Diagnosis of Discrete Event Systems in Rules-based Model using First-order Linear Temporal Logic

Z. Huang, S. Bhattacharyya
Department of ECE
University of Kentucky
Lexington, KY 40506

V. Chandra
Department of Technology
Eastern Kentucky University
Richmond, KY 40475

S. Jiang
GM R&D Planning
MC: 480-106-200
Warren, MI, 48090

R. Kumar
Department of ECE
Iowa State University
Ames, IA 50011

***Abstract*— We study diagnosis of discrete event systems (DESs) modeled in the rule-based modeling formalism introduced in [1], and applied in [5], [6] to model failure-prone systems. An attractive feature of rule-based model is it's compactness (size is polynomial in number of signals). A motivation for the work presented is to develop failure diagnosis techniques that are able to exploit this compactness. In this regard, we develop symbolic techniques for testing diagnosability and computing a diagnoser. Diagnosability test is shown to be an instance of 1^{st} order temporal logic model-checking. An on-line algorithm for diagnoser synthesis is obtained by using predicates and predicate transformers.**

Keywords: Discrete event system, rules based model, diagnosability, 1^{st} temporal logic model checking, on-line diagnoser, predicates and predicate transformers

I. Introduction

Detection and isolation of failures in large, complex systems is a crucial and challenging task. A failure is a deviation of a system from its normal or required behavior, such as occurrence of a failure event, or visiting a failed state, or more generally, violating a design specification. A stuck-close valve, decrease in the efficiency of a heat exchanger, abnormal bias in the output of a sensor, and leakage in pipelines are examples of events that can lead to failures. Failure diagnosis is the process of detecting and identifying such deviations in a system using the information available through sensors. The problem of failure diagnosis has received considerable attention in the literature of reliability engineering, control, and computer science; and a wide variety of schemes have been proposed. Recently, it has also been studied in the framework of discrete event systems (DESs) [10], [4], [11], [7]

In this paper, we study the failure diagnosis problem for systems modeled in a rules based model [1], extended to include faults [5], [6]. State variables and rules for modifying their values are used to compactly model a DES. The representation of a system with faults, in the rules based modeling formalism, is polynomial in the size of signals and faults. The compactness of this model, together with its intuitive nature, makes it user-friendly, less error-prone, more flexible, easily scalable, and provides canonicity of representation for models of systems with faults. The motivation of the work presented here is to develop techniques for failure diagnosis that are able to exploit the compactness of the model. In this regard, we develop techniques based on 1st-order linear temporal logic model-checking (for testing diagnosability) and predicates and their transformers (for on-line diagnoser synthesis).

We illustrate through various examples how the diagnosability of DESs modeled using a rules based formalism [1] prone to faults can be checked, and how an on-line diagnoser for the system can be constructed.

The rest of the paper is organized as follows. In Section 2, the definitions of predicates, predicate transformers and 1^{st} order LTL temporal logic are introduced. In Section 3, diagnosability as a 1^{st} order LTL temporal logic model-checking is studied, and illustrated via an example. An algorithm for on-line diagnoser is provided and illustrated using an example in Section 4. Conclusion is provided in Section 5.

II. Notation and Preliminaries

A. Predicates and their Transformers

A discrete event system, denoted G, is a 4-tuple $G := (X, \Sigma, \rightsquigarrow, X_0)$, where X denotes the state set, Σ is the finite event set, $\rightsquigarrow \subseteq X \times \Sigma \times X$ is the set of state transitions, and $X_0 \subseteq X$ is the set of initial states. We use state variables to represent the states and a finite set of conditional assignment statements, called rules, to represent the state transitions.

The notation $\vec{v}$ is used to denote the vector of state variables of G. The state space X of G equals the Cartesian product of domains of all state variables, i.e., $X := \prod_{i=1}^{n} \mathcal{D}(v_i)$, where $\mathcal{D}(v_i)$ is the domain of v_i. By definition $\mathcal{D}(v_i)$ is a countable set and can be identified with the set of natural numbers $\mathcal{N}$. We use *predicates* for describing various subsets of the state space. Let $\mathcal{P}(\vec{v})$ denote the collection of predicates defined using the state variable vector $\vec{v}$, i.e., if $P(\vec{v}) \in \mathcal{P}(\vec{v})$, then it is a boolean valued map $P(\vec{v}) : X \rightarrow \{0, 1\}$. The symbols $true$ and $false$ are used for denoting predicates that hold on all and none of the states respectively. With every predicate $P(\vec{v}) \in \mathcal{P}(\vec{v})$,

This work was supported in part by the National Science Foundation under the grants NSF-ECS-0218207, NSF-ECS-0244732 and NSF-EPNES-0323379, and a DoD-EPSCoR grant from the Office of Naval Research under the grant N000140110621. S. Jiang contributed to the work while he was at the University of Kentucky.

we associate a set $X_P \subseteq X$ on which $P(\vec{v})$ takes the value one. Thus the collection of predicates $\mathcal{P}(\vec{v})$ has a one-to-one correspondence with the power set 2^X, and the names predicates and state-sets can be used interchangeably. We say that the predicate $P(\vec{v})$ holds on $\hat{X} \subseteq X$ if $\hat{X} \subseteq X_P$.

State transitions map a state to another state. Such mappings are extended to set of states or predicates in a natural way, and are known as *predicate transformers*.

Next we review the rule-based model [1] (which is a specific assignment program model [9]) for representing a DES G described above. The initial state set of G is specified as an *initial predicate*, denoted $I(\vec{v})$, which implies $X_0 = X_I$. The state transitions "$\rightsquigarrow$" of G is specified using a finite set of rules, also called conditional assignment statements, of the form:

$$\sigma : [C_\sigma(\vec{v})] \Rightarrow [\vec{v} \rightsquigarrow f_\sigma(\vec{v})],$$

where $\sigma \in \Sigma$ is an event, $C_\sigma(\vec{v})$ is a predicate, called the enabling condition or the *guard*, and $f_\sigma : X \rightarrow X$ is a map defined on the state space. If no guard is present, then $true$ is treated as the guard. A conditional assignment statement of the above type is *enabled* if the condition $C_\sigma(\vec{v})$ holds. An enabled assignment statement may *execute*. Upon execution, new values are assigned to the state variables according to the map f_σ and a state transition on the event σ occurs. For simplicity, we assume that if multiple assignment statements are simultaneously enabled, only one of them is nondeterministically executed. This assumption may be relaxed to allow *concurrency* of execution.

The substitution predicate transformer can be used to define the *forward one-step reachable*, fr, predicate transformers for G. fr determines the "postcondition" after the occurrence of a state transition for a given "precondition".

For the assignment statement $\sigma : [C_\sigma(\vec{v})] \Rightarrow [\vec{v} \rightsquigarrow f_\sigma(\vec{v})]$ and a condition $P(\vec{v})$, it is formally defined as follows: $fr(P(\vec{v}), \sigma) := C_\sigma(f_\sigma^{-1}(\vec{v})) \wedge P(f_\sigma^{-1}(\vec{v}))$.

For $\hat{\Sigma} \subseteq \Sigma$, we define $fr(P(\vec{v}), \hat{\Sigma}) := \bigvee_{\sigma \in \hat{\Sigma}} fr(P(\vec{v}), \sigma)$. Finally, note that $fr^*(P(\vec{v}), \hat{\Sigma})$ denotes the set of states which are reachable from a state in $P(\vec{v})$ by execution of zero or more transitions of events in $\hat{\Sigma}$. Clearly, fr^* is useful in characterizing the *forward reachability*.

Given $f \in \mathcal{F}$ and $P(\vec{v}) \in \mathcal{P}(\vec{v})$, the *restriction* of f to $P(\vec{v})$, denoted $f|_{P(\vec{v})}$, is the predicate transformer defined as: $f|_{P(\vec{v})}(Q(\vec{v})) := f(P(\vec{v}) \bigwedge Q(\vec{v})) \bigwedge P(\vec{v}), \forall Q(\vec{v}) \in \mathcal{P}(\vec{v})$.

B. 1^{st} order LTL & model checking

Propositional linear temporal logic (PLTL) [3] is an extension of propositional logic (PL) by the *temporal logic quantifiers/operator* $\{X, U, F, G, B\}$, called next time, until, eventually, always, and before, respectively.

First-order linear temporal logic (FOLTL) [3] is obtained by taking propositional linear temporal logic and adding to it a First order language **L**. That is, in addition to atomic propositions, truth-function connectives, and temporal operators we now also have predicates, functions, individual constants, and individual variables, each interpreted over appropriate domain.

A first order language **L** consists of *variable* symbols, *function* symbols and a set of *predicate* symbols. The zero-ary function symbols comprise the subset of *constant* symbols. Similarly, the zero-ary predicate symbols are known as the *proposition* symbols. We also have the quantifier symbols $\forall$ and $\exists$, which are applied to individual variable symbols, using the usual rules regarding scope of quantifiers, and free and bound variables.

The semantics of FOLTL is provided by a first order linear time structure $M = (S, x, L)$, where S is state set, $x : \mathcal{N} \rightarrow S$ is an infinite state sequence, and L associates with each state s an interpretation $L(s)$ of all symbols at s over a domain D such that, for each global symbol w, $L(s)(w) = L(s')(w)$, for all $s, s' \in S$.

A formula p of FOLTL is *valid* if and only if for every first order linear time structure $M = (S, x, L)$ we have $M, x \models p$. The formula p is *satisfiable* if and only if there exists $M = (S, x, L)$ such that $M, x \models p$.

III. Diagnosability as 1^{st} order LTL model-checking

Diagnosability is the ability to infer about past occurrences of unobservable failure events from a bounded number of observed events. We let $\Sigma_F \subseteq \Sigma$ denote the set of fault events, and $M : \Sigma \cup \{\epsilon\} \rightarrow \Delta \cup \{\epsilon\}$ with $M(\epsilon) = \epsilon$ denote the event observation mask. Without loss of generality, $M(\sigma) = \epsilon$ for all $\sigma \in \Sigma_F$.

The diagnosability of DESs is defined in [11], as follows (without loss of generality [7], we only consider a single "failure type"):

Definition 1: A system G is said to be diagnosable with respect to the observation mask M and the failure event set Σ_F if

$$(\exists n \in N)\ (\forall s \in L(G), s_f \in \Sigma_F)\ (\forall v = st \in L(G), ||t|| \geq n) \Rightarrow (\forall w \in L(G), M(w) = M(v))\ (\exists u \in pr(\{w\}), u_f \in \Sigma_F),$$

where $L(G)$ is the set of all event-traces G can execute starting from an initial state, $s_f, u_f \in \Sigma$ denote the last events in traces $s, u \in \Sigma^*$ respectively, and $pr(\{w\}) \subseteq \Sigma^*$ is the set of all prefixes of $w \in \Sigma^*$.

Diagnosability requires that every failure event leads to observations distinct enough to enable identification of the failure within a finite delay.

Using predicate and the extended rule-base model, we perform the diagnosis test as follows.

Algorithm 1: Consider G with state variables $\vec{v}$, event set Σ, and model given by:

Initial condition: $I(\vec{v}) \in \mathcal{P}(\vec{v})$

Event occurrence rules:

$\forall \sigma \in \Sigma : [C_\sigma(\vec{v})] \Rightarrow [\vec{v} \rightsquigarrow f_\sigma(\vec{v})]$.

The set of fault events is denoted by $\Sigma_F \subseteq \Sigma$, and the

events are partially observed through a event observation mask $M : \Sigma \cup \{\epsilon\} \rightarrow \Delta \cup \{\epsilon\}$ with $M(\epsilon) = \epsilon$, and $M(\sigma) = \epsilon$ for each $\sigma \in \Sigma_F$.

- Augment the state variables by a boolean variable, F, to identify whether or not a fault happened in past. The augmented state variable is given by, $\vec{x} = (\vec{v}, F)$. The augmented system model is given by,
  Initial condition: $I(\vec{x}) = I(\vec{v}) \wedge [F = 0]$
  Event occurrence rules:
  $\forall \sigma \in \Sigma_F : [C_\sigma(\vec{v})] \Rightarrow [\vec{x} \rightsquigarrow (f_\sigma(\vec{v}), 1)]$
  $\forall \sigma \notin \Sigma_F : [C_\sigma(\vec{v})] \Rightarrow [\vec{x} \rightsquigarrow (f_\sigma(\vec{v}), F)]$
  Denote the set of states that are visited after a fault has happened in past by the predicate, $B(\vec{x}) = B((\vec{v}, F)) := [F = 1]$.
- Perform a "masked synchronous composition" of augmented G with itself to obtain the system G_d (here $\vec{x}$ and $\vec{y}$ are used to denote the state-variables of the two copies of the augmented G):
  Initial condition: $I(\vec{x}) \bigwedge I(\vec{y})$.
  Event occurrence rule:
  $\forall (\sigma, \sigma') \in [(\Sigma \cup \{\epsilon\})^2 - \{\epsilon, \epsilon\}]$ s.t. $M(\sigma) = M(\sigma')$:
  $[C_{M(\sigma)}(\vec{x}) \wedge C_{M(\sigma')}(\vec{y})] \wedge [\sigma, \sigma' \neq \epsilon]$
  $\Rightarrow [(\vec{x}, \vec{y}) \rightsquigarrow (f_{M(\sigma)}(\vec{x}), f_{M(\sigma')}(\vec{y}))]$
  $[C_{M(\sigma)}(\vec{x})] \wedge [\sigma \neq \epsilon, \sigma' = \epsilon]$
  $\Rightarrow [(\vec{x}, \vec{y}) \rightsquigarrow (f_{M(\sigma)}(\vec{x}), \vec{y})]$
  $[C_{M(\sigma')}(\vec{y})] \wedge [\sigma = \epsilon, \sigma' \neq \epsilon]$
  $\Rightarrow [(\vec{x}, \vec{y}) \rightsquigarrow (\vec{x}, f_{M(\sigma')}(\vec{y}))]$
- Using 1^{st} order linear-time temporal logic model checking check whether there exists an "ambiguous" cycle by model-checking the following formula in G_d:

  $$\exists \vec{x_0}, \vec{y_0}[EGF(\vec{x} = \vec{x_0} \wedge \vec{y} = \vec{y_0} \wedge B(\vec{x_0}) \wedge \neg B(\vec{y_0}))].$$

  Then G is diagnosable if and only if the above formula does not hold in G_d.

The formula to be model-checked checks the for existence of a state pair $(\vec{x}_0, \vec{y}_0) \in (X \times \{0, 1\})^2$ with the property that

- $\vec{x}_0$ is a "faulty" state: $B(\vec{x}_0)$ holds,
- $\vec{y}_0$ is a "non-faulty" state: $\neg B(\vec{y}_0)$ holds,
- $(\vec{x}_0, \vec{y}_0)$ is visited infinitely often along some state trajectory starting from the initial condition $I(\vec{x}) \wedge I(\vec{y})$: $EGF\vec{x} = \vec{x}_0 \wedge \vec{y} = \vec{y}_0$, i.e., exists a path ($E$) such that globally ($G$) along each state of the path, in future (F) it holds that $\vec{x} = \vec{x}_0$ and $\vec{y} = \vec{y}_0$.

Whenever the above formula is satisfiable, there exists a pair of faulty and non-faulty traces in G of arbitrary long length that are indistinguishable, and the system is not diagnosable. The model-checking software tools such *NuSMV* [2] can be used to check the satisfiability of the above formula in G_d.

Example 1: In order to illustrate our result, we give a simple example which consists of a traffic monitoring problem of a mouse in a maze. The maze, shown in Figure 1, consists of four rooms connected by various one-way passages, where some of them have sensors installed to detect the passing of the mouse. There is also a cat which alway stays in room 1. The mouse is initially in room 0, and it can visit other rooms by using one way passages, and it never stays at one room forever. A failure occurs when the mouse visits the room occupied by the cat. Our task is to monitor the behavior of the mouse by observing the sensor signals to detect whether or not a failure occurred.

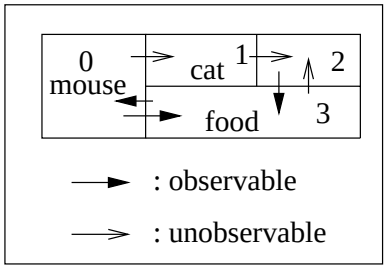

Fig. 1. Mouse in a maze

The above problem can be formulated as a failure diagnosis problem in the rules based model setting. The system to be diagnosed, G, has a single state variable v denoting the location of the mouse in the maze, and it can take the values in the set $\{0, 1, 2, 3\}$; the initial state is $I(v) = [v = 0]$; the event set is $\Sigma = \{o_1, o_2, o_3, u_1, u_2, u_3\}$; the event observation mask M is given as $M(u_i) = \epsilon$ and $M(o_i) = o_i$ for $1 \leq i \leq 3$. The rules based model of mouse in a maze is shown in Figure 2. Since a fault occurs when room 1 is visited, $\Sigma_F = \{u_1\}$, where note that u_1 is an unobservable event.

1) Initial condition: $I(v) = [v = 0]$.
2) Event occurrence rules:

$$o_1 : [v = 0] \Rightarrow [v \rightsquigarrow 3]$$
$$o_2 : [v = 3] \Rightarrow [v \rightsquigarrow 0]$$
$$o_3 : [v = 2] \Rightarrow [v \rightsquigarrow 3]$$
$$u_1 : [v = 0] \Rightarrow [v \rightsquigarrow 1]$$
$$u_2 : [v = 1] \Rightarrow [v \rightsquigarrow 2]$$
$$u_3 : [v = 3] \Rightarrow [v \rightsquigarrow 2]$$

Fig. 2. Rules based model of mouse in a maze

In order to verify diagnosability, we augment the state variable v by the binary valued variable F to obtain $\vec{x} := (v, F)$. Next using the state variable $\vec{y} = (u, E)$ for the second copy of G, we compute the masked composition of augmented G with itself to obtain G_d. Note that the observable events o_1, o_2, o_3 execute synchronously, whereas the unobservable events u_1, u_2, u_3 occur asynchronously.

The NuSMV tool allows computation of masked synchronous composition. Using the NuSMV [2] tool for model-checking the diagnosability condition we verified that the mouse in a maze is diagnosable. Since a rules based model provides a compact model (in contrast, an automaton model enumerates all the states), we hope that the symbolic

techniques developed in this paper will allow for the diagnosability verification of industrial size problems.

Further, we can use the diagnosability algorithm developed above together with the optimal sensor selection algorithm given in [8] to obtain an optimal observation mask ensuring the system diagnosability. Using this approach, we determined that the event o_3 need not be observable for the system to remain diagnosable.

IV. On-line Diagnoser using Predicates & their Transformers

We embark upon the on-line computation of a diagnoser once the system has been determined to be diagnosable. Again as with the test for diagnosability, we develop symbolic methods for the on-line computation of the diagnoser.

The diagnoser maintains two predicates: one, denoted $E_k(\vec{x}) \in \mathcal{P}(\vec{x})$, is an estimate of the possible states following the occurrence of kth observable event, and the other denoted $N_k(\vec{x}) \in \mathcal{P}(\vec{x})$ is a subset of $E_k(\vec{x})$ that is reached along trajectories that never visit a state in $B(\vec{x})$, i.e., along those non-faulty trajectories where $\neg B(\vec{x})$ holds invariantly.

Initially, when no observation has occurred, i.e., when $k = 0$,

$$E_0(\vec{x}) = I(\vec{x}), \quad N_0(\vec{x}) = I(\vec{x}) \wedge \neg B(\vec{x}).$$

Upon the occurrence of the $(k+1)$th observable event ($k \geq 0$), the pair $(E_k(\vec{x}), N_k(\vec{x}))$ is updated to obtain the pair $(E_{k+1}(\vec{x}), N_{k+1}(\vec{x}))$. Whenever $N_k(\vec{x})$ is a strict subset of $E_k(\vec{x})$, and $N_k(\vec{x}) \neq false$, it means that the system could have executed some trajectories that visited a faulty state in past (since $E_k(\vec{x}) \neq false$), and also some other trajectories (that are indistinguishable to the former) that never visited a faulty state in past (since $N_k(\vec{x}) \neq false$). In other words, in such a case, there exists an ambiguity as to whether or not a fault occurred in past. Such an ambiguity does not exist if

$$[E_k(\vec{x}) \neq false] \wedge [N_k(\vec{x}) = false],$$

which means that along all trajectories the system could have executed, a faulty state was visited in past. A fault is reported by the diagnoser at such a point.

Algorithm 2:

- **Initiation step:**
  $E_0(\vec{x}) = I(\vec{x})$
  $N_0(\vec{x}) = I(\vec{x}) \bigwedge \neg B(\vec{x})$
- **Iteration step:**
  Upon $(k+1)$th observation $\delta \in M(\Sigma) - \{\epsilon\}$:
  $E_{k+1}(\vec{x}) = fr_{M^{-1}(\delta)}[fr^*_{M^{-1}(\epsilon)\cap\Sigma}(E_k(\vec{x}))]$
  $N_{k+1}(\vec{x}) =$
  $(fr|_{\neg B(\vec{x})})_{M^{-1}(\delta)}[(fr|_{\neg B(\vec{x})})^*_{M^{-1}(\epsilon)\cap\Sigma}(N_k(\vec{x}))]$
  Declare a fault if:
  $[E_{k+1}(\vec{x}) \neq false] \wedge [N_{k+1}(\vec{x}) = false]$.

In the iteration step, $E_{k+1}(\vec{x})$ is computed using a reachability computation starting from $E_k(\vec{x})$ on sequences of unobservable events in $M^{-1}(\epsilon) \cap \Sigma$ followed by a single event in $M^{-1}(\delta)$ (since the $(k+1)$th observation of δ results from the execution of a sequence of unobservable events in $M^{-1}(\epsilon) \cap \Sigma$ followed by the execution of an event in $M^{-1}(\delta)$). $N_{k+1}(\vec{x})$ is computed in a similar way except the forward reachability predicate transformer fr is replaced by it's restriction to $\neg B(\vec{x})$, i.e., by $(fr|_{\neg B(\vec{x})})$.

Remark 1: The computation of $N_k(\vec{x})$ may alternatively be performed by considering the fault-free subsystem in which the guard condition for each faulty event in Σ_F has been set to FALSE. Then $N_k(\vec{x})$ for the system with faults is the same as $E_k(\vec{x})$ of the fault-free system:

$$N_0(\vec{v}) = fr^*_{M^{-1}(\epsilon)\cap\Sigma} I(\vec{v});$$

$$N_{k+1}(\vec{v}) = fr_{M^{-1}(\delta_k)}[fr^*_{M^{-1}(\epsilon)\cap\Sigma}(N_k(\vec{v}))],$$

where $\delta_k \in M(\Sigma) - \{\epsilon\}$ denotes the kth observation.

We illustrate the above algorithm for on-line computation of the diagnoser using the example of mouse in a maze given earlier.

Example 2: Refer to the example of Section III. We can compute the diagnoser as follows:

- $k = 0$**:** Since $I(\vec{x}) = [v = F = 0]$ and $B(\vec{x}) = [F = 1]$, we set

  $$E_0(\vec{x}) = [v = F = 0], \quad N_0(\vec{x}) = [v = F = 0].$$

- $k = 1$**:** There are three observable events $o_1, o_2, o_3 \in \Sigma$. If the first observation is o_1, then

  $$E_1(\vec{x}) = [v = 3, F = 0], \quad N_1(\vec{x}) = [v = 3, F = 0].$$

  If the first observation is o_2, then

  $$E_1(\vec{x}) = false, \quad N_1(\vec{x}) = false.$$

  (This means o_1 as first observation is not possible.) If the first observation is o_3, then

  $$E_1(\vec{x}) = [v = 3, F = 1], \quad N_1(\vec{x}) = false.$$

  This means that a fault has occurred sometimes in past.
- $k = 2$**:** Suppose the first observation $(k = 1)$ is o_1. If the next observation is o_1, then

  $$E_2(\vec{x}) = false, \quad N_2(\vec{x}) = false.$$

  (This means the observation sequence $o_1 o_1$ is not possible.) If the next observation is o_2, then

  $$E_2(\vec{x}) = [v = 0, F = 0] = E_0(\vec{x}),$$

  $$N_2(\vec{x}) = [v = 0, F = 0] = N_0(\vec{x}).$$

  If the next observation is o_3, then

  $$E_2(\vec{x}) = [v = 3, F = 0], \quad N_2(\vec{x}) = [v = 3, F = 0].$$

  Next suppose the first observation $(k = 1)$ is o_2. Then since $E_1(\vec{x}) = N_1(\vec{x}) = false$, it follows that $E_2(\vec{x}) = N_2(\vec{x}) = false$.

Finally suppose the first observation $(k = 1)$ is o_3. If the next observation is o_1, then

$$E_2(\vec{x}) = false, \quad N_2(\vec{x}) = false.$$

(This means the observation sequence o_3o_1 is not possible.) If the next observation is o_2, then

$$E_2(\vec{x}) = [v = 0, F = 1], \quad N_2(\vec{x}) = false.$$

If the next observation is o_3, then

$$E_2(\vec{x}) = [v = 3, F = 1], \quad N_2(\vec{x}) = false.$$

In both the above cases, we know that a fault has occurred in past.

- $k = 3$**:** After two observations the only predicate pair that is not "re-visited" is the one following the observation sequence o_3o_2:

$$E_2(\vec{x}) = [v = 0, F = 1], \quad N_2(\vec{x}) = false.$$

So we examine this predicate pair. If the next observation is o_1, then

$$E_3(\vec{x}) = [v = 3, F = 1], \quad N_3(\vec{x}) = false.$$

If the next observation is o_2, then

$$E_3(\vec{x}) = false, \quad N_3(\vec{x}) = false.$$

(This means the observation sequence $o_3o_2o_2$ is not possible.) If the next observation is o_3, then

$$E_3(\vec{x}) = [v = 3, F = 1], \quad N_3(\vec{x}) = false.$$

Thus the third iteration step does not introduce a predicate pair that has never been "visited" before, and so there is no need to iterate further. (Said another way, further iterations will yield a predicate pair that has already been computed above.)

In this case, the on-line computation of the three steps considered above yields a off-line diagnoser that can be represented as an automaton as shown in Figure 3. In

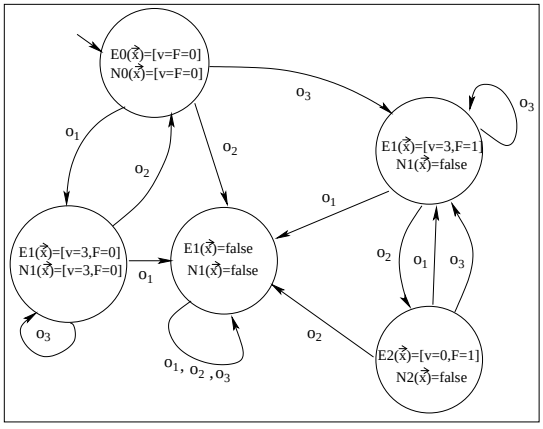

Fig. 3. Diagnoser for mouse in a maze

Figure 3, when there is a transition from a predicate pair $[E(\vec{x}) \neq false] \wedge [N(\vec{x}) \neq false]$ to the predicate pair $[E(\vec{x}) \neq false] \wedge [N(\vec{x}) = false]$, a fault is reported by the diagnoser. Further since the predicate pair $[E(\vec{x}) = false] \wedge [N(\vec{x}) = false]$ represents an impossibility, the diagnoser can be reduced by restricting it to those predicate pairs that are of the type $[E(\vec{x}) \neq false] \wedge [N(\vec{x}) \neq false]$. This restricted diagnoser is shown in Figure 4. Any observation sequence that leads to being outside the restricted diagnoser automaton indicates that a fault must have occurred in past.

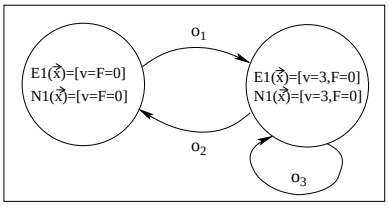

Fig. 4. The reduced diagnoser for mouse in a maze

V. Conclusion

The rules based modeling formalism of [1] has been used to models discrete event systems prone to failures. Stuck-signal faults, and system/equipment faults can be easily modeled in the rules based modeling formalism. Symbolic computation based algorithms for checking the diagnosability of DESs using 1^{st} order model-checking and for on-line diagnoser synthesis using predicates and predicate transformers have been developed. The advantage of using rule-based model is it's compactness since it uses state-variables to represent states. The number of rules in the rule-based model is polynomial in number of state variables. Symbolic methods allow for failure analysis without exhaustively performing a reachability of the entire state space. Plus, software tools such as NuSMV exist for performing 1^{st}-order model-checking for systems with finite/bounded state-space.

References

[1] V. Chandra and R. Kumar. A event occurrence rules based compact modeling formalism for a class of discrete event systems. *Mathematical and Computer Modeling of Dynamical Systems*, 8(1):49–73, 2002.

[2] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, and M. Pistore. NuSMV 2: An opensourse tool for symbolic model checking. In *Proccedings of International Conference on Computer-Aided Verification*, Copenhagen, Denmark, July 2002.

[3] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier Science Publishers, 1990.

[4] 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.

[5] Z. Hunag, V. Chandra, S. Jiang, and R. Kumar. Modeling discrete event systems with faults using a rules-based modeling formalism. In *Proccedings of IEEE Conference on Decision and Control*, pages 4012–4017, Las Vegas, NV, December 2002.

[6] Z. Hunag, V. Chandra, S. Jiang, and R. Kumar. Modeling discrete event systems with faults using a rules-based modeling formalism. *Mathematical and Computer Modeling of Dynamical Systems*, 9(3):233–254, 2003.

[7] 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.

[8] S. Jiang, R. Kumar, and H. E. Garcia. Optimal sensor selection for discrete event systems under partial observation. *IEEE Transactions on Automatic Control*, 48(3):369–381, 2003.

[9] R. Kumar, V. K. Garg, and S. I. Marcus. Predicates and predicate transformers for supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, 38(2):232–247, February 1993.

[10] F. Lin. Diagnosability of discrete event systems and its applications. *Discrete Event Dynamic Systems: Theory and Applications*, 4(1):197–212, 1994.

[11] M. Sampath, R. Sengupta, S. Lafortune, K. Sinaamohideen, and D. Teneketzis. Diagonsability of discrete event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.