Recoverability of Faulty Discrete Event Systems

Shaolong Shu
School of Electronics and Information Engineering
Tongji University, Shanghai, China
Shushaolong@tongji.edu.cn

Wenhao Zong
School of Electronics and Information Engineering
Tongji University, Shanghai, China
Wenhaozong@yahoo.com.cn

Abstract—Here we discussed recoverability of faulty discrete event systems. A faulty discrete event system includes the healthy dynamics and the faulty dynamics. In faulty modes, the performance of the system often degrades. Hence we need to execute some recovery actions to recover the performance of the system. Recovery actions may be renewing/reparing the faulty device or re-configuring the healthy devices. Generally recovery actions can be fired not for all the time, but for some time. They were modeled as a mapping from the state set of the faulty mode to a binary set in which 1 means recovery actions can be fired and 0 means recovery actions cannot be fired. Then we defined two types of recoverabilities: recoverability and weak recoverability. Recoverability means we periodically have chances to do recovery actions and weak recoverability means we have at least one chance to do recovery actions when the system has a fault. With observer technology, we proposed criteria to check these two types of recoverabilities.

Keywords—Faulty Discrete Event Systems, Supervisory Control, Fault-Tolerant Control, Recoverability.

I. INTRODUCTION

Nowadays high dependability of complex engineering systems has become a dominant goal in industry. As a consequence, the handling of faults plays an increasingly important role in modern technology. These engineering systems such as manufacturing systems, communication protocols and reactive software are typically governed by operational rules that can be modeled by discrete event systems[1-3]. Investigating fault-tolerant supervisory control problem with discrete event system framework is very meaningful.

Until now, there has been some prior work on fault-tolerant control of discrete event systems. Some approaches for sensor faults involved controller switching upon the occurrence of a fault as in [4], or re-computation of a controller as in [5]. Case studies involving synthesis of fault-tolerant supervisors can be found in [6-8]. Design of certain coordination protocols for automated highway systems to achieve fault-tolerance under vehicle failures is reported in [9]. Takai et al. considered the problem of reliable decentralized supervisory control[10], where they studied fault-tolerance with respect to the failures of the supervisors. Fault-tolerance in Petri Net was considered in [11], where liveness enforcing strategies were designed to deal with failures using system reconfigurations. Paoli and Lafortune[12-13] introduced a fault-tolerant framework for supervisory control of discrete event systems. Safe diagnosability was introduced[12] which is a necessary step in order to achieve fault tolerant supervision of discrete event systems and then safe controllability[14], which represents the capability, after the occurrence of a fault, of steering the system away from forbidden zones was used to investigate fault tolerant supervision of discrete event systems. The approach is an active approach in which the supervisor actively reacts to the detection of a malfunctioning component in order to meet eventually degraded control specifications. The control is done based on the diagnoser which is exponential with the given systems. Wen et al.[14] investigated fault-tolerant supervisory control for fault recovery with complete event observations. A supervisor is used not only to satisfy certain control specifications but also to ensure recovery following any fault. A necessary and sufficient condition was provided for the existence of such a supervisor.

Generally speaking, for faulty discrete event systems, safety should be ensured for any time. If a faulty system is not safe, we have no way to ensure its performance. Our previous paper discussed fault-tolerant control for safety[15]. Safety was defined the ability to prevent any faulty mode from running into some illegal (that is, unsafe) states without reducing the dynamics of the healthy mode of the system. And we proposed algorithms to find fault-tolerant control policy for safety. However, safety is not enough for a faulty discrete event system. If the system always stays in faulty modes, then the best result is often to run with a degraded performance. This is not what we want. We hope the system can execute the required performance. Hence one more property for faulty discrete event systems should be that when a fault occurs the system can recover the required performance with a bounded delay via some recovery actions such as repairing or reconfiguration.

Here we discussed recoverability of faulty discrete event systems. A faulty discrete event system includes a healthy mode and multiple faulty modes. In faulty modes, we execute some recovery actions to recover the performance of the system. Recovery actions may be renewing/reparing the faulty device or re-configuring the healthy devices. Generally recovery actions can be fired not for all the time, but for some time. Hence they were modeled as a mapping from the state set of the faulty mode to a binary set in which 1 means recovery actions can be fired and 0 means recovery actions cannot be fired. Then recoverability was defined that we periodically have chances to do recovery actions. We proposed criteria to check it. In some cases, recoverability may be strong. Hence
we discussed a weaker version of recoverability which required that we at least have one chance to do recovery action when there is a fault.

Recoverability has also been investigated in [14] where recoverability is defined as the ability that the system runs to equivalent states in the faulty mode with a bounded delay after the system has a fault. Also the paper assumes that all the events are observable. In practical systems, recovery often is implemented by repairing or reconfiguration and some events are unobservable. Hence we need to consider recoverability more generally. In fact, if we define the equivalent states in [14] as recoverable states, then the recovery problem in [14] can be solved in our framework.

The paper is organized as follows. Section 2 modeled faulty discrete event systems. Section 3 investigated recoverability and weak recoverability.

II. FAULTY DISCRETE EVENT SYSTEMS

A. Modeling faults

To model faults in a discrete event system, we assume that the system has a healthy mode and \( k \) faulty modes. The healthy mode is modeled by automaton \( G_0 \):

\[ G_0 = (Q_0, \Sigma_0, \delta_0, q_0). \]

Where \( Q_0 \) is the set of discrete states; \( \Sigma_0 \) is the set of discrete events; \( \delta_0 : Q_0 \times \Sigma_0 \rightarrow Q_0 \) is the transition function and \( q_0 \in Q_0 \) is the initial state.

The \( i \)th faulty mode is modeled as:

\[ G_i = (Q_i, \Sigma_i, \delta_i) \quad i = 1, 2, \ldots, k \]

Note that the initial state of \( G_i \) is not specified because it depends on at which state in \( G_0 \) the fault occurs.

We assume that the fault can only occur in healthy mode \( G_0 \) (that is, no multiple faults). We use \( f_j \) to denote the fault that takes the system from the healthy mode \( G_0 \) to faulty mode \( G_i \). To simplify the notation, we also use \( f_j \) to denote the mapping

\[ f_j : Q_0 \rightarrow Q_i \quad i = 1, 2, \ldots, k \]

which describe the state transition when fault \( f_j \) occurs. In other words, let:

\[ Q_0 = \{ q_{1,0}, q_{2,0}, \ldots, q_{\mid \Sigma_0 \mid, 0} \} \]

\[ Q_j = \{ q_{1,j}, q_{2,j}, \ldots, q_{\mid \Sigma_0 \mid, j} \} \]

Then \( f_j(q_{1,0}) = q_{m_j} \) means that if fault \( f_j \) occurs at state \( q_{1,0} \), then the system moves to \( G_j \) and the next state is \( q_{m_j} \).

Denote the set of all the faults as:

\[ FT = \{ f_j : i = 1, \ldots, k \}. \]

Then a faulty discrete event system is denoted as:

\[ H = (G_0, G_1, \ldots, G_k, FT). \]

A faulty discrete event system runs as follows: Initially the system stays in the healthy mode \( G_0 \). The initial state is \( q_{0,0} \). (Note that in a faulty discrete event system, we need use two labels to describe a state like \( q_{i,j} \) of which \( j \) means that it belongs to mode \( G_j \) and \( k \) means the position in mode \( G_j \).)

When some event occurs, the system will be driven to a new state \( q_{i,0} \). At some time point \( q_{i,0} \), fault \( f_i \) occurs, then the system runs into faulty mode \( G_j \). The current state \( q_{i,j} \) is determined by \( f_i(q_{i,0}) \).

While it is convenient and intuitive to model a faulty discrete event system by identifying each faulty mode as described above, we can also view the system as one large extended automaton as:

\[ G_{ew} = Ac(Q_{ew}, \Sigma_{ew}, \delta_{ew}, q_{0}). \]

Where

\[ Q_{ew} = Q_0 \cup Q_1 \cup \cdots \cup Q_k, \]

\[ \Sigma_{ew} = \Sigma_1 \cup \Sigma_2 \cup \cdots \cup \Sigma_k \cup \{ f_i : i = 1, 2, \ldots, k \}, \]

\[ \delta_{ew} = \delta_1 \cup \delta_2 \cup \cdots \cup \delta_k \cup \{ (q_{i,0}, f_i, q_{m}) : f_i(q_{i,0}) = q_{m} \}. \]

B. Recovery Actions

Generally running in a faulty mode degrades the performance of the system, even worse, may cause the system to generate more serious failures. In this sense, it is very important to drive the system out of the faulty mode via recovery actions. Recovery actions may be renewing/repairing the faulty device or re-configuring the healthy devices. Let us define the recovery actions as:

\[ \Sigma_r = \{ r_1, r_2, \ldots, r_1, \ldots, r_k \}. \]

Where \( r_i \) is the corresponding recovery action of fault \( f_i \). For a fault, we may have more than one recovery action. Here for simplification, we assume there is one recovery action for every fault.

Practically, recovery actions cannot be fired for any time, only for some time. For example, a disconnected power line can be renewed only when it is idle. Specifically, we assume recovery action is state-dependent. Then state set \( Q(i = 1, 2, \ldots, k) \) is divided into two disjoint parts: recoverable state set \( Q_{rw} \) and unrecoverable state set \( Q_{w}. \) In \( Q_{rw} \), recovery action \( r_i \) can be fired. In \( Q_{w} \), recovery action \( r_i \) cannot be fired. We use a mapping to describe it as:

\[ R_i : Q_i \rightarrow \{ 0, 1 \} \quad i = 1, 2, \ldots, k. \]

And

\[ R_i(q_{hi}) = \begin{cases} 1 & \text{if } q_{hi} \in Q_{rw}, \\ 0 & \text{if } q_{hi} \in Q_{w}. \end{cases} \]

If \( R_i(q_{hi}) = 1 \), we call \( q_{hi} \) recoverable state. In practical systems, firing recovery actions(such as renewing a device) in
the healthy mode is possible, but it is not rational and we do not do it. Hence we assume recovery events can occur only in faulty modes.

Executing recovery actions is required by the supervisor. That is, recovery actions are enforced events. In a system with partial event observation, when the supervisor observe an observable string $s \in L(G_{\text{out}})$, then what it knows is the current state estimate. The feasible condition for enforcing the occurrences of recovery action $r_i$ is all the states in the current state estimate are recoverable states, that is, belong to $Q_{r,i}$.

For every mode $G_i$ ($i = 0,1,2,\ldots,k$), the event set $\Sigma_i$ can be divided into two parts: observable event subset $\Sigma_{oi}$ and unobservable event subset $\Sigma_{ui}$. Partial event observation is described a mapping from event set to observable event set for every mode $G_i$ ($i = 0,1,2,\ldots,k$):

$$P_i(e) = \epsilon, \quad P_i(s) = \begin{cases} P_i(s)\sigma & \text{if } \sigma \in \Sigma_{oi} \\ P_i(s) & \text{if } \sigma \notin \Sigma_{oi} \end{cases}$$

In practical systems, sometimes we can observe the occurrence of a fault via some symptoms. Then the fault is observable. We divide the faults set into two parts: observable and unobservable part,

$$FT = FT_o \cup FT_u.$$ Then $\Sigma_{\text{obs}} = \Sigma_{oi} \cup \Sigma_{ui} \cup \cdot \cup \Sigma_{ui} \cup FT_o$. For the extended automaton $G_{\text{ned}}$, we have:

$$P_{\text{ned}}(e) = \epsilon, \quad P_{\text{ned}}(s) = \begin{cases} P_{\text{ned}}(s)\sigma & \text{if } \sigma \in \Sigma_{\text{ned}} \\ P_{\text{ned}}(s) & \text{if } \sigma \notin \Sigma_{\text{ned}} \end{cases}$$

Let us define the faulty language for fault $f_i$ as:

$$L_f = \{s : L(G_{\text{ned}}) : f_i \in s\}.$$

For a given $s \in L_f$, the set of all the indistinguishable strings is:

$$L_{\text{ind}}(s) = \{s' : \delta_{\text{ned}}(q_0,s') \wedge P_{\text{ned}}(s) = P_{\text{ned}}(s')\}.$$

Feasibility of executing recovery actions can be defined as:

**Definition 1 (Feasibility)**

For a given string $s \in L_f$, enforcing the occurrence of recovery action $r_i$ is feasible if all the indistinguishable strings can enforce the occurrence of recovery action $r_i$. That is:

$$(\forall s' \in L_{\text{ind}}(s))R_i(\delta_{\text{ned}}(q_0,s')) = 1.$$**Example 1**

The given faulty discrete event system is shown in figure 1(a). We assume event $\alpha$ is observable and event $\beta$ is unobservable. Initially the system stays in state $q_{1,0}$. There is one fault $f_1$ and there is one recovery action $r$ in faulty state $q_{1,1}$. We assume the fault is observable.

After string $\alpha f_1$, the system comes to state $q_{2,1}$. Because event $\beta$ is unobservable, the current state estimate is $\{q_{1,1},q_{2,1}\}$. The supervisor cannot enforce the occurrence of recovery action $r$ because $q_{2,1}$ is not a recoverable state. Hence string $\alpha f_1$ is not feasible.

![Fig. 1. Faulty discrete event systems](image)

**III. RECOVERABILITY**

**A. Definitions**

In order to investigate recoverability, let us take a look back at example 1. We assume that all the events and the fault are observable. When fault $f_1$ occurs, the system runs to state $q_{2,1}$ in the faulty mode. Event $\beta$ will drive the system to state $q_{1,1}$ from state $q_{2,1}$. Then we can enforce the occurrence of recovery action $r$. If we miss to fire recovery action $r$, we still have chance to enforce the occurrence of the recovery action $r$. With the occurrence of string $\alpha f_1$, the system will comes to $q_{1,1}$. Intuitively, the system periodically visits state $q_{1,1}$. It means we periodically have chance to do recovery action $r$. Intuitively such system is recoverable. However, let us change the system in figure 1 as in the following example. We will see the system is not always recoverable.

**Example 2**

The faulty discrete event system is shown in figure 1(b). We assume both the two events are observable. Initially the system stays in state $q_{1,0}$. There is one fault $f_1$ and there is one recovery action $r$ at faulty state $q_{1,1}$. We assume the fault is observable.

When the system enters the faulty mode, it can always stay in state $q_{2,1}$ with event sequence $\alpha \alpha \alpha \alpha \cdots$. In such case, we have no way to force the system to do recovery action. Naturally the system is not recoverable.

Before we give out the definition of recoverability, let us introduce some necessary notations. We use $\epsilon$ to denote the empty string, $\Sigma^*$ to denote the set of all finite strings
constructed by concatenation of elements of \( \Sigma \). Given strings \( s \), \( s_1 \), \( s_2 \) such that \( s = s_1 s_2 \), then \( s_1 \) is a prefix of \( s \). Note that \( s \) is a prefix of itself. For a string \( s (\in \Sigma^* ) \), denote the set of all its prefixes by \( \text{Pr}(s) \). Let us also denote the set of natural numbers by \( N \).

Many practical systems only have partial observation and then enforcing occurrence of recovery actions need the support of feasibility. We have the following definition for recoverability:

**Definition 2 (recoverability)**

For a given faulty discrete event system with partial observation, we say it is recoverable if whenever the system has a fault \( f_i \), we periodically have chances to force the system to execute recovery action \( r_i \). That is,

\[
(\forall s \in L_j)(\forall t \in \Sigma_j)(\exists n \in N)[|t| > n \land \delta_{\text{cmd}}(q_0, st)] = 1
\]

In order to better investigate recoverability, as in [17-19], we make the following two assumptions to avoid triviality.

**Assumption 1**

Any mode \( G_i \), \( i = 0, 1, 2, \ldots, k \) is deadlock free, that is, for any state of mode \( G_i \), at least one event is defined at that state:

\[
(\forall q \in Q_i)(\exists \sigma \in \Sigma_i)\delta(q, \sigma) = 1
\]

**Assumption 2**

No loops in any mode \( G_i \), \( i = 0, 1, 2, \ldots, k \) contain only unobservable events:

\[-(\exists q \in Q_i)(\exists s \in \Sigma_{\text{cmd}})s \neq \varepsilon \land q \in \delta_i(q, s)\]

**B. Criteria**

Now let us discuss how to check recoverability. The necessary condition is that the system is of diagnosability. That is,

\[
(\forall s \in L_j)(\forall t \in \Sigma_j)(\exists n \in N)[|t| > n \land \delta_{\text{cmd}}(q_0, st)] = 1
\]

\[
(\forall s' \in L_{\text{cmd}}(st))s' \in L_j, \quad i = 1, 2, \ldots, k
\]

If the system is not diagnosable, then for some faulty trajectories, we cannot determine the system has a fault \( f_i \). Naturally we cannot do recovery action. We assume the system is diagnosable.

In order to investigate the relation between the state estimates and the observed event sequences, we construct the observer for the given faulty discrete event system. As discussed in [16-18], the observer can be constructed by first replacing all unobservable events in \( G_{\text{cmd}} \) by the empty string \( \varepsilon \) and then converting the nondeterministic automaton with \( \varepsilon \) transitions into a deterministic automaton in the usual way. Denote the observer of \( G_{\text{cmd}} \) by

\[
G_{\text{obs}} = (X, \Sigma_{\text{cmd}} \times \Sigma_{\text{cmd}} \times \varepsilon, x_0) = Ac(2^\varepsilon, \Sigma_{\text{cmd}} \times \varepsilon, \varepsilon, UR(q_0))
\]

where \( Ac(\cdot) \) denotes the accessible part and \( UR(q_0) \) is the unobservable reach of \( q_0 \). Note that a state \( x \in X \) is a subset of \( Q_{\text{cmd}}(x \subseteq Q_{\text{cmd}}) \). The transition function \( \xi : X \times \Sigma_{\text{cmd}} \times \varepsilon \rightarrow X \) is defined, for \( x \subseteq Q_{\text{cmd}} \) and \( \sigma \in \Sigma_{\text{cmd}} \), as:

\[
\xi(x, \sigma) = UR((q \in Q : (\exists q' \in x) q \in \delta_{\text{cmd}}(q', \sigma))
\]

If the above set is empty, then \( \xi(x, \sigma) \) is undefined. We extend \( \xi \) to \( \xi : X \times \Sigma_{\text{cmd}} \rightarrow X \) in the usual way. Note that \( \xi(x_0, t) \) represent the current state estimate after observing \( t \in \Sigma_{\text{cmd}} \).

Therefore, we sometimes refer \( q \in Q_{\text{cmd}} \) as state and \( x \in X \) as state estimate in order to distinguish them.

To check recoverability for fault \( f_i \), we mark the states in \( G_{\text{obs}} \) which can be used to determine the occurrence of fault \( f_i \) as

\[
X_{D_{ij}} = \{ x \in X : (\forall q \in x)q \in Q_i \}
\]

And we mark the states in \( G_{\text{obs}} \) from which the system can be enforced to execute recovery action \( r_i \) as

\[
X_{R_{ij}} = \{ x \in X : (\forall q \in x)R_i(q) = 1 \}
\]

Note that \( X_{R_{ij}} \subseteq X_{D_{ij}} \). Let us denote the set of all loops in \( X_{D_{ij}} \) as

\[
\text{Loop}_i = \{ (x, u) \in X_{D_{ij}} \times \Sigma_{\text{cmd}} : |u| \geq 1 \land \xi(x, u) = x \}
\]

Then we have the following theorem:

**Theorem 1 (recoverability)**

A given faulty discrete event system is recoverable if and only if any loop in \( X_{D_{ij}} \) includes at least one state in \( X_{R_{ij}} \) in \( G_{\text{obs}} \) for every faulty \( f_i \). That is:

\[
(\forall (x, u) \in \text{Loop}_i)(\exists w \in \text{Pr}(u))\xi(x, w) \in X_{R_{ij}}, \quad i = 1, 2, \ldots, k
\]

**Proof:**

With the assumption that the system is diagnosable, we know if the system has a fault \( f_i \), then with finite delay, we will know the system comes to faulty mode \( G_i \), that is,

\[
(\forall s \in L_j)(\forall t \in \Sigma_j)(\exists n \in N)[|t| > n \land \delta_{\text{cmd}}(q_0, st)] = 1
\]

\[
(\forall s' \in L_{\text{cmd}}(st))s' \in L_j, \quad i = 1, 2, \ldots, k
\]

Hence the system will eventually runs in \( X_{D_{ij}} \).

For any string \( r \), if it is enough long, then \( P_{\text{cmd}}(r) \) is also enough long and includes some loop \( (x, u) \in \text{Loop}_i \), then by:

\[
(\forall (x, u) \in \text{Loop}_i)(\exists w \in \text{Pr}(u))\xi(x, w) \in X_{R_{ij}}
\]
We have:
\[
(\forall s \in L_i)(\forall t \in \Sigma_i^*)(\exists n \in N)|P|^t > n \land \delta_{cud}( q_0, st')!
\Rightarrow (\exists t' \in Pr(t))\xi(x_i, P(st')) \in X_{r,i},
\]

Hence:
\[
(\forall s \in L_i)(\forall t \in \Sigma_i^*)(\exists n \in N)|P|^t > n \land \delta_{cud}( q_0, st')!
\Rightarrow (\exists t' \in Pr(t))(\forall s' \in L_{cud}(st')) R_i(\delta_{cud}( q_0, s')) = 1
\]

The system is recoverable.

If \((\forall (x, u) \in \text{Loop}_{i} )(\exists w \in Pr(u))\xi(x, w) \in X_{r,i}\) does not satisfy. Then we have:
\[
(\exists (x, u) \in \text{Loop}_{i} )(\forall w \in Pr(u))\xi(x, w) \notin X_{r,i}
\]

We assume \(\xi(x_0, s) = x\), then we have:
\[
(\forall m \in N)(\forall w \in Pr(u))\xi(x_0, su^m w) \notin X_{r,i}
\]

That is, after string \(s\), along the string \(u^m w\), we cannot do recovery action. \(u^m w\) can be enough long, hence the system is not recoverable.

\[Q.E.D\]

**Example 3**

The faulty discrete event system is shown in figure 2.

![Fig. 2. A faulty discrete event system](image)

We assume the fault \(f_i\) is unobservable and the unobservable event is \(\mu\). In the faulty mode, at state \(q_{3,1}\), recovery action can be fired. The system can be verified to be diagnosable. Let us construct the observer for the system as shown in figure 3.

From figure 3, we get the state set \(X_{D,j}\) as:
\[
X_{D,j} = \{q_{2,1}, q_{1,1}, q_{3,1}, q_{3,3}, q_{4,1}, q_{5,1}\}
\]

And get the recoverable state set as:
\[
X_{r,j} = \{q_{3,3}\}
\]

With theorem 1, we know the system is not recoverable because loop \((q_{2,1}, q_{1,1}, \alpha)\) does not include states in \(X_{r,i} = \{q_{3,3}\}\).

The scale of the observer is exponential with the number of the faults. Here we proposed a revised approach to check recoverability as follows. For every fault \(f_i\), we construct the corresponding faulty discrete event system as:

\[
H_i = (G_o, G, \{f_i\})
\]

Then get the extended automaton \(G_{i,\text{end}}\). Based on the automaton \(G_{i,\text{end}}\), we construct the observer

\[
G_{i,\text{obs}} = (X_i, \Sigma_{i,\text{end}}, \xi_i, x_{i,0})
\]

In \(G_{i,\text{obs}}\), similarly we define:
\[
X_{i}^{\prime} = \{x \in X_i : (\forall q \in x) q \in Q_i\}
\]

\[
X_{r}^{\prime} = \{x \in X_i : (\forall q \in x) R_i(q) = 1\}
\]

And
\[
\text{Loop}^{\prime} = \{(x, u) \in X_{D}^{\prime} \times \Sigma_{i,\text{end}}^* | u| \geq 1 \land \xi(x, u) = x\}
\]

Then we have:

**Theorem 2(recoverability)**

A given faulty discrete event system is recoverable if and only if any loop in \(X_{D}^{\prime}\) includes at least one state in \(X_{r}^{\prime}\) in every \(G_{i,\text{obs}}\). That is, for every \(G_{i,\text{obs}}\):

\[
(\forall (x, u) \in \text{Loop}^{\prime})(\exists w \in Pr(u))\xi_i(x, w) \in X_{r}^{\prime}
\]

**Proof:**

With the assumption that the system is diagnosable, we know if the system has a fault \(f_i\), then with finite delay, we will know the system comes to the faulty mode, that is,
\[
(\forall s \in L_i)(\forall t \in \Sigma_i^*)(\exists n \in N)|P|^t > n \land \delta_{cud}( q_0, st')!
\Rightarrow (\forall s' \in L_{cud}(st')) s' \in L_i
\]

Then
Theorem 3 (weak recoverability)
We have the following theorem:

\[ (\forall s \in L_i)(\exists t \in \Sigma^r_\epsilon)(\exists m \in N)(t > n \land \delta_{\text{sub}}(q_0, st)!) \Rightarrow \xi(x_0, P(st)) = \xi(x_0, P(st)) \]

It means after the system enters the fault mode \( G_i \), the observers \( G_{\text{sub}, i} \) and \( G_{\text{obs}} \) have the same state estimates with finite delay. Hence we have:

\[ (\forall (x, u) \in \text{Loop}_i)(\exists w \in \text{Pr}(u))\xi(x, w) \in X_{R,i} \]
\[ \Leftrightarrow (\forall (x, u) \in \text{Loop}_i)(\exists w \in \text{Pr}(u))\xi(x, w) \in X_{R,i} \]

Q.E.D

C. Weak Recoverability
Recoverability requires when the system runs in a faulty mode, we periodically have chances to do recovery actions. It is maybe too strong. Some systems cannot ensure to periodically have such chances. However, in the worst case, for any trajectory of the system, we should at least have one chance to do recovery action. Then we have a weaker version of recoverability.

Definition 3 (weak recoverability)

For any faulty trajectory of given faulty discrete event system, we should at least have one chance to do recovery action. That is, for \( i = 1, 2, \ldots, k \)

\[ (\forall s \in L_i)(\exists t \in \text{Pr}(s))(\forall t' \in L_{\text{ref}}(t))R_i(\delta_{\text{ref}}(q_0, t')) = 1 \]

Checking weak recoverability can be done based on the observer which is constructed in subsection 3.2. In order to get the criteria, we need to make some revision for the observer. We firstly remove all the states in \( X_{R,i} \) and all the transitions connected with them. Then we remove all the deadlock states in the observer recursively. The new automaton is denoted as

\[ G_{\text{obs}, i} = (X_i, \Sigma_{u, x}, \xi, q_0) \]

We have the following theorem:

Theorem 3 (weak recoverability)

A given faulty discrete event system is weakly recoverable if and only if the revised observer \( G_{\text{obs}, i} \) does not include states in \( X_{D,i} \). That is:

\[ X_i \cap X_{D,i} = \emptyset \quad i = 1, 2, \ldots, k \]

Proof:

We firstly remove all the states in \( X_{R,i} \) and all the transitions connected with them. That is, we removed all the strings along which we have at least one chance to do recovery actions. The operation which removes all the deadlock states in the observer recursively removes all the strings which will come to states in \( X_{R,i} \) with the evolution of the system.

If the revised observer \( X_i \cap X_{D,i} = \emptyset \), that is, all strings which has enough length can ensure the system has a chance to do recovery action. Of course, the system is weakly recoverable. If \( X_i \cap X_{D,i} \neq \emptyset \), we know if the observer comes to \( X_{D,i} \), then it will stay in it forever. Also we know by a removing all the deadlock states in the observer, we removed all the deadlock states in \( X_{D,i} \). Hence there are some loops in the set \( X_i \cap X_{D,i} \), when the observer runs along these loops, we have no chance to do recovery action forever. Hence the system is not weakly recoverable.

Q.E.D

REFERENCES


