EFFICIENT ON-LINE FAILURE IDENTIFICATION FOR DISCRETE-EVENT SYSTEMS [1]

Anika Schumann [*] **Yannick Pencolé** [**]

[*] *The Australian National University, National ICT Australia*
[**] *The Australian National University, National ICT Australia*

Abstract: The paper addresses the problem of diagnosing complex embedded systems. Given a flow of observations from the system, the goal is to explain these observations by identifying possible failures. Approaches that efficiently compute the possible failures, like the classical diagnoser approach, suffer from high space complexity. This paper presents a method that generalises the observable behaviour of a system resulting in a smaller finite state machine, that efficiently maps observations to possible failures.

Keywords: diagnosis, model-based representation and reasoning, discrete-event dynamic systems

1. INTRODUCTION

The problem which is target in this paper is the supervision of complex embedded discrete event systems. Given a supervision agent continuously receiving observations sent by the system, the purpose is to help him to identify failures.

It is now widely recognised that diagnosing complex systems is best achieved through model-based representation (Lamperti and Zanella, 2001). That is, rather than relying on procedural expert knowledge, model-based approaches exploit models of individual components of the system and combine them to automate the reasoning about system wide interactions. Among these approaches are diagnoser based ones (introduced in (Sampath *et al.*, 1996)) that compile, off-line, a representation of the system model by a finite state machine (the diagnoser), that directly maps observations to possible system states and failures. Since on-line diagnosis algorithms based on the diagnoser require not a lot of computational resources, it is an interesting approach for the implementation of an embedded monitoring system. However, even if the diagnoser of an embedded system is computable off-line with classical computers, it does not mean that it can be embedded in the small device that is in charge of monitoring this system due to limited memory resources.

In (Zad *et al.*, 2003) the authors have shown how the size of a state-based diagnoser can be reduced. In such a state-based approach, in which observations and failures are part of the state label, it is assumed that the state set of the system can be partitioned according to the failures of the system.

In contrast, this paper describes an event-based diagnoser approach, in which observations and failures are modelled as events. Here the states of the system are not related to the failures. The aim is to efficiently compute an even smaller finite state machine than the one retrieved in (Zad *et al.*, 2003), that directly maps observations to the failure sets of the system. This reduction is achieved by generalising the observable system behaviour.

In contrast to other reduction techniques for finite state machines (see e.g. (Gold, 1972), (Christensen *et al.*, 1992)), it is not necessary to maintain the same or a similar behaviour of the original finite state machine,

[1] This research was supported by National ICT Australia (NICTA) in the framework of the SuperCom project (Model-Based Supervision of Composite Systems). NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian National Research Council.

that is the diagnoser. Since the global model defines the complete observable behaviour of the system it is only possible to observe event sequences defined in this model. Thus the observable behaviour can be generalised if it can be guaranteed that the resulting finite state machine still allows the computation of the consistent failure sets for all possible sequences of observations. All failure identifiers satisfy this condition.

They can be used for a wide range of applications which require only a knowledge of the possible failures that explain a sequence of observations and not a knowledge of the corresponding system states.

The approach is implemented in a symbolic way that has shown to provide a good trade-off for optimising the memory consumption and the time performance of any diagnoser (Schumann *et al.*, 2004). An experimental size comparison of the symbolic representations of diagnoser and reduced failure identifier clearly highlights the advantage of the latter.

The paper is organised as follows. Section 2 introduces the failure identification problem of discrete event systems. In the next section the failure identifier is defined and section 4 sketches the computation of the *restricted failure identifier*, that is defined only for the observable system behaviour. Section 5 describes an algorithm to reduce the size of the restricted failure identifier and section 6 presents an experimental comparison between such a failure identifier and the original diagnoser.

2. FAILURE IDENTIFICATION IN DISCRETE-EVENT SYSTEMS

To start with, a brief introduction to failure identification of systems modelled as discrete-event is given. These systems can be represented as finite state machines, whereby the states and transitions reflect the normal and the failure behaviour of the system. This is achieved by modelling observations and failures as events. [2]

Definition 1. (System Model). A system model is a finite state machine $G = \langle X, \Sigma_o, \Sigma_f, x_0, T\rangle$, characterised by its state set X, its observable and failure events Σ_o and Σ_f, its initial state x_0 and its transition set T.

Figure 1 depicts such a system model. The failure identification problem consists in determining all failures that are consistent with a sequence of observations.

Definition 2. (Consistent Failure Sets).
Let $P_F = x_{j_1} \xrightarrow{f_{j_1}} x_{j_2} \cdots \xrightarrow{f_{j_{q-1}}} x_{j_q}$ denote a failure path in system $G = \langle X, \Sigma_o, \Sigma_f, x_0, T\rangle$ with $f_{j_i} \in \Sigma_f \ \forall f_{j_i}$, that can consist of a single state only. Further let $Start(P_F) \in X$ denote the start state of the path and $Event(P_F) \in 2^{\Sigma_f}$ the set of all event labels in P_F.

The set of failure sets $F_S \subseteq 2^{2^{\Sigma_f}}$ that are consistent with a sequence of observable events $S = [o_1, \ldots, o_k]$ is defined as follows:

$$F_S = \{\bigcup_{j=1}^{k} Event(P_{F_j}) \mid \quad p = P_{F_1} \xrightarrow{o_1} P_{F_2} \cdots \xrightarrow{o_{k-1}} P_{F_k} \xrightarrow{o_k} x \text{ is path in G with } Start(P_{F_1}) = x_0\}$$

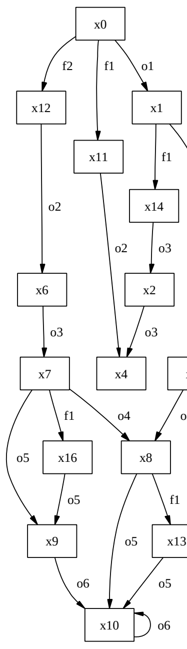


Figure 1. System model with events $\Sigma_o = \{o_1, \ldots, o_6\}$ and $\Sigma_f = \{f_1, f_2\}$.

For example, consider all paths of the system shown in Figure 1 that would explain why first o_2 then o_3 and finally o_5 is observed, that is consider the paths $x_0 \xrightarrow{f_2} x_{12} \xrightarrow{o_2} x_6 \xrightarrow{o_3} x_7 \xrightarrow{o_5} x_9$ and $x_0 \xrightarrow{f_2} x_{12} \xrightarrow{o_2} x_6 \xrightarrow{o_3} x_7 \xrightarrow{f_1} x_{16} \xrightarrow{o_5} x_9$. The failure sets consistent with the sequence $S = [o_2, o_3, o_5]$ are $F_S = \{\{f_2\}, \{f_1, f_2\}\}$. Note that consistent failure sets need to be determined only for the observable system behaviour, that is for all observable sequences defined in G.

Definition 3. (Possible observable sequence). An observable sequence $S = [o_1, \ldots, o_k]$ with $o_i \in \Sigma_o, \ \forall o_i$ is called *possible* with respect to a system $G = \langle X, \Sigma_o, \Sigma_f, x_0, T\rangle$, iff the following condition holds:

$$psbl(S) \Leftrightarrow \exists \text{ path } p = P_{F_1} \xrightarrow{o_1} P_{F_2} \cdots \xrightarrow{o_{k-1}} P_{F_k} \xrightarrow{o_k} x \text{ in } G \text{ with Start}(P_{F_{i_1}}) = x_0 \text{ and } P_{F_i} \text{ is failure path (as defined in definition 2)}$$

[2] For the sake of simplicity and without loss of generality, the system model is defined in a centralised way and events other than observations and failures are omitted. See e.g. (Sampath *et al.*, 1996) for a complete definition.

For instance, in the system depicted in Figure 1, the sequence $S = [o_2, o_3, o_5]$ is possible as shown above. In contrast, the sequence $S = [o_3, o_4]$ is not possible. The observable system behaviour constitutes of all possible observable sequences.

3. FAILURE IDENTIFIER

The previous section has shown how the system model can be used to retrieve all failure sets consistent with a sequence of observations. However, the computation of these failure sets requires the retrieval of all paths that explain the observed event sequence, which can be very slow for large complex systems. Now a model, the *failure identifier*, that efficiently maps all possible observable sequences to the corresponding consistent failure sets is introduced.

Definition 4. (Failure Identifier). Given a system $G = \langle X, \Sigma_o, \Sigma_f, x_0, T\rangle$. The model of a failure identifier is a deterministic finite state machine $\widetilde{FI} = \langle \widetilde{X}, \widetilde{F}, \widetilde{\Sigma}, \widetilde{x}_o, \widetilde{R}, \widetilde{T}\rangle$, with states $\widetilde{X}$, state labels $\widetilde{F} \subseteq 2^{2^{\Sigma_f}}$, transition labels $\widetilde{\Sigma} = \Sigma_o$, initial state $\widetilde{x}_0$, state labelling function $\widetilde{R} : \widetilde{X} \to \widetilde{F}$ with $\widetilde{R}(\widetilde{x}_0) = \emptyset$ and transitions $\widetilde{T} \subseteq \widetilde{X} \times \widetilde{\Sigma} \times \widetilde{X}$ such that

$$\begin{aligned}
&\forall S = [o_1, \ldots, o_k] \text{ such that } psbl(S)\\
&\quad \exists \widetilde{p} = \widetilde{x}_0 \xrightarrow{o_1} \widetilde{x}_{j_1} \cdots \xrightarrow{o_{k-1}} \widetilde{x}_{j_{k-1}} \xrightarrow{o_k} \widetilde{x}_{j_k} \text{ in } \widetilde{FI} \text{ with}\\
&\quad \widetilde{R}(\widetilde{x}_{j_k}) = \{\bigcup_j \text{Event}(P_{F_j}) \mid\\
&\qquad\qquad P_{F_1} \xrightarrow{o_1} P_{F_2} \cdots \xrightarrow{o_{k-1}} P_{F_k} \xrightarrow{o_k} x\\
&\qquad\qquad \text{is path in } G \text{ with Start}(P_{F_1}) = x_0\}
\end{aligned}$$

Consider, for instance, the system depicted in Figure 1. The possible observable sequences are $[o_1, o_3, o_3, o_4, o_5, o_6^*]$, $[o_2, o_3, o_4, o_5, o_6^*]$ and $[o_2, o_3, o_5, o_6^*]$ and any subsequences of it containing the first $k \in \mathbb{N}$ events.[3] Figure 2 illustrates some of its failure identifiers. The examples show, that the failure identifiers of a system cannot only differ in the number of states and transitions, but also in the number of observable sequences for which they are defined. For instance, the sequence $S = [o_3, o_4]$ is defined for $\widetilde{FI}_C$ but not for $\widetilde{FI}_A$. However, since this sequence is not observable, it is irrelevant whether a failure identifier defines it or not. Only the possible observable sequences of a system need to be defined in all its failure identifiers.

In order to compute the failure sets that are consistent with a sequence of observable events S based on an arbitrary failure identifier of the system, it is sufficient to follow the unique path of $\widetilde{FI}$ transitions labelled with the observed events and to look up the label of the target state. Consider for example the failure identifier $\widetilde{FI}_B$ depicted in Figure 2. When observing the events o_2, o_3, and o_5, it is sufficient to follow the path $b_0 \xrightarrow{o_2} b_4 \xrightarrow{o_3} b_5 \xrightarrow{o_5} b_7$ and to retrieve the consistent failure sets directly from the label of state b_7 which is $\{\{f_2\}, \{f_1, f_2\}\}$.

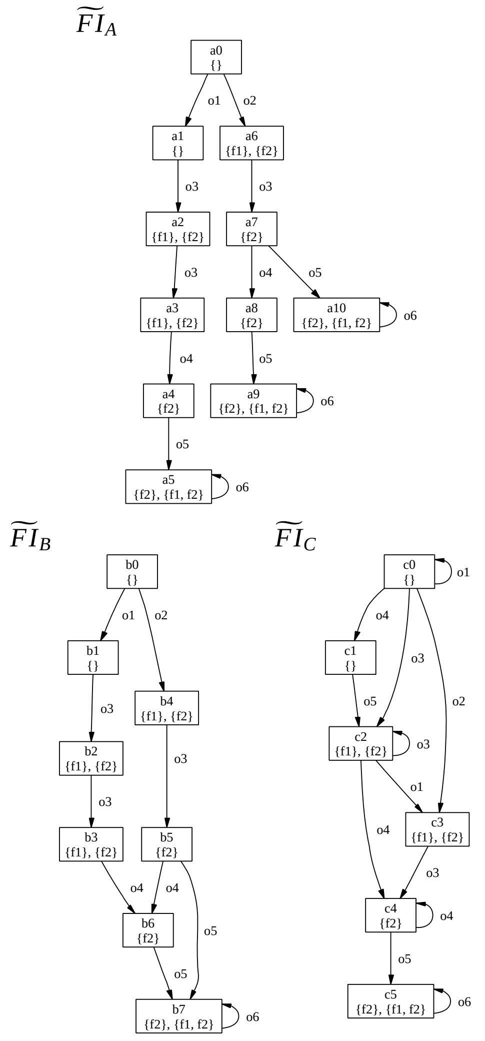


Figure 2. Three failure identifiers for the system shown in Figure 1.

The classical diagnoser (Sampath *et al.*, 1996) is a special instance of a failure identifier. Its states are not only labelled with the consistent failure sets but also with the system states that are consistent with a sequence of observations. Furthermore the diagnoser has the special property that it is only defined for sequences of observations that are possible in the system.

Definition 5. (Diagnoser). Let $G = \langle X, \Sigma_o, \Sigma_f, x_0, T\rangle$ denote the model of the system, P the set of path defined in G, and $L = 2^{\Sigma_f}$ the set of failure labels.

The model of a diagnoser is a deterministic finite state machine $\hat{D} = \langle \hat{X}, \hat{\Pi}, \hat{\Sigma}, \hat{x}_o, \hat{R}, \hat{T}\rangle$, where $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_q\}$ is the set of diagnoser states, $\hat{\Pi} = X \times L$ is the set of pairs that can belong to diagnoser state labels, $\hat{\Sigma} = \Sigma_o$ is the set of diagnoser transition labels, $\hat{x}_0$ is the initial diagnoser state, $\hat{R} : \hat{X} \mapsto \hat{\Pi}$ is the diagnoser state labelling function that verifies $\hat{R}(\hat{x}_0) =$

[3] The notion o_6^* refers to an infi nite number of observations o_6.

$\{(x_0, \emptyset)\}$, and $\hat{T} \subseteq \hat{X} \times \hat{\Sigma} \times \hat{X}$ is the set of diagnoser transitions, which verify:

$$
\begin{aligned}
&\hat{x} \xrightarrow{\sigma} \hat{x}' \in \hat{T} \text{ iff} \\
&\quad \hat{R}(\hat{x}') = \\
&\quad \Big\{(x', l') \mid \exists (x, l) \in \hat{R}(\hat{x}) \text{ such that either} \\
&\qquad\qquad x \xrightarrow{\sigma} x' \in T \text{ and } l' = l, \\
&\qquad\qquad \text{or} \\
&\qquad\qquad P_F \xrightarrow{\sigma} x' \in P \text{ with } Start(P_F) = x, \text{ and} \\
&\qquad\qquad\qquad l' = l \cup Event(P_F)\Big\}
\end{aligned}
$$

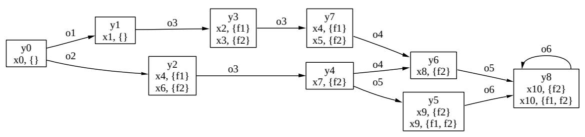


Figure 3. Diagnoser for the system shown in Figure 1.

Figure 3 depicts the diagnoser for the example shown in Figure 1. Let $\hat{R}_F : \hat{\Pi} \to \widetilde{F}$ denote the function that abstracts the system states from the diagnoser state label. The diagnoser definition verifies

$$
\begin{aligned}
&\forall S = [o_1, \ldots, o_k] \text{ such that } psbl(S) \\
&\ \exists \hat{p} = \hat{x}_0 \xrightarrow{o_1} \hat{x}_{i_1} \cdots \xrightarrow{o_{k-1}} \hat{x}_{i_{k-1}} \xrightarrow{o_k} \hat{x}_{i_k} \text{ in } \hat{G} \text{ with} \\
&\quad \hat{R}(\hat{x}_{j_k}) = \Big\{(x, \bigcup_j \text{Event}(P_{F_j})) \mid \\
&\qquad\qquad P_{F_1} \xrightarrow{o_1} P_{F_2} \cdots \xrightarrow{o_{k-1}} P_{F_k} \xrightarrow{o_k} x \\
&\qquad\qquad \text{is path in } G \text{ with } \text{Start}(P_{F_1}) = x_0\Big\} \text{ and} \\
&\quad \hat{R}_F(\hat{R}(\hat{x}_{j_k})) = \Big\{ \bigcup_j \text{Event}(P_{F_j}) \mid \\
&\qquad\qquad P_{F_1} \xrightarrow{o_1} P_{F_2} \cdots \xrightarrow{o_{k-1}} P_{F_k} \xrightarrow{o_k} x \\
&\qquad\qquad \text{is path in } G \text{ with } \text{Start}(P_{F_1}) = x_0\Big\}
\end{aligned}
$$

This means that the diagnoser is indeed a failure identifier as defined in definition 4. In contrast to a failure identifier in general, the diagnoser is only defined for the possible observable system behaviour and is uniquely determined. It can be computed directly from the system description and can be used for efficiently retrieving the consistent failure sets. However, such an algorithm has a very high space complexity due to the need of storing the diagnoser. As stated, for instance, in (Pencolé *et al.*, 2002) the size of the diagnoser constitutes a major problem. Since the time complexity of computing the consistent failure sets is the same for all failure identifiers of a system, it is interesting to obtain one with minimal size.

4. RESTRICTED FAILURE IDENTIFIER

This section defines the smallest failure identifier that is defined only for the possible observable sequences of a system. For this purpose it is necessary to introduce the concept of bisimilarity (Milner, 1980) (Park, 1981). The essence of bisimilarity as stated in (Hennessy and Milner, 1985), "is that the behaviour of a program is determined by how it communicates with an observer". This implies that there is no need to distinguish two states of a finite state machine that have the same set of outgoing transitions and for which the transition pairs with the same label lead to two states that are also not distinguishable. With respect to the failure identifier bisimulation is referred to in the following way:

Definition 6. (Bisimulation). A *bisimulation* is a binary Relation B on two states $\widetilde{x_i}$, $\widetilde{x_j}$ of a failure identifier $\widetilde{FI} = \langle \widetilde{X}, \widetilde{F}, \widetilde{\Sigma}, \widetilde{x_o}, \widetilde{R}, \widetilde{T} \rangle$ iff the following condition holds:

$$
\begin{aligned}
&B(\widetilde{x_i}, \widetilde{x_j}) \Leftrightarrow \widetilde{R}(\widetilde{x_i}) = \widetilde{R}(\widetilde{x_j}) \text{ and} \\
&\qquad \forall \widetilde{x_i}' \in \widetilde{X} \text{ and } \forall \sigma \in \widetilde{\Sigma} \text{ such that } (\widetilde{x_i} \xrightarrow{\sigma} \widetilde{x_i}') \in \widetilde{T} \\
&\qquad\quad \exists (\widetilde{x_j} \xrightarrow{\sigma} \widetilde{x_j}') \in \widetilde{T} \text{ with } B(\widetilde{x_i}', \widetilde{x_j}') \text{ and} \\
&\qquad \forall \widetilde{x_j}' \in \widetilde{X} \text{ and } \forall \sigma \in \widetilde{\Sigma} \text{ such that } (\widetilde{x_j} \xrightarrow{\sigma} \widetilde{x_j}') \in \widetilde{T} \\
&\qquad\quad \exists (\widetilde{x_i} \xrightarrow{\sigma} \widetilde{x_i}') \in \widetilde{T} \text{ with } B(\widetilde{x_i}', \widetilde{x_j}')
\end{aligned}
$$

Let $\widetilde{FI} = \langle \widetilde{X}, \widetilde{F}, \widetilde{\Sigma}, \widetilde{x_o}, \widetilde{R}, \widetilde{T} \rangle$ denote a failure identifier with the bisimular state pair $B(\widetilde{x_i}, \widetilde{x_j})$. The removal of state $\widetilde{x_j}$ leads to the new state set $\widetilde{X}' = \widetilde{X} \setminus \{\widetilde{x_j}\}$ and new transition set $\widetilde{T}' = \widetilde{T} \setminus \{\widetilde{x_a} \xrightarrow{o} \widetilde{x_b} \mid \widetilde{x_a} = \widetilde{x_j} \text{ or } \widetilde{x_b} = \widetilde{x_j}\} \cup \{\widetilde{x_a} \xrightarrow{o} \widetilde{x_i} \mid \widetilde{x_a} \xrightarrow{o} \widetilde{x_j} \in \widetilde{T}\}$. The finite state machine $\widetilde{FI}' = \langle \widetilde{X}', \widetilde{F}, \widetilde{\Sigma}, \widetilde{x_o}, \widetilde{R}, \widetilde{T}' \rangle$ is also a failure identifier since for all paths $\widetilde{p} = \widetilde{x_0} \xrightarrow{o_1} \widetilde{x_{a_1}} \cdots \xrightarrow{o_k} \widetilde{x_{a_k}}$ in $\widetilde{FI}$ there exists a path $\widetilde{p} = \widetilde{x_0} \xrightarrow{o_1} \widetilde{x_{b_1}} \cdots \xrightarrow{o_k} \widetilde{x_{b_k}}$ in $\widetilde{FI}'$ with

$$
\widetilde{R}(x_{a_c}) = \widetilde{R}(x_{b_c}) \text{ and } \widetilde{x_{b_c}} = \begin{cases} \widetilde{x_i} & \text{if } \widetilde{x_{a_c}} = \widetilde{x_j} \\ \widetilde{x_{a_c}} & \text{otherwise} \end{cases}
$$

Note that when removing all bisimular states of a finite state machine, it is not necessary to check whether the target states of the removed state are still reachable from the initial states, since all target states of a bisimular state are also bisimular and therefore removed.

Consider for instance the failure identifier $\widetilde{FI_A}$ depicted in Figure 2. The states a_5, a_9 and a_{10} are pairwise bisimilar as well as the states a_4 and a_8. $\widetilde{FI_B}$ of the same Figure shows the failure identifier after the removal of all bisimular states.

Definition 7. (Restricted failure identifier). The failure identifier $\widetilde{FI}_{rest}$ of a system G is called *restricted* iff it is defined only for the observable behaviour of the system and if it does not contain bisimilar states.

Every system G has a unique restricted failure identifier. It can be obtained from the diagnoser by removing all bisimilar states as described above. For the example system (see Figure 1) the restricted failure identifier $\widetilde{FI_B}$ is shown in the middle of Figure 2.

5. REDUCED FAILURE IDENTIFIERS AND THEIR COMPUTATION

The size of the restricted failure identifier can be reduced by merging states that share the same failure label and that do not contain any transitions with the same label that lead to different target states. Latter condition verifies that the resulting finite state machine is also deterministic.

Definition 8. (Mergable states). A set of states $\widetilde{Z} \subseteq \widetilde{X}$ of the restricted failure identifier $\widetilde{FI}_{rest} = \langle \widetilde{X}, \widetilde{F}, \widetilde{\Sigma}, \widetilde{x}_o, \widetilde{R}, \widetilde{T} \rangle$ is called mergable iff the following condition holds:

$$\begin{aligned} merg(\widetilde{Z}) \;\leftrightarrow\; & \forall \widetilde{x}_i, \widetilde{x}_j \in \widetilde{Z} \text{ it holds } \widetilde{R}(\widetilde{x}_i) = \widetilde{R}(\widetilde{x}_j) \text{ and} \\ & \forall o \in \widetilde{\Sigma} \;\; \widetilde{x}_i \overset{o}{\rightarrow} \widetilde{x}'_i \in \widetilde{T} \text{ and } \widetilde{x}_j \overset{o}{\rightarrow} \widetilde{x}'_j \in \widetilde{T} \\ & \quad \text{implies either } \widetilde{x}'_i = \widetilde{x}'_j \text{ or } merg(\{\widetilde{x}'_i, \widetilde{x}'_j\}) \end{aligned}$$

Hence two states are mergable even if their outgoing transition labels are not identical. In fact, since the restricted failure identifier does not contain any bisimular states, a merge of states results necessarily in a generalisation of the observable system behaviour.

Let $\widetilde{Z}_i = \{\widetilde{x}_{i_1}, \ldots, \widetilde{x}_{i_r}\} \subseteq \widetilde{X}$ denote a set of mergable states of $\widetilde{FI}_{rest}$. The merge of the states in $\widetilde{Z}_i$ to state $\widetilde{z}_i$ results in the finite state machine $\widetilde{FI}_{mgr} = \langle \widetilde{X}_{mgr}, \Pi, \widetilde{\Sigma}, \widetilde{x}_{0_{mgr}}, \widetilde{R}, \widetilde{T}_{mgr} \rangle$, where

$$\begin{aligned} \widetilde{X}_{mgr} = \;& \widetilde{X} \;\cup\; \widetilde{z}_i \setminus \widetilde{Z}_i \\ \widetilde{x}_{0_{mgr}} = \;& \begin{cases} \widetilde{z}_i & \text{if } \widetilde{x}_0 \in \widetilde{Z}_i \\ \widetilde{x}_0 & \text{otherwise} \end{cases} \\ \widetilde{T}_{mgr} = \;& \widetilde{T} \;\setminus\; \{\widetilde{x}_a \overset{o}{\rightarrow} \widetilde{x}_b \mid \widetilde{x}_a \in \widetilde{Z}_i \text{ or } \widetilde{x}_b \in \widetilde{Z}_i\} \\ & \cup\; \{\widetilde{x}_a \overset{o}{\rightarrow} \widetilde{z}_i \mid \widetilde{x}_a \overset{o}{\rightarrow} \widetilde{x}_b \in \widetilde{T} \text{ with } \widetilde{x}_b \in \widetilde{Z}_i\} \\ & \cup\; \{\widetilde{z}_i \overset{o}{\rightarrow} \widetilde{x}_b \mid \widetilde{x}_a \overset{o}{\rightarrow} \widetilde{x}_b \in \widetilde{T} \text{ with } \widetilde{x}_a \in \widetilde{Z}_i\} \end{aligned}$$

Definition 9. (Valid mergable state partition). A state partition $\widetilde{X}_1, \ldots, \widetilde{X}_s = \{\widetilde{x}_{1_1}, \ldots, \widetilde{x}_{1_r}\}, \ldots, \{\widetilde{x}_{s_1}, \ldots, \widetilde{x}_{s_r}\}$ of the failure identifier $\widetilde{FI}_{rest} = \langle \widetilde{X}, \widetilde{F}, \widetilde{\Sigma}, \widetilde{x}_o, \widetilde{R}, \widetilde{T} \rangle$ is called mergably valid iff the following condition holds:

$$\begin{aligned} valid(\widetilde{X}_1, \ldots, \widetilde{X}_s) \;\leftrightarrow\; & merg(\widetilde{X}_i) \; \forall i = 1 \ldots s \text{ and} \\ & \forall \widetilde{X}_i, \; \forall \widetilde{x}_i, \widetilde{x}_j \in \widetilde{X}_i, \; \forall o \in \widetilde{\Sigma} \\ & \widetilde{x}_i \overset{o}{\rightarrow} \widetilde{x}'_i \in \widetilde{T} \text{ and } \widetilde{x}_j \overset{o}{\rightarrow} \widetilde{x}'_j \in \widetilde{T} \\ & \quad \text{implies } \exists \widetilde{X}_j \text{ s.t. } \widetilde{x}'_i, \widetilde{x}'_j \in \widetilde{X}_j \end{aligned}$$

The mergable state partition condition verifies that the resulting finite state machine is deterministic. Referring to the failure identifier $\widetilde{FI}_B$ depicted in Figure 2, the state partition $\{b_0, b_1\}, \{b_2, b_3\}, \{b_4\}, \{b_5, b_6\}, \{b_7\}$ is mergably valid. The merge of these states leads to the finite state machine $\widetilde{FI}_Z$ which is shown in Figure 4.

Proposition 1. (Merging states leads to a failure identifier)
Let $\widetilde{Z}_1, \ldots, \widetilde{Z}_s = \{\widetilde{x}_{1_1}, \ldots, \widetilde{x}_{1_r}\}, \ldots, \{\widetilde{x}_{s_1}, \ldots, \widetilde{x}_{s_r}\}$ denote a valid state partitioning of the failure identifier $\widetilde{FI}_O = \langle \widetilde{X}, \widetilde{F}, \widetilde{\Sigma}, \widetilde{x}_o, \widetilde{R}, \widetilde{T} \rangle$. The finite state machine $\widetilde{FI}_{mgr} = \langle \widetilde{Z}, \widetilde{F}, \widetilde{\Sigma}, \widetilde{x}_{0_{mgr}}, \widetilde{R}, \widetilde{T}_{mgr} \rangle$ derived from merging all states of set $\widetilde{Z}_i$ to state $\widetilde{z}_i$ for all sets of the partition is a failure identifier.

Proof 1.

$$\begin{aligned} & \forall S = [o_1, \ldots, o_k] \text{ such that } psbl(S) \\ & \quad \exists \widetilde{p} = \widetilde{x}_0 \overset{o_1}{\rightarrow} \widetilde{x}_{j_1} \cdots \overset{o_{k-1}}{\rightarrow} \widetilde{x}_{j_{k-1}} \overset{o_k}{\rightarrow} \widetilde{x}_{j_k} \text{ in } \widetilde{FI}_{rest} \text{ with} \\ & \qquad \widetilde{R}(\widetilde{x}_{j_k}) = \Big\{ \bigcup_j \text{Event}(P_{F_j}) \mid \\ & \qquad\qquad P_{F_1} \overset{o_1}{\rightarrow} P_{F_2} \cdots \overset{o_{k-1}}{\rightarrow} P_{F_k} \overset{o_k}{\rightarrow} x \\ & \qquad\qquad \text{is path in } G \text{ with Start}(P_{F_1}) = x_0 \Big\} \\ & \qquad\quad (\text{since } \widetilde{FI}_{rest} \text{ is a failure identifier}) \\ & \text{And therefore} \\ & \quad \exists \widetilde{p}_{mgr} = \widetilde{z}_0 \overset{o_1}{\rightarrow} \widetilde{z}_{j_1} \cdots \overset{o_{k-1}}{\rightarrow} \widetilde{z}_{j_{k-1}} \overset{o_k}{\rightarrow} \widetilde{z}_{j_k} \text{ in } \widetilde{FI}_{mgr} \text{ with} \\ & \qquad \widetilde{x}_{j_i} \in \widetilde{Z}_{j_i} \; \forall \widetilde{x}_{j_i} \text{ and } \widetilde{R}(\widetilde{z}_{j_k}) = \widetilde{R}(\widetilde{x}_{j_k}) \\ & \qquad\quad (\text{see definition 9}) \end{aligned}$$

■

Hence the size of a restricted failure identifier can be reduced without loosing its property of efficiently retrieving all failure sets consistent with a sequence of observations.

Definition 10. (Reduced failure identifier). A failure identifier $\widetilde{FI}_{red}$ of a system G is called *reduced* if it does not contain any states that can be merged.

The reduced failure identifiers of a system are not uniquely defined. Figure 4 illustrates the smallest reduced failure identifiers for the system shown in Figure 1.

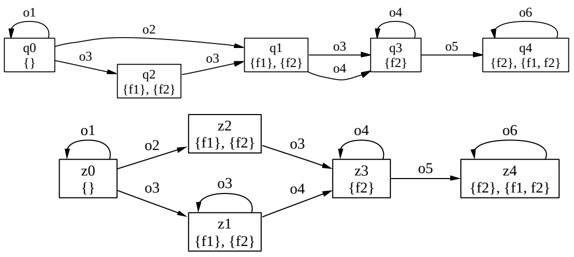


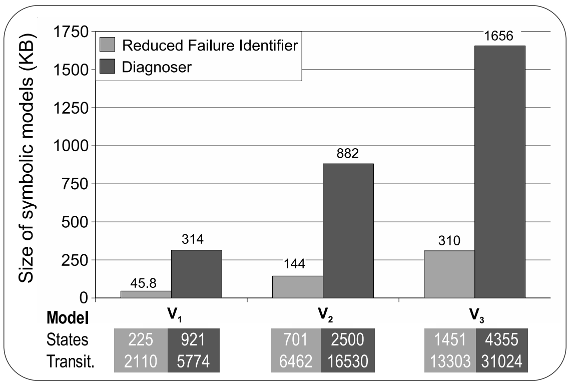
Figure 4. Smallest reduced failure identifiers $\widetilde{FI}_Q$ (on top) and $\widetilde{FI}_Z$ (on bottom) for the system depicted in Figure 1.

6. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In order to determine the decrease of size of a reduced failure identifier in comparison to the classical diagnoser we chose to implement the state merge procedure symbolically in C++ using the efficient symbolic framework presented in (Schumann *et al.*, 2004). The procedure has been tested based on an example derived from a telecommunication network application consisting of a switch and two control stations (Rozé and Cordier, 2002).

In this example, there are 9 observable events, 6 failure types, and 8 other unobservable events. The switch model has 12 states and 18 transitions, the primary control station 13 states and 15 transitions and the backup control station 19 states and 28 transitions. This yields a global model of 1062 states and 2911 transitions. In order to observe how the two approaches scale, two "lighter" versions of the example, where groups of failure types are fusioned are also considered. This yields 3 different Versions $V_1 \ldots V_3$, with a number of failure types ranging from 3 to the original 6.

Figure 5 shows for each of the 3 example versions the size of the diagnoser in comparison to the size of a reduced failure identifier. In all cases the reduced failure identifiers are considerably smaller. This results on the one hand from the reduction of the model's states and transitions and on the other hand from the fact that the states of the failure identifier are only labelled by the consistent failure sets and not additionally by the system states as it is the case for the diagnoser state labels. Note that the diagnoser reduction is considerably faster than its computation. Thus the size decrease gained from the reduction can be achieved at little expense.



Model	V_1		V_2		V_3	
States	225	921	701	2500	1451	4355
Transit.	2110	5774	6462	16530	13303	31024

Figure 5. Size of reduced failure identifiers in comparison to the diagnoser size

7. CONCLUSION AND FUTURE WORK

The paper defines a finite state machine, the failure identifier, that allows the efficient identification of possible failures that are consistent with a sequence of observations. The failure identifiers of a system are not uniquely determined and can differ in the size and the set of observable sequences for which they are defined. It has been shown how the restricted failure identifier can be derived, that is the smallest failure identifier that is defined only for the observable behaviour of the system and it has been described how a failure identifier with even smaller size can be obtained by merging states and thus generalising the observable behaviour.

In the future we plan to analyse whether one can compute a reduced failure identifier with minimal size based on the restricted failure identifier and aim to determine heuristics for the state merge that will lead to the efficient computation of small reduced failure identifiers.

Furthermore we plan to extend the failure identifier approach to a decentralised setting. Rather than precomputing the failure identification information for the whole system, we plan to precompute them for local subsystems and to combine this information as the observations become available. By relying on failure identifiers rather than diagnosers we want to reduce the space and possibly the time[4] complexity of the diagnosis approach.

REFERENCES

Christensen, Soren, Hans Huttel and Colin Stirling (1992). Bisimulation equivalence is decidable for all context-free processes. In: *International Conference on Concurrency Theory*. pp. 138–147.

Gold, E. M. (1972). System identification via state characterization. *Automatica* **8**, 621–636.

Hennessy, J. and R. Milner (1985). Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery* **32**(1), 137–161.

Lamperti, G. and M. Zanella (2001). *Diagnosis of active systems*. Kluwer Academic Publishers.

Milner, R. (1980). A calculus of communicating systems. *Lecture Notes in Computer Science*.

Park, D. M. R. (1981). Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science* **104**, 167–183.

Pencolé, Y., M. O. Cordier and L. Rozé (2002). A decentralized model-based diagnostic tool for complex systems. *International Journal on Artificial Intelligence Tools* **11**(3), 327–346.

Rozé, L. and M. O. Cordier (2002). Diagnosing discrete-event systems : extending the "diagnoser approach" to deal with telecommunication networks. *Journal on Discrete-Event Dynamic Systems : Theory and Applications* **12**(1), 43–81.

Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis (1996). Failure diagnosis using discrete event models. *IEEE Transactions on Control Systems Technology* **4**(2), 105–124.

Schumann, A., Y. Pencolé and S. Thiébaux (2004). Diagnosis of discrete-event systems using binary decision diagrams. In: *15th International Workshop on Principles of Diagnosis – DX'04*. pp. 197–202.

Zad, S. H., R. H. Kwong and W. M. Wonham (2003). Fault diagnosis in discrete-event systems: Framework and model reduction. *IEEE Transactions on Automatic Control* **48**(7), 1199–1212.

[4] Due to the smaller model size (in comparison to the diagnoser) more local models can be merged off-line, which leads to a decrease in time complexity (Pencol´e *et al.*, 2002).