diagnoser that is a nondeterministic state machine, and that has a size that is polynomial in the size of the system states and exponential in the length of the LTL specification formula, was also obtained. Having such a nondeterministic representation of the diagnoser makes it practical to have it stored and utilized since a deterministic representation is likely to have a size that is exponential in the size of system states.

Most prior work on failure analysis of discrete event systems considered whether or not a failure occurred *sometimes* in past. But certain failures can occur repeatedly and detection/diagnosis of such failures each time they occur is important. For example at a bottle filling station, a multiple number of bottles may be filled improperly. Another example of a repeatedly occurring failure is an intermittent or non-permanent failure, such as a loose wire. Similarly in a discrete flow network such as manufacturing or communication or transportation, a routing violation can occur repeatedly. The work on template approach [6] to failure detection considered detection of such repeatedly occurring failures, but it did not formalize a notion that will allow detecting/diagnosing a failure each time it occurs. A state-based framework for diagnosing repeated failures in DESs was introduced in [12], where several notions of repeated failure diagnosability were introduced, generalizing the notion of diagnosability developed in [22], where the objective was to diagnose the occurrence of a failure, but not to diagnose each time it occurs. Polynomial algorithms for testing the various notions of repeated failure diagnosability, as well as a procedure of polynomial complexity for the on-line diagnosis of repeated failures, were obtained in [12]. Algorithms of one-order better complexity have recently been reported in [26]. The work reported in [5] considered intermittent failures (failures that can reset automatically), and presented a technique to determine whether or not an intermittent failure is “active”, by reducing this problem to an instance of the diagnosis problem formulated in [22].

In this paper, we present a temporal logic based approach for diagnosis of repeated failures. This also extends our earlier work on temporal logic based approach to diagnosis of (un-repeated) failures [11]. Given a DES to be diagnosed, we use a LTL formula f for the purpose of specifying a specification. An infinite trace of the system is said to be *faulty* if it violates the given LTL formula. If the goal of diagnosis is to detect and diagnose a failure the first time it occurs, it suffices to detect/diagnose the occurrence of an indicator-trace, which indicates that a failure is inevitable (the failure may not have occurred yet). On the other hand, for the goal of diagnosing a failure every time it occurs, we need the stronger notion of a failure-trace, whose occurrence indicates that a failure has already occurred.

Given a LTL specification f , a system is said to be *f -failure prediagnosable* if every infinite f -failure trace generated by the system possesses a finite f -failure trace as its prefix. The property of f -failure prediagnosability should be viewed as a precondition for repeated failure diagnosability, since without this property, the detection of the violation of the specification

can not be deduced through the observation of finite length traces, even under a complete observation of events. Note that this property automatically holds if the specification is only a “safety” specification. We prove that the f -failure prediagnosability is equivalent to a relative ω -closure property.

Since we are interested in the diagnosis of repeatedly occurring failures, it is natural to consider only such specifications that can be violated repeatedly. Without loss of generality, we can consider those specifications that are required to hold globally at each state of the system, rather just at the initial state. In other words, the LTL formula representing the specification is in the form of Gf , where G stands for “Globally” (in all states) and f is a LTL formula.

In order to determine the number of times the specification Gf is violated along a trace executed by the system, the notion of Gf -failure points is introduced, where each Gf -failure point of a trace indicates a violation of the Gf specification along that trace. Thus for diagnosis of repeated failures, we only need to diagnose the number of occurrences of the Gf -failure points along a trace executed by the system.

We introduce the notions of K-diagnosability (diagnosability of Kth occurrence of a failure), [1,K]-diagnosability (diagnosability of first K occurrences of a failure), and $[1,\infty]$ -diagnosability (diagnosability of all occurrences of a failure) in the temporal logic setting, similar to those in the state-based setting [12]. We further show that the repeated diagnosis in the temporal logic setting can be transformed to the one in the state-based setting. An algorithm is provided for this transformation, through which the results of our earlier work on repeated failure diagnosis in state-based setting can be applied.

The main contributions of the paper are as follows:

1. Notion of repeated diagnosis for detection/diagnosis of repeatable failures is formulated in the linear temporal logic (LTL) setting. LTL is more expressive than the $*$ -languages and can specify the properties of ω -languages.
2. The notion of *failure prediagnosability* is introduced, which is a condition under which finite-length observations can be used to deduce violations of properties expressed in LTL, i.e., properties of infinite-length traces.
3. A test for failure prediagnosability is provided.
4. The notion of *failure points* of the traces of a system is introduced. These indicate the positions within a trace where the given specification is violated. Visiting a failure point during the execution of a trace indicates the occurrence of a failure. Thus the number of failure points in a trace indicates the number of times the given specification is violated along the execution of that trace. Repeated diagnosability requires that the visiting of any failure point be detected within a uniformly bounded delay.
5. An algorithm for identifying the failure points is obtained.
6. Repeated diagnosability of temporal logic setting is algorithmically transformed to repeated diagnosability of state-based setting of [12]. This way the existing algorithms can be directly applied for testing the various notions of diagnosability as well as for synthesizing the diagnosers.

The rest of the paper is organized as follows. Section 2 gives the definition of LTL temporal logic. In Section 3, the notions of f -failure trace and f -failure prediagnosability are introduced, and an algorithm for testing the f -failure prediagnosability is provided. In Section 4, the Gf -failure point is defined, and an algorithm for identifying Gf -failure points is obtained. Section 5 presents the definitions of various notions of diagnosability of repeated failure in the temporal logic setting, and shows that the repeated diagnosis in the temporal logic setting can be transformed to that in the stated-based setting. Finally, Section 6 presents an illustrative example, and Section 7 concludes the work presented.

2 Notations and Preliminaries

In this paper, we use LTL temporal logic to express the specifications of DESs for the purpose of failure diagnosis. In the following, we give the definition of LTL. For a complete introduction to temporal logic, readers may refer to [8].

Let AP be a finite set of atomic proposition symbols. Using the atomic propositions and boolean connectives such as conjunction, disjunction, and negation, one can construct more expressions describing properties of states. However one is also interested in describing the properties of sequences of states. Such properties are expressed using *temporal operators* of a temporal logic. LTL temporal logic is a specific temporal logic formalism. The following temporal operators are used in LTL for describing the properties along a specific state-trace.

- X (“next time”): It requires that a property hold in the next state of the state-trace.
- U (“until”): It is used to combine two properties. The combined property holds if there is a state in the state-trace where the second property holds, and at every preceding state in the state-trace, the first property holds.
- F (“eventually” or “in the future”): It is used to assert that a property will hold at some future state in the state-trace. It is a special case of “until”.
- G (“globally” or “always”): It specifies that a property holds at every state in the state-trace.
- B (“before”): It also combines two properties. It requires that if there is a state in the state-trace where the second property holds, then there exists a preceding state in the state-trace where the first property holds.

Next we give the syntax of LTL. LTL formulae are generated by rules P1-P3 given below.

P1 If $p \in AP$, then p is a LTL formula.

P2 If f_1 and f_2 are LTL formulae, then so are $\neg f_1$, $f_1 \vee f_2$, and $f_1 \wedge f_2$.

P3 If f_1 and f_2 are LTL formulae, then so are Xf_1 , f_1Uf_2 , Ff_1 , Gf_1 , and f_1Bf_2 .

We use $|f|$ to denote the length of f , which is the number of boolean and temporal operators in f .

Next we give the semantics of LTL, which is defined over infinite propositions-traces. Let $\Sigma_{AP} = 2^{AP}$, then Σ_{AP}^* be the set of all finite propositions-traces over AP , and Σ_{AP}^ω be the set of all infinite propositions-traces over AP . For a LTL formula f and $\pi \in \Sigma_{AP}^\omega$, the notation $\pi \models f$ (resp., $\pi \not\models f$) means that f holds (resp., does not hold) along the infinite propositions-trace π . The relation \models is defined inductively as follows, where we assume that f_1 and f_2 are LTL formulae, and for $\pi = (L_0, L_1, \dots) \in \Sigma_{AP}^\omega$, $\pi^i = (L_i, \dots)$ for any $i \geq 0$:

1. If $f_1 \in AP$, then $\pi \models f_1 \iff f_1 \in L_0$.
2. $\pi \not\models f_1 \iff \neg(\pi \models f_1)$.
3. $\pi \models \neg f_1 \iff \pi \not\models f_1$.
4. $\pi \models f_1 \vee f_2 \iff \pi \models f_1$ or $\pi \models f_2$.
5. $\pi \models f_1 \wedge f_2 \iff \pi \models f_1$ and $\pi \models f_2$.
6. $\pi \models Xf_1 \iff \pi^1 \models f_1$.
7. $\pi \models f_1 U f_2 \iff \exists k \geq 0, \pi^k \models f_2$ and $\forall j \in \{0, 1, \dots, k-1\}, \pi^j \models f_1$.
8. $\pi \models Ff_1 \iff \exists k \geq 0, \pi^k \models f_1$.
9. $\pi \models Gf_1 \iff \forall k \geq 0, \pi^k \models f_1$.
10. $\pi \models f_1 B f_2 \iff \forall k \geq 0$ with $\pi^k \models f_2, \exists j \in \{0, 1, \dots, k-1\}, \pi^j \models f_1$.

The semantics of LTL are defined only over infinite propositions-traces above. As mentioned in [8], we can extend the semantics of LTL to finite propositions-traces as follows: for a finite propositions-trace (L_0, \dots, L_n) , it satisfies a LTL formula f if and only if the infinite propositions-trace $(L_0, \dots, L_n, L_n, \dots) = (L_0, \dots, L_n^\omega)$ satisfies f .

In the following, we first introduce some notations of formal languages [13], and then mention that a LTL formula can be characterized as the language accepted by a non-deterministic generalized Büchi automaton [4].

Let Σ be a finite event set, Σ^* be the set of all finite length sequences of events from Σ including the zero length trace ϵ , Σ^ω be the set of all infinite length sequences of events from Σ , and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. K is called a **-language* over Σ if $K \subseteq \Sigma^*$, and B is called a *ω -language* over Σ if $B \subseteq \Sigma^\omega$. The *prefix* operation $pr : \Sigma^\infty \rightarrow \Sigma^*$ is defined as:

$$\forall B \subseteq \Sigma^\infty, pr(B) := \{s \in \Sigma^* \mid \exists e \in B : s \text{ is a prefix of } e\}.$$

The *limit* operation $lim : \Sigma^* \rightarrow \Sigma^\omega$ is defined as:

$$\forall K \in \Sigma^*, lim(K) := \{e \in \Sigma^\omega \mid \exists \text{ infinitely many } n \in \mathcal{N} \text{ s.t. } e^n \in K\},$$

where e^n denotes the prefix of length n of e , i.e., $e^n = e(1)e(2)\dots e(n)$ provided that $e = e(1)e(2)\dots$. Given a ω -languages $B \subseteq \Sigma^\omega$, B is said to be *ω -closed* if $B = lim(pr(B))$. Note that $B \subseteq lim(pr(B))$ always holds. B is said to be *relatively ω -closed* with respect to another ω -language $S \subseteq \Sigma^\omega$ if $B \cap S = lim(pr(B)) \cap S$.

Given a LTL formula f , let S_f denote the set of all infinitely long propositions-traces over AP satisfying f . Then one can obtain a non-deterministic generalized Büchi automaton (for details, refer to [4]) that accepts the ω -language S_f . The non-deterministic generalized Büchi automaton can be represented as

$$T_f = (Q_f, \Sigma_{AP}, R_f, q_0^f, \mathcal{F}),$$

where

- Q_f is a finite state set;
- $\Sigma_{AP} = 2^{AP}$ is the event set;
- $R_f \subseteq Q_f \times \Sigma_{AP} \times Q_f$ is the state-transition relation;
- $q_0^f \in Q_f$ is the initial state;
- $\mathcal{F} = \{F_i, 1 \leq i \leq r\} \subseteq 2^{Q_f}$ is the generalized Büchi acceptance condition.

An infinite length propositions-trace $\pi_p = (L_1, L_2, \dots) \in \Sigma_{AP}^\omega$ is accepted by T_f if and only if there exists an infinite length state-trace $\pi = (q_0, q_1, \dots)$ in T_f such that $q_0 = q_0^f$, $(q_{i-1}, L_i, q_i) \in R_f$ for all $i \geq 1$, and π visits each $F_i \in \mathcal{F}$ ($i = 1, \dots, r$) infinitely often. We use $L_{T_f}^\omega$ to denote the ω -language accepted by T_f , then we have $L_{T_f}^\omega = S_f$. Both the complexity of the construction of T_f , and the number of states in T_f , are exponential in the length of the LTL formula f .

The following definition of repeated failure diagnosability in a state-based setting is taken from [12]. Suppose that the discrete event system P to be diagnosed for repeated failures is modeled by a 6-tuple, $P = (X_P, \Sigma, R, X_0, \mathcal{F}, \psi)$, where

- X_P is a finite set of states;
- Σ is a finite set of event labels;
- $R \subseteq X_P \times (\Sigma \cup \{\epsilon\}) \times X_P$ is a state transition relation (here P is allowed to be non-deterministic);
- $X_0 \subseteq X_P$ is the set of initial states;
- $\mathcal{F} = \{F_i, i = 1, 2, \dots, m\}$ is the set of failure types;
- $\psi : X_P \rightarrow 2^{\mathcal{F}}$ be the failure assignment function for each state in X_P .

A finite or infinite state-trace $\pi = (x_0, x_1, \dots)$ is called *contained* in P if for all $i > 0$ there exists a $\sigma_i \in \Sigma \cup \{\epsilon\}$ such that $(x_{i-1}, \sigma_i, x_i) \in R$; π is called *generated* by P if π is contained in P and $x_0 \in X_0$. We use Tr_P to denote the set of all finite state-traces generated by P .

Observations of events executed by P are filtered through an observation mask $M : \Sigma \cup \{\epsilon\} \rightarrow \Delta \cup \{\epsilon\}$ with $M(\epsilon) = \epsilon$, where Δ is the set of observed symbols and it may be disjoint with Σ . The observation mask M is used to capture the effect of sensors that monitor and report the occurrence of events. The definition of M can be extended to event-traces inductively as follows: $\forall s \in \Sigma^*, \sigma \in \Sigma, M(s\sigma) = M(s)M(\sigma)$.

A finite or infinite event-trace (e_1, e_2, \dots) over $\Sigma \cup \{\epsilon\}$ is said to be associated with a state-trace $\pi = (x_0, x_1, \dots)$ if $\forall i > 0, (x_{i-1}, e_i, x_i) \in R$. We use E_π to denote the set of all event-traces associated with a state-trace $\pi \in \text{Tr}_P$, and O_π denote the observations of event-traces in E_π , i.e., $O_\pi = \{M(s) \in \Delta^* \mid s \in E_\pi\}$. For any two finite state-traces $\pi_1 = (x_0^1, x_1^1, \dots, x_{k_1}^1)$ and $\pi_2 = (x_0^2, x_1^2, \dots, x_{k_2}^2)$ in Tr_P , π_1 and π_2 are called *indistinguishable* (with respect to the mask M) if $O_{\pi_1} \cap O_{\pi_2} \neq \emptyset$, i.e., if they share observations resulting from the execution of associated event-traces.

For all $F_i \in \mathcal{F}$ and $\pi \in \text{Tr}_P$, let $N_\pi^{F_i}$ denote the number of states in π labeled with a F_i -type failure; in which case π is said to contain $N_\pi^{F_i}$ failures of type F_i .

Definition 1 Given a system P , an observation mask M , and a failure assignment function ψ , P is said to be K -diagnosable ($K \geq 1$), (resp., $[1, K]$ -diagnosable, or $[1, \infty]$ -diagnosable) with respect to M and ψ if the following holds:

$$\begin{aligned} & (\forall F_i \in \mathcal{F})(\exists n_i \in \mathcal{N}) \\ & (J = K) \text{ (resp., } (\forall J, 1 \leq J \leq K), \text{ or } (\forall J \geq 1)) \\ & (\forall \pi_0 \in \text{Tr}_P, N_{\pi_0}^{F_i} \geq J) \\ & (\forall \pi = \pi_0 \pi_1 \in \text{Tr}_P, |\pi_1| \geq n_i) \\ & (\forall \pi' \in \text{Tr}_P, O_\pi \cap O_{\pi'} \neq \emptyset) \\ & \Rightarrow (N_{\pi'}^{F_i} \geq J), \end{aligned}$$

where \mathcal{N} is the set of all natural numbers and Tr_P is the set of all finite state-traces generated by P .

Definition 1 states that a system is K -diagnosable (resp., $[1, K]$ -diagnosable, or $[1, \infty]$ -diagnosable), if the execution of any state-trace containing at least J failures of a same failure type ($J = K$ for K -diagnosability, $1 \leq J \leq K$ for $[1, K]$ -diagnosability, and $J \geq 1$ for $[1, \infty]$ -diagnosability) can be deduced with a finite delay from the observed behavior through the mask M . More precisely, for any failure type F_i , there exists a number n_i such that for any state-trace π_0 containing at least J failures of the type F_i , for any sufficient long (at least n_i states longer) extension π_1 of π_0 and for any finite state-trace π' generated by P , if π' and $\pi = \pi_0 \pi_1$ are indistinguishable with respect to M , i.e., if they can generate a same masked event-trace ($O_\pi \cap O_{\pi'} \neq \emptyset$), then π' must also contain at least J failures of the type F_i .

3 Notions of f -Failure Traces and Prediagnosability

In the temporal logic setting, the system P to be diagnosed for occurrence of failures is modeled by a 6-tuple, $P = (X_P, \Sigma, R, X_0, AP, L)$, where X_P, Σ, R , and X_0 are the same as before, AP is a finite set of atomic propositions, and $L : X_P \rightarrow 2^{AP}$ is a labeling function such that $\forall x \in X_P, p \in L(x)$ means that p holds at x , and $p \notin L(x)$ means that p does not hold at x .

A finite or infinite propositions-trace (L_0, L_1, \dots) over AP is said to be associated with a state-trace $\pi = (x_0, x_1, \dots)$ if $\forall i > 0, L_i = L(x_i)$. A finite or infinite propositions-trace over

AP is called contained (resp., generated) by P if it is associated with a state-trace contained (resp., generated) by P . We use $\pi_{AP} = (L(x_0), L(x_1), \dots)$ to denote the propositions-trace associated with the state-trace π , and use $L_P^{(\omega, AP)} \subseteq \Sigma_{AP}^\omega$ to denote the set of all infinite length propositions-traces over AP that are generated by P . For a given LTL formula f and an infinite state-trace $\pi = (x_0, x_1, \dots)$, $\pi \models f$ if and only if $\pi_{AP} \models f$. For a state-trace $\pi_1 = (x_0^1, x_1^1, \dots)$ (finite or infinite) and a finite state-trace $\pi_2 = (x_0^2, x_1^2, \dots, x_k^2)$, if the number of states in π_1 is more than k , i.e., $|\pi_1| > k$, and $x_i^1 = x_i^2$ for $0 \leq i \leq k$, then π_2 is called a k -prefix of π_1 , π_1 is called an *extension* of π_2 in P , and π_1 can be represented as $\pi_1 = \pi_2\pi$, where $\pi = (x_{k+1}^1, \dots)$ is called the k -suffix of π_1 .

For diagnosing the occurrence of a failure the first time it occurs, the notion of indicator-traces was introduced in [11]. The execution of an indicator-trace implies that a failure is inevitable in future (if it has not already occurred). For diagnosis of repeated failures, it is not enough to just detect the inevitability of a failure, rather its actual occurrence. So, a stronger notion, that of failure-traces, is being introduced next.

Definition 2 Let P be a system, f be a LTL formula, π be a finite or infinite state-trace generated in P , and π_{AP} be the propositions-trace associated with π , then π (resp., π_{AP}) is called a f -failure state-trace (resp., f -failure propositions-trace), if all infinite extensions of π_{AP} in Σ_{AP}^ω are faulty, i.e., for any infinite propositions-trace $\pi' = \pi_{AP}\pi_1 \in \Sigma_{AP}^\omega$, $\pi' \not\models f$. A Gf -failure trace of $P = (X_P, \Sigma, R, X_0, AP, L)$ is defined to be a f -failure trace of $P^0 = (X_P, \Sigma, R, X_P, AP, L)$, where P^0 is obtained by treating every state in P as an initial state.

Remark 1 Note that in Definition 2, when π is an infinite trace, then it is f -failure trace if and only if $\pi \not\models f$. This coincides with the definition introduced in [11], and so the above definition should be viewed as a generalization of the one given in [11]. Also note that the specification Gf holds starting from an initial state of P if and only if the specification f holds at each state of P , or equivalently, the specification f holds for P^0 , which is obtained from P by treating each state of P as an initial state.

In [11], we introduced the notion of indicator traces, where a finite indicator trace indicates that a failure is inevitable, but it may not have happened. A finite failure trace on the other hand implies that a failure has already occurred. The main difference between a finite failure state-trace and an indicator state-trace is that for a finite failure state-trace, all its infinite extensions (not necessarily those feasible in the system) are faulty, whereas for a finite indicator state-trace, only those infinite extensions that are feasible in the system are faulty. It follows that a finite failure state-trace indicates that a failure has already occurred, and a finite indicator state-trace only indicates that a failure is inevitable in the system. It is evident that a finite failure trace is an indicator trace, but the converse may not hold, as shown by Example 1 below.

Note that for an infinite trace violating a specification, it may or may not have a finite failure state-trace as its prefix. Since only finite failure state-traces could be detected through the finite observations, this motivates the following definition of f -failure prediagnosability, which is similar to the definition of indicator-prediagnosability given in [11].

Definition 3 Let P be a system and f be a LTL formula, P is said to be f -failure prediagnosable if every infinite f -failure state-trace in P possesses a finite f -failure state-trace as

its prefix. P is said to be Gf -failure prediagnosable if every infinite Gf -failure state-trace in P possesses a finite Gf -failure state-trace as its prefix.

Remark 2 From Definitions 2 and 3, we know that the Gf -failure prediagnosability requires that every infinite Gf -failure state-trace contained in the system (not necessarily started from an initial state of the system) should possess a finite Gf -failure state-trace as its prefix.

Example 1 Consider the system shown in Figure 1, and suppose the specification is given

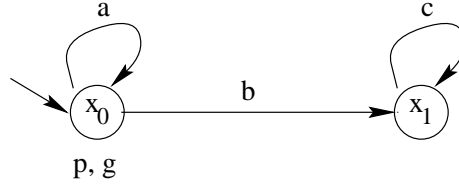


Figure 1: Example of prediagnosability for failures

by, $f = GFp$. It is easy to verify that for an infinite failure trace $\pi = (x_0^k, x_1^\omega)$ generated by the system, no finite prefix of π is a f -failure state-trace. Because for any prefix $\pi^i = (x_0^k, x_1^i)$, $(\pi^i x_0^\omega) \models f$ (note that $(\pi^i x_0^\omega)$ is not feasible in the system). However the finite prefix (x_0^k, x_1) of π is an indicator state-trace. Thus, the system is not prediagnosable for *failures* with respect to f , but it is prediagnosable for *indicators* with respect to f .

Now suppose the specification is given as $f' = G(g \Rightarrow Fp)$, then it is easy to check that the system is trivially f' -failure prediagnosable. (No infinite f' -failure state-trace can be generated by the system.)

The following result follows directly from Definitions 2 and 3, and reduces the problem of Gf -failure prediagnosability to that of f -failure prediagnosability.

Theorem 1 Let P be a system and f be a LTL formula, then P is Gf -failure prediagnosable if and only if P^0 is f -failure prediagnosable, where P^0 is the same as P except that every state in P is an initial state of P^0 .

For the test of f -failure prediagnosability, we have the following result.

Theorem 2 Let P be a system and f be a LTL specification. P is f -failure prediagnosable if and only if the ω -language $S_f \subseteq \Sigma_{AP}^\omega$ is relatively ω -closed with respect to $L_P^{(\omega, AP)}$, i.e., $S_f \cap L_P^{(\omega, AP)} = \lim(\text{pr}(S_f)) \cap L_P^{(\omega, AP)}$, where $\Sigma_{AP} = 2^{AP}$, S_f denotes the set of all infinite propositions-traces over AP satisfying f , and $L_P^{(\omega, AP)}$ denotes the set of all propositions-traces generated by P .

Proof: For necessity, suppose P is f -failure prediagnosable. For contradiction, suppose $S_f \subseteq \Sigma_{AP}^\omega$ is not relatively ω -closed with respect to $L_P^{(\omega, AP)}$, i.e., $\exists u = (e_1, e_2, \dots)$ in $(\lim(\text{pr}(S_f)) \cap L_P^{(\omega, AP)}) - (S_f \cap L_P^{(\omega, AP)})$. Since $u \in \lim(\text{pr}(S_f)) \cap L_P^{(\omega, AP)} \subseteq L_P^{(\omega, AP)}$ and $u \notin S_f \cap L_P^{(\omega, AP)}$, so $u \notin S_f$, i.e., $u \not\models f$. Since u is an infinite f -failure trace in P and P is f -failure prediagnosable, u must have a finite f -failure trace prefix $u^n = (e_1, \dots, e_n)$. Because $u \in \lim(\text{pr}(S_f)) \cap L_P^{(\omega, AP)}$, from the definition of \lim operation, there must exist a $k > n$ such

that $u^k = (e_1, \dots, e_k) \in pr(S_f)$, which implies that u^k is a prefix of some non-faulty trace in S_f . It follows that u^n could not be a f -failure trace, which is a contradiction. So the necessity holds.

For sufficiency, suppose suppose $S_f \subseteq \Sigma_{AP}^\omega$ is relatively ω -closed with respect to $L_P^{(\omega, AP)}$, i.e., $(lim(pr(S_f)) \cap L_P^{(\omega, AP)}) - (S_f \cap L_P^{(\omega, AP)}) = \emptyset$. For contradiction, if P is not f -failure prediagnosable, then from Definition 3 we know there exists an infinite f -failure state-trace $\pi = (x_1, x_2, \dots)$ in P such that no finite prefix of π is a f -failure state-trace. In other words, $u^n = (L(x_1), \dots, L(x_n))$ is a prefix of some non-faulty trace in S_f for every $n \geq 1$, i.e., $u^n \in pr(S_f)$ for every $n \geq 1$. Let $u = (L(x_1), \dots)$ be the propositions-trace associated with π , then it is obvious that $u \in L_P^{(\omega, AP)}$ and $u \not\models f$. Because $u^n \in pr(S_f)$ for every $n \geq 1$, we have $u \in lim(pr(S_f))$. Since $u \not\models f$, $u \notin S_f$. From above we have

$$u \in (lim(pr(S_f)) \cap L_P^{(\omega, AP)}) - (S_f \cap L_P^{(\omega, AP)}),$$

i.e., $(lim(pr(S_f)) \cap L_P^{(\omega, AP)}) - (S_f \cap L_P^{(\omega, AP)}) \neq \emptyset$, which is a contradiction to the hypothesis. So P is f -failure prediagnosable. \blacksquare

Remark 3 Note that if f represents some safety properties only, then it can be verified that S_f is ω -closed, i.e., $lim(pr(S_f)) = S_f$. Thus, it holds automatically that S_f is relatively ω -closed with respect to any $L_P^{(\omega, AP)}$, i.e., every system is failure-prediagnosable with respect to a formula f that represents safety properties only.

Remark 4 In [11], we proved that P is prediagnosable for indicators if and only if $S_f \cap L_P^{(\omega, AP)}$ is ω -closed. Thus, the condition for indicator-prediagnosability is the ω -closure of $S_f \cap L_P^{(\omega, AP)}$, whereas the condition for failure-prediagnosability is the relative ω -closure of S_f with respect to $L_P^{(\omega, AP)}$. The two conditions seem different in nature, but they are not. In fact the first one (for indicator-prediagnosability) is equivalent to the relative ω -closure of $S_f \cap L_P^{(\omega, AP)}$ with respect to $L_P^{(\omega, AP)}$. To see this, the relative ω -closure of $S_f \cap L_P^{(\omega, AP)}$ with respect to $L_P^{(\omega, AP)}$ requires that

$$lim(pr(S_f \cap L_P^{(\omega, AP)})) \cap L_P^{(\omega, AP)} \subseteq S_f \cap L_P^{(\omega, AP)}.$$

Since $lim(pr(S_f \cap L_P^{(\omega, AP)})) \subseteq lim(pr(L_P^{(\omega, AP)}))$, and $L_P^{(\omega, AP)}$ is ω -closed (P simply “accepts” all infinite state-traces it “generates”), i.e., $lim(pr(L_P^{(\omega, AP)})) = L_P^{(\omega, AP)}$, the relative ω -closure of $S_f \cap L_P^{(\omega, AP)}$ in above is equivalent to

$$lim(pr(S_f \cap L_P^{(\omega, AP)})) \subseteq S_f \cap L_P^{(\omega, AP)},$$

which is the ω -closure of $S_f \cap L_P^{(\omega, AP)}$.

Next, it is clear that the relative ω -closure of S_f implies the relative ω -closure of $S_f \cap L_P^{(\omega, AP)}$, as $lim(pr(S_f \cap L_P^{(\omega, AP)})) \subseteq lim(pr(S_f))$, which implies

$$lim(pr(S_f \cap L_P^{(\omega, AP)})) \cap L_P^{(\omega, AP)} \subseteq lim(pr(S_f)) \cap L_P^{(\omega, AP)} \subseteq S_f \cap L_P^{(\omega, AP)},$$

where the last inequality follows from the relative ω -closure of S_f . So, the failure prediagnosability is a stronger notion than the indicator prediagnosability, as expected.

Finally, for failure-prediagnosability, S_f plays the role of $S_f \cap L_P^{(\omega, AP)}$. In fact if we set $L_P^{(\omega, AP)} = \Sigma_{AP}^\omega$, then S_f and $S_f \cap L_P^{(\omega, AP)}$ become the same, and the failure-prediagnosability and the indicator-prediagnosability also become the same. This is to be expected, since when $L_P^{(\omega, AP)} = \Sigma_{AP}^\omega$, then it follows from the definitions of indicator and failure traces that the two notions coincide.

The following algorithm for testing failure-prediagnosability follows directly from Theorem 2.

Algorithm 1 Algorithm for testing failure-prediagnosability

1. This step is for the construction of a non-deterministic generalized Büchi automaton $T_f = (C_f, \Sigma_{AP}, R_f, q_0^f, \mathcal{F})$ that accepts all the infinite propositions-traces satisfying f . Let $L_{T_f}^\omega$ denote the ω -language accepted by T_f , i.e., the set of all infinite propositions-traces accepted by T_f , which could also be denoted as $L_{T_f}^{(\omega, AP)}$, then we have $L_{T_f}^\omega = L_{T_f}^{(\omega, AP)} = S_f$.
2. This step is for the construction of a state machine that generates the ω -language $\text{lim}(pr(S_f))$. The state machine is obtained from T_f by iteratively deleting those states $q \in C_f$ and their associated transitions if either q has no successor, or no state labeled with some $F_i \in \mathcal{F}$ can be reached from q , until no more such states and transitions can be deleted. With a slight abuse of notation, we use T_f to denote the state machine obtained in this step.
3. This step is for the construction of a state machine T_1 that generates the ω -language $\text{lim}(pr(S_f)) \cap L_P^{(\omega, AP)}$. $T_1 = (Q_1, \Sigma, R_1, Q_0^1, AP \cup \mathcal{F}, L_1)$ is constructed from the proposition synchronization of T_f and $P = (X_P, \Sigma, R, X_0, AP, L)$ as follows:
 - $Q_1 = C_f \times X$ is the set of states;
 - Σ is the set of events;
 - $R_1 \subseteq Q_1 \times (\Sigma \cup \{\epsilon\}) \times Q_1$ is the transition relation, $R_1 = \{((c, x), \sigma, (c', x')) \in Q_1 \times (\Sigma \cup \{\epsilon\}) \times Q_1 \mid (c, L(x'), c') \in R_f, (x, \sigma, x') \in R\}$;
 - $Q_0^1 = \{(c, x) \in Q_1 \mid (q_0^f, L(x), c) \in R_f, x \in X_0\}$ is the set of initial states;
 - $AP \cup \mathcal{F}$ is the new set of atomic propositions;
 - $L_1 : Q_1 \rightarrow 2^{AP \cup \mathcal{F}}$ is the labeling function such that

$$\begin{aligned} \forall (c, x) \in Q_1, F_i \in \mathcal{F}, p \in AP : \\ [F_i \in L_1(c, x) \Leftrightarrow c \in F_i] \wedge [p \in L_1(c, x) \Leftrightarrow p \in L(x)]. \end{aligned}$$

- Iteratively delete each state $q \in Q_1$ and its associated transitions if q has no successor, until no more such states and transitions can be deleted.

Let $L_{T_1}^{(\omega, AP)}$ denote the set of all infinite length propositions-traces that are generated by T_1 . Then from the construction of T_1 above, $L_{T_1}^{(\omega, AP)} = \text{lim}(pr(S_f)) \cap L_P^{(\omega, AP)}$.

4. Check whether S_f is relatively ω -closed with respect to $L_P^{(\omega, AP)}$, i.e., whether

$$L_{T_1}^{(\omega, AP)} = \text{lim}(pr(S_f)) \cap L_P^{(\omega, AP)} \subseteq S_f.$$

This is done by checking whether every infinite propositions-trace generated by T_1 visits the F_i -labeled ($1 \leq i \leq r$) states infinitely often, or equivalently, whether T_1 satisfies the the LTL formula $\bigwedge_{i=1}^r GFF_i$.

If the LTL formula is not satisfied by T_1 , then output that “the system is not prediagnosable for failures”; otherwise output that “the system is prediagnosable for failures”.

Remark 5 It is known that the first and second steps of Algorithm 1 have a complexity of $O(2^{|f|})$. The third step has a complexity of $O(2^{O(|f|)}|X_P|^2)$. This is because the complexity for the LTL model checking with the special formula $\bigwedge_{i=1}^r GFF_i$ is linear in both the size of the system (number of states and transitions) and the value of r (we can model check the formula GFF_i for each $i = 1, \dots, r$), and $r \leq |f|$. Thus, Algorithm 1 has a complexity of $O(2^{O(|f|)}|X_P|^2)$, and its correctness follows from the construction together with Theorem 2.

The following example is used for the illustration of Algorithm 1.

Example 2 Consider again the system shown in Figure 1 of Example 1 along with the specification $f = GFp$. Per the first step of Algorithm 1, we construct a generalized Büchi automaton $T_f = (C_f, \Sigma_{AP}, R_f, q_0^f, \mathcal{F})$ that accepts all infinite propositions-traces satisfying f as follows: $C_f = \{t_0, t_1, t_2\}$, $\Sigma_{AP} = \{\emptyset, \{p\}, \{g\}, \{p, g\}\}$, R_f as shown in Figure 2, $q_0^f = t_0$ is the initial state, and $\mathcal{F} = \{F_1\}$ is the generalized Büchi acceptance condition with $F_1 = \{t_1\}$. The second step of Algorithm 1 does not change T_f . By applying the third step of

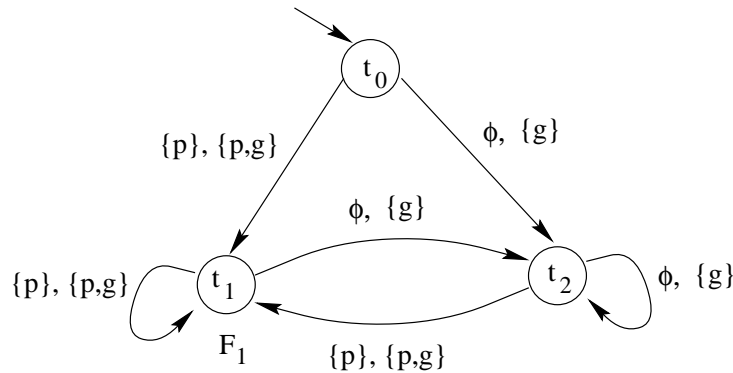


Figure 2: Büchi automaton T_f

Algorithm 1, we obtain T_1 as shown in Figure 3. One can verify that the LTL formula GFF_1 is not satisfied by T_1 since the trace $((x_0, t_1)^k(x_1, t_2)^\omega)$ does not visit the F_1 -labeled state (i.e., the state (x_0, t_1)) infinitely often. From the last step of Algorithm 1 we know that the system is not prediagnosable for failures, as illustrated in Example 1.

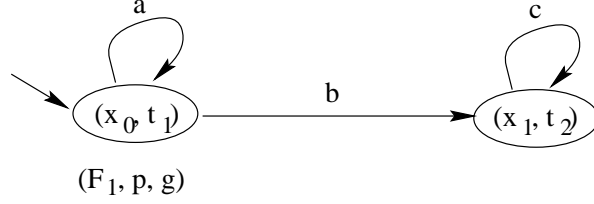


Figure 3: State machine T_1

4 Notion of Gf -Failure Point

As mentioned above, for diagnosis of repeated failures, it is natural to have a specification that must hold perpetually, regardless of the current state of the system. So, the specification can be taken to be of the form of Gf , where G stands for ‘‘Globally’’ (in all states) and f is a LTL formula. It should also be noted that there is no loss of expressibility in using a LTL formulae in the form of Gf , as any LTL formula can also be expressed in the form of Gf . To see this, if for a system $P = (X_P, \Sigma, R, X_0, AP, L)$ its specification is given as f , then we can transform this into an equivalent specification $f_{new} := G(\text{init} \Rightarrow f)$ and transform the system into a new system $P_{new} = (X_{new}, \Sigma, R_{new}, X_0^{new}, AP_{new}, L_{new})$ such that

- $X_{new} = X_P \cup X_0^{new}$,
- $R_{new} = R \cup \{(x, \text{init}), \sigma, x') \mid (x, \sigma, x') \in R\}$,
- $X_0^{new} = \{(x, \text{init}) \mid x \in X_0\}$,
- $AP_{new} = AP \cup \{\text{init}\}$,
- $\forall x \in X_P, L_{new}(x) = L(x)$, and $\forall (x, \text{init}) \in X_0^{new}, L_{new}((x, \text{init})) = L(x) \cup \{\text{init}\}$.

It is easy to verify that there is an one-to-one correspondence between the state traces generated by P and P_{new} , and for an infinite state-trace π generated by P , $\pi \models f$ if and only if $\pi_{new} \models f_{new}$, where π_{new} is the corresponding trace of π in P_{new} .

Next, in order to determine the number of failures that have occurred along a trace executed by the system, we introduce the notion of Gf -failure points.

Definition 4 Let P be a system, Gf be a LTL formula representing the specification. Then for a finite or infinite state-trace $\pi = (x_1(\pi), \dots)$ generated by P , $x_i(\pi)$, $\forall i \geq 1$, is called a Gf -failure point of π if $\exists j$ with $i \geq j \geq 1$, such that

1. $(x_j(\pi), \dots, x_i(\pi))$ is a Gf -failure state-trace;
2. $\forall k$ with $i > k \geq j$, $(x_j(\pi), \dots, x_k(\pi))$ is not a Gf -failure state-trace.

We use N_π^f to denote the number of all Gf -failure points of π .

Thus the i th state $x_i(\pi)$ of an infinite state-trace π is a Gf -failure point if there exists a state-trace segment $\pi_{ji} = (x_j(\pi), \dots, x_i(\pi))$ of π ending in $x_i(\pi)$ such that π_{ji} is a Gf -failure state-trace and no prefix of π_{ji} is a Gf -failure state-trace (π_{ji} is the shortest Gf -failure state-trace ending in $x_i(\pi)$).

Remark 6 From the semantics of Gf , we know that each Gf -failure point of a trace π indicates one violation of the specification Gf along the trace π . Thus for the diagnosis of repeated failures, we just need to diagnose the number of the occurrence of the Gf -failure points along any state-trace π executed by the system. From now on, we do not distinguish between the number of Gf -failure points and the number of failures of the specification Gf . Since now we are interested in the diagnosis of Gf -failure points, it is natural to require that every infinite Gf -failure trace in P shall contain a Gf -failure point, i.e., the system shall be Gf -failure prediagnosable. From now on, we assume this to be the case.

Remark 7 Note that one can also have a notion of Gf -indicator point by replacing “ Gf -failure state-trace” with “ Gf -indicator state-trace” in Definition 4. Since an indicator point only indicates the inevitability of a failure in future (and not its actual occurrence), there may exist multiple number of indicator points for a single failure point. This is the reason we adopt the notion of Gf -failure point and not that of Gf -indicator point. This is illustrated more concretely by the following example.

Example 3 Consider the system shown in Figure 4. Suppose the specification is given as Gp , i.e., “ p ” should hold in every state visited in any infinite trace of the system. It is

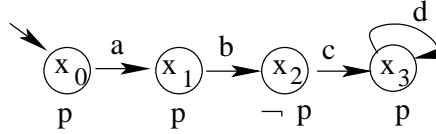


Figure 4: Example for Gf -failure point

easy to verify that in the infinite trace $\pi = (x_0, x_1, x_2, x_3^\omega)$, there is only one failure in π , namely at x_2 , which is also the only Gp -failure point of π . However, if we adopt the notion of Gf -indicator point, then we find that x_0 , x_1 , and x_2 all are Gp -indicator points of π . This is because the failure at x_2 will occur inevitably if we are at x_0 , as well as at x_1 and x_2 . The adoption of Gf -indicator state-trace is a viable alternative if we are interested in only detecting whether or not a failure has occurred, but if we must also determine each time a failure has occurred, then we must adopt the notion of Gf -failure state-trace.

In the following, we provide an algorithm for deciding the number of Gf -failure points of state-traces in the system by labeling each Gf -failure point as a faulty state. Let $P = (X_P, \Sigma, R, X_0, AP, L)$ be a non-terminating system, M be an observation mask, and $SP = \{Gf_i, m \geq i \geq 1\}$ be a set of LTL specifications, then the algorithm for identifying Gf_i -failure points in P for any $i \in \{1, \dots, m\}$ is described in the following.

The algorithm proceeds as follows. First, a tableau T_{f_i} is constructed for the formula f_i such that all the infinite propositions-traces satisfying f_i is contained in T_{f_i} . Next, a deterministic finite state machine A_i^1 is constructed from T_{f_i} , which accepts all finite propositions-traces that are not f_i -failure traces (Definition 2). In the third step, a non-deterministic finite state machine A_i^2 with a transition labeling function M^2 is constructed such that A_i^2 accepts all finite propositions-traces, but those that are not accepted by A_i^1 is identified by M^2 (such traces are f_i -failure traces). In the next step, A_i^2 is determinized to A_i^3 and the transition

labeling function M^2 is changed to a state labeling function M^3 . Any trace of A_i^3 ending at a M^3 labeled state is a f_i -failure trace. In the last step, A_i^3 is synchronized with the system P to obtain P_i . In P_i , a state is marked by ψ_i as a Gf_i -failure point if and only if its first coordinate is a M^3 labeled state of A_i^3 .

Algorithm 2 Algorithm for identifying Gf -failure points

1. This step is for the construction of a tableau T_{f_i} for formula f_i that contains all the infinite propositions-traces satisfying f_i . Following the method in [8], T_{f_i} can be constructed as $T_{f_i} = (C_i, R_i, AP_i, L_i)$, where C_i is the state set, $R_i \subseteq C_i \times C_i$ is the transition relation, $AP_i = \{p \in AP \mid p \text{ is a sub-formula of } f_i\}$ is the set of atomic propositions for T_{f_i} , and L_i is the labeling function.
2. This step is for the construction of a deterministic finite state machine A_i^1 that accepts the language $\{s \in \Sigma_{AP_i}^* \mid \exists t \in \Sigma_{AP_i}^\omega \text{ s.t. } st \models f_i\}$. $A_i^1 = (X_i^1, \Sigma_{AP_i}, R_i^1, x_{i0}^1)$ is constructed from T_{f_i} as follows:

- (a) Construct a non-deterministic finite state machine

$$NA_i = (X_i, \Sigma_{AP_i}, NR_i, x_{i0})$$

from T_{f_i} , where $X_i = C_i \cup \{x_{i0}\}$ is the state set; $\Sigma_{AP_i} = 2^{AP_i}$ is the event set; $NR_i \subseteq X_i \times \Sigma_{AP_i} \times X_i$ is the set of transitions, which is defined as:

- $\forall x_1, x_2 \in X_i - \{x_{i0}\}, \sigma \in \Sigma_{AP_i}, (x_1, \sigma, x_2) \in NR_i$ iff $(x_1, x_2) \in R_i$ and $\sigma = L_i(x_2) \cap AP_i$,
- $\forall x_1 \in X_i - \{x_{i0}\}, \sigma \in \Sigma_{AP_i}, (x_{i0}, \sigma, x_1) \in NR_i$ iff $f_i \in L_i(x_1)$ and $\sigma = L_i(x_1) \cap AP_i$;

and x_{i0} , the state added, is the initial state.

- (b) Obtain A_i^1 as the determinization of NA_i .

We require A_i^1 to be accessible, i.e., every state in A_i^1 is reachable from its initial state. If A_i^1 is not accessible, then we make it accessible by deleting those states and the associated transitions from A_i^1 that are not reachable from x_{i0}^1 .

3. This step is for the construction of a non-deterministic finite state machine $A_i^2 = (X_i^2, \Sigma_{AP_i}, R_i^2, x_{i0}^2)$ that accepts the language $\Sigma_{AP_i}^*$, and has a labeling function $M^2 : R_i^2 \rightarrow \{F_i, \emptyset\}$ for distinguishing the languages generated by A_i^1 and A_i^2 . A_i^2 and M^2 are constructed from A_i^1 as follows:

- $X_i^2 = X_i^1 \cup X_{\neg f_i}$ is the state set, where

$$X_{\neg f_i} = \{e \in \Sigma_{AP_i} \mid \nexists x \in X_i^1 \text{ s.t. } (x_{i0}^1, e, x) \in R_i^1\}.$$

- $R_i^2 = R_i^1 \cup R^1 \cup R^2 \cup R^3$ is the set of transitions, where R^1 , R^2 , and R^3 are defined as: $\forall (x_1, e, x_2) \in X_i^2 \times \Sigma_{AP_i} \times X_i^2$,
 - $(x_1, e, x_2) \in R^1$ iff $[\exists x' \text{ s.t. } (x_1, e, x') \in R_i^1] \wedge [((e \in X_{\neg f_i}) \wedge (x_2 = e)) \vee ((e \notin X_{\neg f_i}) \wedge ((x_{i0}^1, e, x_2) \in R_i^1))]$;

- $(x_1, e, x_2) \in R^2$ iff $[x_1 \in X_{\neg f_i}] \wedge [((e \in X_{\neg f_i}) \wedge (x_2 = e)) \vee ((e \notin X_{\neg f_i}) \wedge ((x_{i0}^1, e, x_2) \in R_i^1))]$;
- $(x_1, e, x_2) \in R^3$ iff $[x_1 \in X_i^1] \wedge [\nexists x' \text{ s.t. } (x_1, e, x') \in R_i^1] \wedge [((e \in X_{\neg f_i}) \wedge (x_2 = e)) \vee ((e \notin X_{\neg f_i}) \wedge ((x_{i0}^1, e, x_2) \in R_i^1))]$.
- $x_{i0}^2 = x_{i0}^1$ is the initial state.
- $M^2 : R_i^2 \rightarrow \{F_i, \emptyset\}$ is the labeling function defined as: $\forall (x_1, e, x_2) \in R_i^2$,

$$M^2(x_1, e, x_2) = \begin{cases} F_i & \text{if } [x_2 \in X_{\neg f_i}] \vee [(x_1, e, x_2) \in R^3], \\ \emptyset & \text{otherwise.} \end{cases}$$

We make A_i^2 to be accessible by removing those states and their associated transitions from A_i^2 that are not reachable from x_{i0}^2 .

4. This step is for the determinization of A_i^2 and the construction of a new labeling function M^3 defined on the set of states (instead of on the set of transitions). A deterministic finite state machine $A_i^3 = (X_i^3, \Sigma_{AP_i}, R_i^3, x_{i0}^3)$ together with the labeling function $M^3 : X_i^3 \rightarrow \{F_i, \emptyset\}$ are constructed from A_i^2 and M^2 as follows:

- Obtain $A_{2i} = (X_{2i}, \Sigma_{AP_i}, R_{2i}, x_{i0}^2)$ as the determinization of A_i^2 .
- Construct A_i^3 from A_{2i} as follows:
 - $X_i^3 = X_{2i} \cup X_{new}$ is the state set, where

$$X_{new} = \{(x, 1) \mid x \in X_{2i} \text{ s.t. } [x \cap X_{\neg f_i} = \emptyset] \wedge [\exists (x', e, x) \in R_{2i} \text{ s.t. } (\exists x'_1 \in x', x_1 \in x, M^2(x'_1, e, x_1) = F_i)]\}.$$

- R_i^3 is the set of transitions which is defined as: $\forall (x, e, x') \in X_i^3 \times \Sigma_{AP_i} \times X_i^3$, $(x, e, x') \in R_i^3$ if and only if one of the following holds:
 - * $[(x, e, x') \in R_{2i}] \wedge [\nexists x_1 \in x, x'_1 \in x' \text{ s.t. } M^2(x_1, e, x'_1) = F_i]$;
 - * $[x = (y, 1) \in X_{new}] \wedge [x' \notin X_{new}] \wedge [(y, e, x') \in R_{2i}] \wedge [\nexists x_1 \in y, x'_1 \in x' \text{ s.t. } M^2(x_1, e, x'_1) = F_i]$;
 - * $[x \notin X_{new}] \wedge [x' = (y, 1) \in X_{new}] \wedge [(x, e, y) \in R_{2i}] \wedge [\exists x_1 \in x, x'_1 \in y \text{ s.t. } M^2(x_1, e, x'_1) = F_i]$;
 - * $[x = (y, 1) \in X_{new}] \wedge [x' = (y', 1) \in X_{new}] \wedge [(y, e, y') \in R_{2i}] \wedge [\exists x_1 \in y, x'_1 \in y' \text{ s.t. } M^2(x_1, e, x'_1) = F_i]$.
- $x_{i0}^3 = x_{i0}^2$ is the initial state.
- M^3 is defined as: $\forall x \in X_i^3$, $M^3(x) = F_i$ if $[x \cap X_{\neg f_i} \neq \emptyset] \vee [x \in X_{new}]$, otherwise $M^3(x) = \emptyset$.

We make A_i^3 to be accessible by removing those states and their associated transitions from A_i^3 that are not reachable from x_{i0}^3 .

5. This step is for the construction of a finite state machine $P_i = (Y_i, \Sigma, R_i^t, Y_{i0}, F_i, \psi_i)$ that identify all the Gf_i -failure points in P , where $\psi_i : Y_i \rightarrow \{F_i, \emptyset\}$ is the fault assignment function. P_i is obtained from the synchronous composition of P and A_i^3 as follows:

- $Y_i = X_i^3 \times X_P$ is the set of states,
- $R_i^t \subseteq Y_i \times \Sigma \cup \{\epsilon\} \times Y_i$ is the set of transitions, which is defined as:
 $\forall((x_1, x_2), \sigma, (x'_1 x'_2)) \in Y_i \times \Sigma \cup \{\epsilon\} \times Y_i,$

$$((x_1, x_2), \sigma, (x'_1 x'_2)) \in R_i^t \Leftrightarrow [(x_1, L(x'_2) \cap AP_i, x'_1) \in R_i^3] \wedge [(x_2, \sigma, x'_2) \in R].$$
- $Y_{i0} = \{(x, x') \in X_i^3 \times X_0 \mid (x_{i0}^3, L(x') \cap AP_i, x) \in R_i^3\}$ is the set of initial states.
- $\psi_i : Y_i \rightarrow \{F_i, \emptyset\}$ is the labeling function, which is defined as: $\forall(x_1, x_2) \in Y_i,$
 $\psi_i(x_1, x_2) = M^3(x_1).$

We make P_i to be accessible by removing those states and their associated transitions from P_i that are not reachable from Y_{i0} .

Remark 8 It is known that the tableau T_{f_i} has at most $2^{O(|f_i|)}$ number of states; and because of the deterministic operation A_i^1 has at most $2^{2^{O(|f_i|)}}$ number of states. A_i^2 is derived from A_i^1 and it also has at most $2^{2^{O(|f_i|)}}$ number of states. A_i^3 has at most $2^{2^{2^{O(|f_i|)}}$ number of states because of another deterministic operation; and it follows directly from the construction that P_i has at most $O(2^{2^{2^{O(|f_i|)}}} \times |X_P|)$ number of states. Thus, Algorithm 2 has a worst case complexity that is triple exponential in the length of the formula f_i and linear in the number of the states of the system P . Note that in most real applications, usually the number of the system states is very large whereas the length of the formula f_i is very small. Since our algorithm has a linear complexity in the number of system states, our algorithm performs very well with respect to the number of system states. On the other hand, since the length of the specification formula is usually very small and also since in most situations the worst case complexity is not encountered, our algorithm should provide good performance in practice. Further research is needed to determine whether or not a better complexity algorithm exists.

We have the following theorem which describes some properties of the state machines constructed in Algorithm 2.

Theorem 3 Let $P = (X_P, \Sigma, R, X_0, AP, L)$ be a non-terminating system with a specification set $SP = \{Gf_i, 1 \leq i \leq m\}$. Let $A_i^1 = (X_i^1, \Sigma_{AP_i}, R_i^1, x_{i0}^1)$, $A_i^2 = (X_i^2, \Sigma_{AP_i}, R_i^2, x_{i0}^2)$ with the labeling function $M^2 : R_i^2 \rightarrow \{F_i, \emptyset\}$, $A_i^3 = (X_i^3, \Sigma_{AP_i}, R_i^3, x_{i0}^3)$ with the labeling function $M^3 : X_i^3 \rightarrow \{F_i, \emptyset\}$, and $P_i = (Y_i, \Sigma, R_i^t, Y_{i0}, F_i, \psi_i)$ be the finite state machines constructed in Algorithm 2, then we have:

1. The language generated by A_i^1 , denoted by $L_{A_i^1}$, is $\{s \in \Sigma_{AP_i}^* \mid \exists t \in \Sigma_{AP_i}^\omega \text{ s.t. } st \models f_i\}$.
2. The language generated by A_i^2 , denoted by $L_{A_i^2}$, is $\Sigma_{AP_i}^*$; and $\forall(e_1, \dots, e_n) \in \Sigma_{AP_i}^*$ with $n \geq 1$, there exists a $j \geq 1$ such that (e_j, \dots, e_n) is a f_i -failure propositions-trace if and only if there exists a path

$$(x_1 = x_{i0}^2, e_1, x_2, \dots, x_n, e_n, x_{n+1})$$

in A_i^2 (i.e., $\forall i \in \{1, \dots, n\}$, $(x_i, e_i, x_{i+1}) \in R_i^2$) with either $M^2(x_n, e_n, x_{n+1}) = F_i$ or $x_{n+1} \in X_{\neg f_i}$.

3. A_i^3 is deterministic; the language generated by A_i^3 , denoted by $L_{A_i^3}$, is $\Sigma_{AP_i}^*$; and $\forall (e_1, \dots, e_n) \in \Sigma_{AP_i}^*$ with $n \geq 1$, there exists a $j \geq 1$ such that (e_j, \dots, e_n) is a f_i -failure propositions-trace if and only if for the path

$$(x_1 = x_{i0}^3, e_1, x_2, \dots, x_n, e_n, x_{n+1})$$

in A_i^3 (i.e., $\forall i \in \{1, \dots, n\}$, $(x_i, e_i, x_{i+1}) \in R_i^3$) it holds that $M^3(x_{n+1}) = F_i$.

4. There is an one-to-one correspondence between Tr_P and Tr_{P_i} :

- $\forall \pi^t = (x_0, \dots, x_n) \in \text{Tr}_P$
 $\Rightarrow [\exists \pi = ((z_0, x_0), \dots, (z_n, x_n)) \in \text{Tr}_{P_i}] \wedge [O_\pi = O_{\pi^t}];$
- $\forall \pi = ((z_0, x_0), \dots, (z_n, x_n)) \in \text{Tr}_{P_i}$
 $\Rightarrow [\pi^t = (x_0, \dots, x_n) \in \text{Tr}_P] \wedge [O_{\pi^t} = O_\pi].$

5. $\forall \pi^t = (x_0, \dots, x_n) \in \text{Tr}_P$ and its corresponding trace $\pi = ((z_0, x_0), \dots, (z_n, x_n)) \in \text{Tr}_{P_i}$, any state x_j ($0 \leq j \leq n$) in π^t is a Gf_i -failure point if and only if $\psi_i(z_j, x_j) = F_i$.

Proof: The first and the fourth assertions follow directly from the construction of A_i^1 and P_i . The third assertion follows directly from the construction of A_i^3 and the second assertion. The last assertion follows directly from the construction of P_i and the third assertion. So we only need to prove the second assertion.

For the second assertion, from the construction of A_i^2 , it is easy to verify that A_i^2 accepts the language $\Sigma_{AP_i}^*$. In the following, we prove that $\forall (e_1, \dots, e_n) \in \Sigma_{AP_i}^*$ with $n \geq 1$, there exists a $j \geq 1$ such that (e_j, \dots, e_n) is a f_i -failure propositions-trace if and only if there exists a path

$$(x_1 = x_{i0}^2, e_1, x_2, \dots, x_n, e_n, x_{n+1})$$

in A_i^2 with either $M^2(x_n, e_n, x_{n+1}) = F_i$ or $x_{n+1} \in X_{\neg f_i}$.

For the sufficiency of the above assertion, suppose there exists a path

$$(x_1 = x_{i0}^2, e_1, x_2, \dots, x_n, e_n, x_{n+1})$$

in A_i^2 with either $M^2(x_n, e_n, x_{n+1}) = F_i$ or $x_{n+1} \in X_{\neg f_i}$. If $x_{n+1} \in X_{\neg f_i}$ then $e_n = x_{n+1}$. From the definition of $X_{\neg f_i}$ we know that (e_n) is a f_i -failure propositions-trace. If $x_{n+1} \notin X_{\neg f_i}$ but $M^2(x_n, e_n, x_{n+1}) = F_i$ then we must have that $(x_n, e_n, x_{n+1}) \in R_3$, which implies that $x_n \in X_i^1$. From the construction of A_i^2 we know that x_n must be reached through a path (x_j, e_j, \dots, x_n) entirely contained in A_i^1 , and $(x_{i0}^1, e_{j-1}, x_j) \in R_i^1$. It further implies that $(e_{j-1}, \dots, e_{n-1}) \in L_{A_i^1}$ but $(e_{j-1}, \dots, e_n) \notin L_{A_i^1}$, i.e., (e_{j-1}, \dots, e_n) is a f_i -failure propositions-trace. So the sufficiency of the above assertion holds.

For the necessity of the above assertion, suppose there exists a $j \geq 1$ such that (e_j, \dots, e_n) is a f_i -failure propositions-trace. Let (e_j, \dots, e_n) be the f_i -failure propositions-trace such that no (e_k, \dots, e_n) with $k > j$ is a f_i -failure propositions-trace. Then we have that $(e_j, \dots, e_{n-1}) \in L_{A_i^1}$ but $(e_j, \dots, e_n) \notin L_{A_i^1}$. We have two cases here.

1. If $j = n$ then we must have that $e_n \in X_{-f_i}$. Since A_i^2 accepts the language $\Sigma_{AP_i}^*$, we know that there exists a path $(x_1 = x_{i0}^2, e_1, x_2, \dots, e_{n-1}, x_n)$ in A_i^2 . From the construction of A_i^2 , we know that the path

$$(x_1 = x_{i0}^2, e_1, x_2, \dots, e_{n-1}, x_n, e_n, x_{n+1} = e_n)$$

is also contained in A_i^2 , here $x_{n+1} \in X_{-f_i}$.

2. If $j < n$ then there must exist a path $(x_{i0}^1, e_j, x_{j+1}, \dots, e_{n-1}, x_n)$ in A_i^1 and no transition in the form of (x_n, e_n, x') exists in R_i^1 . From the construction of A_i^2 , we know that there exists a path

$$(x_1 = x_{i0}^2, e_1, \dots, x_j, e_j, x_{j+1}, \dots, e_{n-1}, x_n, e_n, x_{n+1})$$

in A_i^2 . Since $(x_n, e_n, x_{n+1}) \notin R_i^1$, we have that $M^2(x_n, e_n, x_{n+1}) = F_i$.

From the above, we know that the necessity of the assertion also holds. This completes the proof of the second assertion. \blacksquare

From Theorem 3, we know that every Gf_i -failure point in P is identified in P_i as a state labeled with F_i . Thus for the purpose of repeated failure diagnosis, if the system P is Gf_i -failure prediagnosable, then we just need to diagnose each visit to a F_i -labeled state in P_i .

5 Diagnosis for Repeated Failures

In this section, we give definitions of various notions of diagnosability for diagnosing the multiplicity of the occurrence of failures in the temporal logic setting adapted from the corresponding definitions given in the state-based setting in [12], and show that the diagnosis for repeated failures in the temporal logic setting can be transformed to the diagnosis in the state-based setting which was studied in [12].

The definition of diagnosability for repeated failures in the temporal logic setting is given below.

Definition 5 Given a system P , an observation mask M , and a set of specifications $SP = \{Gf_i, m \geq i \geq 1\}$, P is said to be K -diagnosable ($K \geq 1$), (resp., $[1, K]$ -diagnosable, or $[1, \infty]$ -diagnosable) with respect to M and SP if P is Gf_i -failure prediagnosable for each Gf_i and

$$\begin{aligned} & (\forall Gf_i \in SP)(\exists n_i \in \mathcal{N}) \\ & (\forall J = K)((\text{resp.}, (\forall J, 1 \leq J \leq K), \text{ or } (\forall J \geq 1))) \\ & (\forall \pi_0 \in \text{Tr}_P, N_{\pi_0}^{f_i} \geq J) \\ & (\forall \pi = \pi_0 \pi_1 \in \text{Tr}_P, |\pi_1| \geq n_i) \\ & (\forall \pi' \in \text{Tr}_P, O_\pi \cap O_{\pi'} \neq \emptyset) \\ & \Rightarrow (N_{\pi'}^{f_i} \geq J). \end{aligned}$$

From Algorithm 2, we know that each Gf_i -failure point can be identified by a F_i labeled state. This suggests that the repeated diagnosis in the temporal logic setting can be transformed to that in the state-based setting. In the following, we show that the above definitions of diagnosability for repeated failures in the temporal logic setting are equivalent to those defined in the state-based setting in [12].

Given a system $P = (X_P, \Sigma, R, X_0, AP, L)$ with an observation mask M and a specification set $SP = \{Gf_i, 1 \leq i \leq m\}$, from Algorithm 2 we can obtain a system $P_i = (Y_i, \Sigma, R_i^t, Y_{i0}, F_i, \psi_i)$ for each $i = 1, \dots, m$. From Theorem 3, and the definitions of diagnosability in temporal logic and state-based settings, the following result holds obviously.

Theorem 4 If P is Gf_i -failure prediagnosable for each Gf_i in SP , then P is K -diagnosable (resp., $[1, K]$ - or $[1, \infty]$ -diagnosable) with respect to M and SP if and only if $\forall i \in \{1, \dots, m\}$, P_i is K -diagnosable (resp., $[1, K]$ - or $[1, \infty]$ -diagnosable) with respect to M and ψ_i .

Theorem 4 solves the problem of repeated failure diagnosis in the temporal logic setting by reducing it to the corresponding problem in the state based setting. Conversely, given a system $P = (X_P, \Sigma, R, X_0, \mathcal{F}, \psi)$ with an observation mask M , we can obtain another system $P' = (X_P, \Sigma, R, X_0, AP, L)$ with $AP = \mathcal{F}$, $L(x) = \psi(x)$ for all $x \in X_P$, and a specification set $SP = \{G\neg F_i, 1 \leq i \leq m\}$. It is easy to verify that P' and P accept a same set of state-traces, i.e., $\text{Tr}_{P'} = \text{Tr}_P$, and for any state-trace $\pi = (x_0, x_1, \dots)$ generated by P' , x_i ($i \geq 0$) is a $G\neg F_i$ -failure point in π if and only if $F_i \in L(x_i) = \psi(x_i)$. It is obvious that the above P' is $G\neg F_i$ -failure prediagnosable for each $G\neg F_i$ in SP , since every $G\neg F_i$ in SP is a safety property. From the definitions of diagnosability in temporal logic setting and in state-based setting, the following result follows.

Theorem 5 P is K -diagnosable (resp., $[1, K]$ - or $[1, \infty]$ -diagnosable) with respect to M and ψ if and only if P' is K -diagnosable (resp., $[1, K]$ - or $[1, \infty]$ -diagnosable) with respect to M and SP .

Remark 9 Theorems 5 and 4 establish the equivalence of notions of diagnosability for repeated failures in the state-based setting and those in the temporal logic setting. Further from Theorem 4 we know that once the system passes the test of prediagnosability for failures developed above, then we can perform the diagnosis for repeated failures in the state-based setting, and the results of [12] can be directly applied.

6 Illustrative Example

In this section, we present a simple example to illustrate the concepts and algorithms developed in this paper. Consider a traffic monitoring problem of a mouse in a maze. The maze, shown in Figure 5, consists of four rooms connected by various one-way passages, and some of them have sensors installed to detect the motion of the mouse through them. The mouse is initially in room 0, and it can visit other rooms by using the one way passages, and it never stays in one room forever. A failure is said to have occurred if the mouse make a cycle between rooms 0 and 2 without visiting any other room, i.e., the sub-sequences (room 0, room 2, room 0) and (room 2, room 0, room 2) are illegal. The task of repeated failure

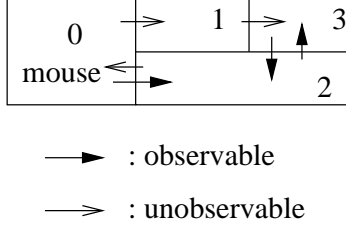


Figure 5: Mouse in a maze: repeated monitoring

diagnosis is to monitor the motion of the mouse by observing the sensor signals, and to detect the number of times the failure has occurred.

The above problem can be viewed as one of $[1, \infty]$ -diagnosis in the temporal logic setting, where the system $P = (X_P, \Sigma, R, x_0, AP, L)$ to be diagnosed is as shown in Figure 6. Here

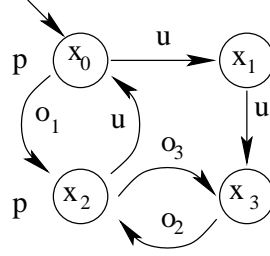


Figure 6: System model

$X_P = \{x_i, 0 \leq i \leq 3\}$, $\Sigma = \{o_1, o_2, o_3, u\}$, $\Delta = \{o_1, o_2, o_3\}$, the mask M is given as $M(u) = \epsilon$ and $M(o_i) = o_i$ for $i = 1, 2, 3$, $AP = \{p\}$, and the labeling function L is given as $L(x_1) = L(x_3) = \emptyset$ and $L(x_0) = L(x_2) = \{p\}$. The specification is represented by $Gf_1 = G(\neg p \vee X\neg p \vee XX\neg p)$.

It is obvious that the above system is trivially Gf_1 -failure prediagnosable, since Gf_1 is a safety property. Next, we reduce the above problem in the temporal logic setting to the one in a state-based setting by using Algorithm 2 to identify those Gf_1 -failure points in P .

The tableau T_{f_1} for f_1 is given in Figure 7.

The deterministic state machine A_1^1 , obtained from T_{f_1} , is as shown in Figure 8.

The state machine A_1^2 , derived from A_1^1 , is as shown in Figure 9, where only the transition (u_3, p, u_1) is labeled with F_1 by the labeling function M^2 , i.e., $M^2(u_3, p, u_1) = F_1$.

The state machine A_1^3 , derived from A_1^2 , is as shown in Figure 10, where only states w_3 and w_8 are labeled with F_1 by the labeling function M^3 .

The state machine P_1 , derived from the synchronous composition of A_1^3 and P , is as shown in Figure 11, where only states y_3, y_4, y_{10} , and y_{11} are labeled with F_1 by the labeling function ψ_1 .

Now the problem can be expressed as a $[1, \infty]$ -diagnosis problem in the state-based setting as follows. The system is represented by P_1 ; the set of failure types is $\mathcal{F} = \{F_1\}$; the failure assignment function is ψ_1 ; the observation mask is M .

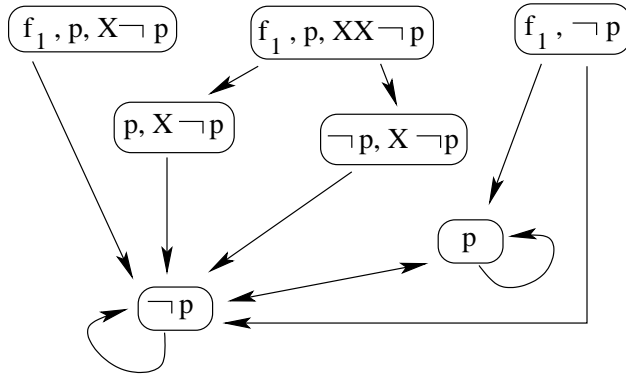


Figure 7: Tableau for f_1

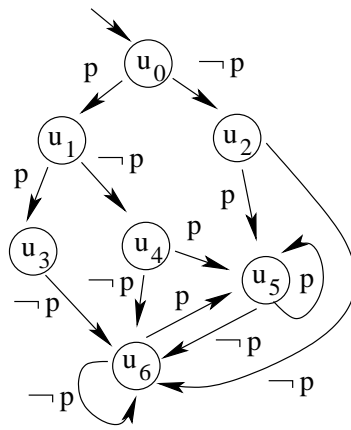


Figure 8: Diagram of A_1^1

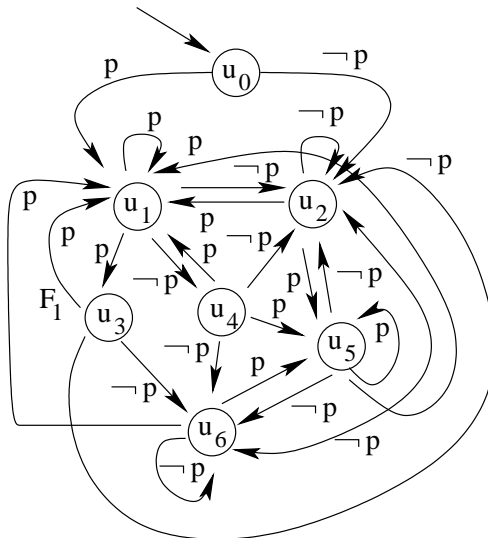


Figure 9: Diagram of A_1^2

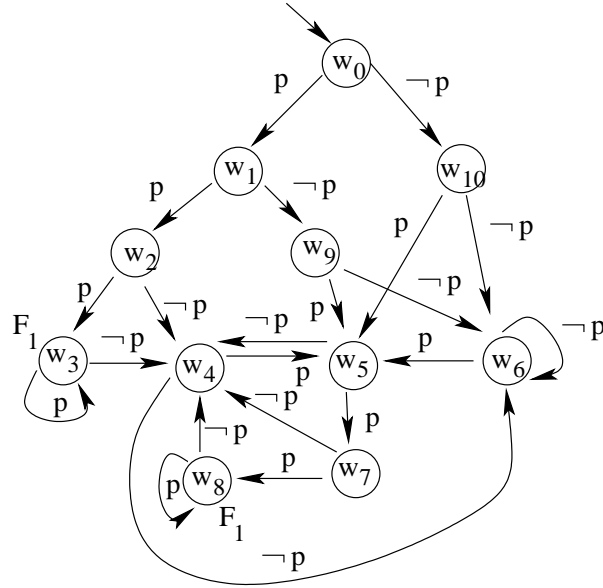


Figure 10: Diagram of A_1^3

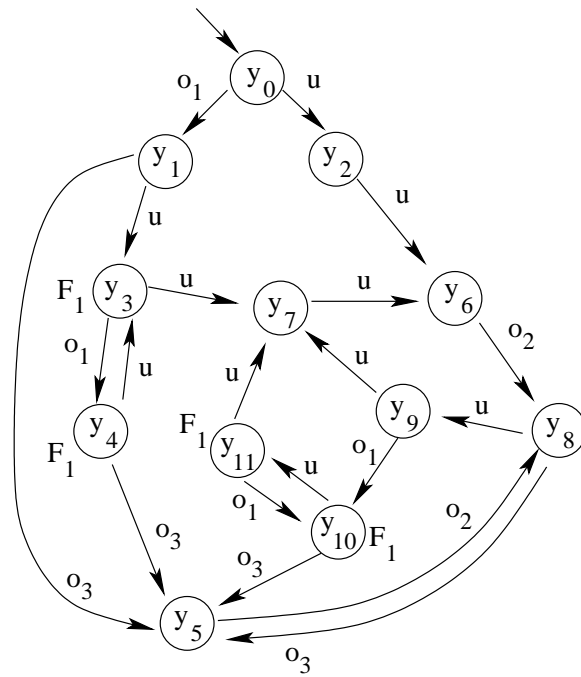


Figure 11: Diagram of P_1

By applying the results in [12] for the above state-based repeated diagnosis problem (details are omitted), we can find that P_1 is $[1, \infty]$ -diagnosable with respect to M and ψ_1 ; and an off-line diagnoser for the example can be derived as shown in Figure 12. The diagnoser

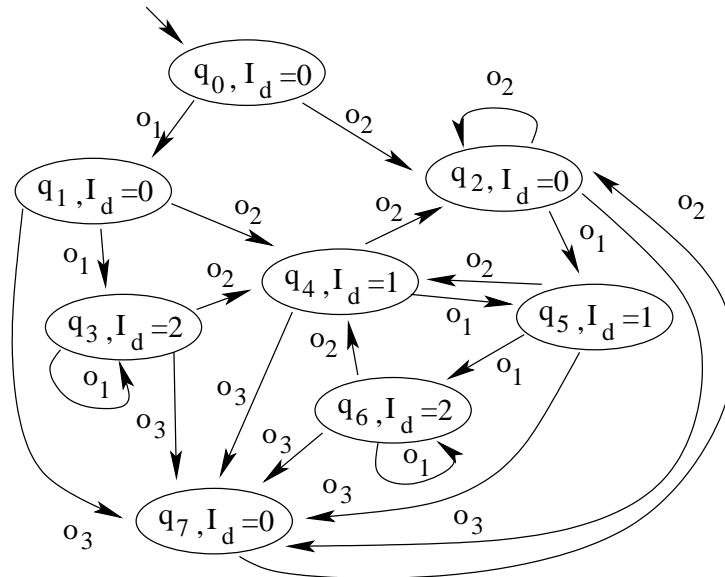


Figure 12: Off-line diagnoser

reports the detection of I_d new failures each time it reaches a state with $I_d > 0$.

7 Conclusion

In this paper, the problem of repeated failure diagnosis for discrete event systems with LTL specifications is studied. Notions of prediagnosability for failures and repeated diagnosability in the temporal logic setting are defined. Algorithms for the test of prediagnosability for failures and for the transformation of repeated failure diagnosis in the temporal logic setting to that in the state-based setting are provided. An illustrative example is also given.

Note that we have solved the problem of repeated failure diagnosis in the temporal logic setting by reducing it to the one in the state-based setting. As a future research direction, one may want to obtain a method that performs the diagnosis directly in the temporal logic setting. Also in practice, after a failure is detected, the system should be reconfigured. Thus the failure-adaptive control of DESs is also a future research topic.

References

- [1] A. Benveniste, E. Fabre, S. Haar, and C. Jard. Diagnosis of asynchronous discrete-event systems: a net unfolding approach. *IEEE Transactions on Automatic Control*, 48(5):714–727, May 2003.

- [2] R. K. Boel and J. H. van Schuppen. Decentralized failure diagnosis for discrete-event systems with constrained communication between diagnosers. In *Proceedings of International Workshop on Discrete Event Systems*, 2002.
- [3] E. M. Clarke and E. A. Emerson. Automatic verification of finite-state concurrent systems using temporal logic. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.
- [4] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.
- [5] O. Contant, S. Lafortune, and D. Teneketzis. Diagnosis of intermittent faults. *Discrete Event Dynamical Systems: Theory and Application*, 14:171–202, 2004.
- [6] S. R. Das and L. E. Holloway. Characterizing a confidence space for discrete event timings for fault monitoring using discrete sensing and actuation signals. *IEEE Transactions on Systems, Man, and Cybernetics—Part A: Systems and Humans*, 30(1):52–66, 2000.
- [7] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamical Systems: Theory and Applications*, 10:33–79, 2000.
- [8] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers, 1990.
- [9] L. E. Holloway and S. Chand. Distributed fault monitoring in manufacturing systems using concurrent discrete-event observations. *Integrated Computer-Aided Engineering*, 3(4):244–254, 1996.
- [10] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial time algorithm for diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46(8):1318–1321, 2001.
- [11] S. Jiang and R. Kumar. Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications. *IEEE Transactions on Automatic Control*, 49(6):934–945, 2004.
- [12] S. Jiang, R. Kumar, and H. E. Garcia. Diagnosis of repeated/intermittent failures in discrete event systems. *IEEE Transactions on Robotics and Automation*, 19(2):310–323, 2003.
- [13] R. Kumar and V. K. Garg. *Modeling and Control of Logical Discrete Event Systems*. Kluwer Academic Publishers, Boston, MA, 1995.
- [14] C. Kuo and H. Huang. Failure modeling and process monitoring for flexible manufacturing systems using colored timed petri nets. *IEEE Transactions on Robotics and Automation*, 16(3):301–312, June 2000.

- [15] M. Larsson. *Behavioral and structural model based approaches to discrete diagnosis*. PhD thesis, Linköping University, Linköping, Sweden, 1999.
- [16] F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems: Theory and Applications*, 4(1):197–212, 1994.
- [17] D. Pandalai and L. Holloway. Template languages for fault monitoring of timed discrete event processes. *IEEE Transactions on Automatic Control*, 45(5):868–882, May 2000.
- [18] L. Portinale. Behavioral petri nets: a model for diagnostic knowledge representation and reasoning. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 27(2):184–195, April 1997.
- [19] W. Qiu and R. Kumar. Decentralized failure diagnosis of discrete event systems. In *Proceedings of 2004 International Workshop on Discrete Event Systems*, Reims, France, September 2004.
- [20] S. L. Ricker and J. H. van Schuppen. Decentralized failure diagnosis with asynchronous communication between supervisors. In *Proceedings of the European Control Conference*, pages 1002–1006, 2001.
- [21] M. Sampath and S. Lafortune. Active diagnosis of discrete event systems. *IEEE Transactions on Automatic Control*, 43(7):908–929, 1998.
- [22] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.
- [23] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems Technology*, 4(2):105–124, March 1996.
- [24] R. Sengupta and S. Tripakis. Decentralized diagnosis of regular language is undecidable. In *Proceedings of IEEE Conference on Decision and Control*, pages 423–428, Las Vegas, NV, December 2002.
- [25] R. Su, W. M. Wonham, J. Kurien, and X. Koutsoukos. Distributed diagnosis for qualitative systems. In *Proceedings of International Workshop on Discrete Event Systems*, 2002.
- [26] T. Yoo and H. E. Garcia. Event diagnosis of discrete-event systems with uniformly and nonuniformly bounded diagnosis delays. In *Proceedings of 2004 American Control Conference*, pages 5102–5107, Boston, MA, June 2004.
- [27] T. S. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially observed discrete-event systems. *IEEE Transactions on Automatic Control*, 47(9):1491–1495, 2002.
- [28] S. H. Zad. *Fault diagnosis in discrete-event and hybrid systems*. PhD thesis, University of Toronto, Toronto, Canada, 1999.