On the construction of modular observers and diagnosers for discrete-event systems*

S.L. Ricker
CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

E. Fabre
IRISA-INRIA
Campus de Beaulieu
35042 Rennes Cedex
France

E-Mail: S.L.Ricker@cwi.nl, Eric.Fabre@irisa.fr

Abstract

A modular approach for designing observers and diagnosers for detecting faults in large distributed systems is presented. In particular, the model is based on a discrete-event system framework. The model assumes that the system is composed of distributed components that interact with each other via sets of common resources. Modularity can be achieved by imposing a total ordering on access to the common resources. Further, a component's own access to a common resource is an observable event. It is then possible to partition component behavior such that an observer of a component's behavior need not be aware of all the behavior of other components that use the same resources. Procedures for the design of such observers and the subsequent construction of diagnosers are given.

1. Introduction

Diagnosis of the breakdown of large distributed systems is often made more complex by the need to construct a global model of system behavior. This becomes especially problematic for those systems that also exhibit dynamic behavior in the form of the addition or removal of system components. We are interested in the diagnosis of distributed systems where some components interact or cooperate. Moreover, the addition of new components or the removal of components should have a minimal effect on existing diagnosers.

Our interest in constructing diagnosers for this class of systems is motivated by a specific network management application, initially presented in [4]. In this particular network, alarms are emitted when a fault occurs in the system. A distributed agent sees sequences of alarms (not the faults) and in the event that one alarm corresponds to the occurrence of several different faults, the agent must determine which fault or faults could have taken place. The results of [4] considered the construction of a centralized diagnoser. In this paper, we are concerned with the construction of modular diagnosers for our network management problem.

One aspect of our system is the presence of shared resources. Shared resources in discrete-event systems was considered in [6]. Different components of the system interact with each other by accessing such resources. We can express a component's overall behavior in terms of its own behavior and its interactions with other components. It is this separation of behavior that allows us to take advantage of a modular modeling approach and avoid the reconstruction of a global state. In addition, we assume that there are no system-wide faults to diagnose.

Another characteristic of the system is its dynamic nature. That is, components can be added to and removed from the network. When a new component is introduced to the system, the local diagnosers for those components that do not interact with the new component should be effectively oblivious to the change in the system. Thus we seek a strategy for producing a local diagnoser for each component that can be constructed without taking into consideration the behavior of the entire system. We want the collection of modular diagnosers to detect the same faults as a centralized diagnoser.

In the network application under consideration here, we initially assume that alarms are not lost. That is, when a fault occurs a corresponding alarm is emitted and detected. Although the diagnosers observe all the alarms, there is still uncertainty in the observations. Since an alarm can correspond to the occurrence of more than one fault, it may not

*This work is partially supported by an ERCIM fellowship for the first author and by RNRT (National Research Network in Telecommunication) through the MAGDA project (Modeling and Learning for Distributed Management of Alarms). For more information see http://magda.elibel.tm.fr/.

be clear which fault has given rise to the observed alarm.

We use a discrete-event system approach to diagnose the faults in a telecommunication system. Notably, we do not take timing information of the alarms into account when producing a diagnosis. Diagnosis of untimed discrete-event systems include [3, 7, 9]. In addition, communicating finite-state machines (e.g., [1]) have been used to construct modular diagnosers for large systems.

2. Background

In this paper, we adopt the control-theoretic framework of [8] to examine the diagnosis of distributed systems. This formalism views the discrete-event system as a generator of a formal language. Refer to [2] for an introduction to the area of discrete-event control theory.

The system we are interested in diagnosing is composed of a set of n components. Each component i (for $i = 1, \ldots, n$) is formally modeled as a (possibly) non-deterministic automaton $C_i = (X_i, \Sigma_i, \delta_i, x_{i,0})$, where X_i is the set of states; Σ_i is the set of events; δ_i is the transition function, a partial function $\delta_i : \Sigma_i \times X_i \rightarrow 2^{X_i}$; and $x_{i,0}$ is the initial state.

As the system evolves, it produces sequences of events. It may be the case that an observer of C_i no longer uniquely distinguishes every event in Σ_i, but instead observes events in some set $\Sigma_{i,o} \subset \Sigma_i$. For purposes of this paper, we describe an observer's view of the system using a *mask* function $M_i : \Sigma_i^* \rightarrow \Sigma_{i,o}^*$. This function effectively either replaces or erases those elements σ from a sequence s that are not found in the set of observable events $\Sigma_{i,o}$. In particular, $M_i(\sigma)$ represents an observer's view of the event σ where $M_i(\sigma) = \sigma$, if $\sigma \in \Sigma_{i,o}$, or $M_i(\sigma) \in \Sigma_{i,o} \cup \{\varepsilon\}$ if $\sigma \notin \Sigma_{i,o}$, where ε is the empty string.

When an observer of a system C_i no longer uniquely distinguishes certain events in Σ_i, we can build an observer automaton, denoted by $Obs(C_i)$ to capture this partial view of C_i. That is, when an observer cannot determine exactly to which state of C_i a particular sequence of events leads, it estimates the current state of the the system based on its current observation. For instance, suppose that an observer of some C_i cannot distinguish event b from event d nor event c from g and it observes a sequence of events abc. If the observer considers it possibe that either sequence adc (leading to state 4 in C_i) or abg (leading to state 6 in C_i) has really occurred, then its state estimate after observing abc is the set $\{4, 6\}$.

A diagnoser for C_i, denoted $Diag(C_i)$, is constructed from its observer $Obs(C_i)$. The state space of $Diag(C_i)$ contains additional information about the occurrence of failures in the system [9]. If a state of the diagnoser is appended with a label Y, this is interpreted as "there is a path to this state along which a failure event has occurred" while a label of N corresponds to "normal" system behaviour (i.e., no fault has occurred along a path to this state). Note that while the label set in [9] is expanded from the set $\{Y, N\}$to include references to failure classes, for our purposes we will restrict ourselves to the simpler label set.

3. Modular decomposition for a distributed system

Our interest in diagnosing distributed systems arises from an application analyzing faults in a telecommunication network. In particular, we must produce a diagnosis of faults that occur across a distributed architecture. The distributed system is described in terms of components where some components may share resources (e.g., equipment) and where components may be added to or removed from the network.

The behavior of component i (for $i \in \{1, \ldots, n\}$) is modeled as an automaton C_i where faults occur at states and alarms are represented as transitions out of faulty states. The event set Σ_i of component i is partitioned into two disjoint sets: those events that are involved in the access of a common resource, denoted by $\Sigma_{i,g}$, and those that are not, denoted by $\Sigma_{i,\ell}$. Events in $\Sigma_{i,g}$ are called *global events* and events in $\Sigma_{i,\ell}$ are called *local events*. Some events in Σ_i denote alarms which arise from the occurrence of a fault. There is a set of faulty states $X_{i,f} \subseteq X_i$. We further assume that a components do not observe each other's behavior (i.e., $\Sigma_i \cap \Sigma_j = \emptyset$ for $i, j \in \{1, \ldots, n\}$ and $i \neq j$).

In the general case, components may interact with many other components. This behavior is modeled by an "interaction" automaton that describes the interaction between k components, where $0 \leq k \leq n$. The event set of an interaction automaton for k components is the union of the k global event sets. The overall behavior of component i is then the parallel composition of C_i and all interaction automata in which component i collaborates. Here we consider the case when $k = 2$.

Formally, when $k = 2$, the interaction automaton for two components i and j is given by $C_{ij} = (X_{ij}, \Sigma_{i,g} \cup \Sigma_{j,g}, \delta_{ij}, x_{o,ij})$, where $i, j \in \{1, \ldots, n\}$ and $i \neq j$, and $X_{ij,f} \subseteq X_{ij}$ is the set of faulty states

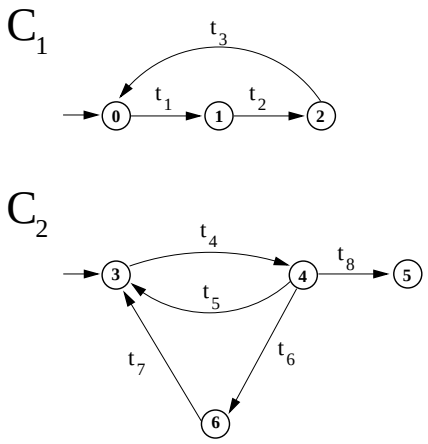

Figure 1: Two components of a distributed system.

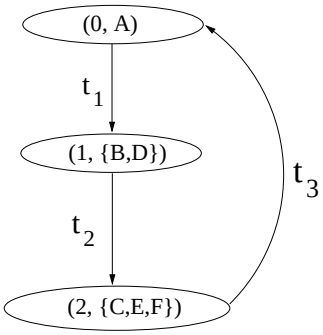

Figure 3: $Obs(C_1)$ for the components in figures 1 and 2.

involved in accessing the common resources. We assume that we are given this automaton and note that C_{ij} imposes a total order on i and j's access to the common resources. For the remainder of this paper, we will assume that $n = 2$.

It is the separation of behavior into local and global events coupled with the assumption that a component does not observe events in any other component's event set that allows us to design modular diagnosers. A modular diagnoser for component i needs to detect faults arising from local alarms (i.e., an event in $E_{i,\ell}$), faults arising from global alarms (i.e., an event in $E_{i,g}$). A strategy is presented in [5] for reconstructing the causality of alarms that arise from several components accessing a set of common resources. This is necessary when different patterns of the same alarms indicate the occurrence of different faults.

3.1. Modular observers

We want to design observers of the behavior of each component and its interactions with other components. Using the observers, we can subsequently construct diagnosers. An observer of the behavior of a component is aware of the events corresponding to alarms but does not see the fault occur. It could be the case that two different faults give rise to the same alarm or that more than one alarm is associated with a given fault (and an observer cannot distinguish between these alarms). We assume that observers of the components have a partial view in the sense described above of sequences of events that characterize the overall behavior of a component.

Consider the two components C_1 and C_2 in figure 1. All the events in component C_1 are global while component C_2 has both global and local events. The interaction between C_1 and C_2 is shown in figure 2. We assume that the interaction between C_1 and C_2 is identical to the interaction between C_2 and C_1 and therefore $C_{12} = C_{21}$. The behavior of the entire system is described by the parallel composition of C_1, C_{12} and C_2. We will construct an observer for component 1's overall behavior $C_1||C_{12}$ and an observer for component 2's overall behaviour $C_{12}||C_2$. Note that we invert the order of composition for component 2. This will make it easier to compare the collection of modular diagnosers to the performance of a centralized diagnoser for $C_1||C_{12}||C_2$.

An observer of component i sees all global and local transitions associated with $C_i||C_{ij}$, for any component j that shares resources with component i. An observer i does not see the global and local transitions of other components. Additionally, we assume that no alarms are lost. Thus the ambiguity of an observer i is derived from either the global transitions of another component j that occur in C_{ij} or from the inability of an observer to uniquely distinguish some its own local and global transitions (as described by the mask function M_i).

An observer of the behavior of $C_i||C_{ij}$ is an automaton we $Obs(C_i)$. Formally, $Obs(C_i) = (Q, \Sigma_{i,o}, \delta_{obsi}, q_0)$. Note that although the event set of C_{ij} includes the global events of component j, these events are unobservable to component i and thus the observer is constructed purely from observations of events in $\Sigma_{i,o}$. Since we assume that mutual exclusion is satisfied, prior to generating observers, component i does not need to be aware of the exact local state that component j is in. The observer does, though, keep track of the possible states in C_{ij}. Therefore, a state in $Obs(C_i)$ has the form (x, q), where $x \in X_i$ and $q \in Q$. Observers for $C_1||C_{12}$ and $C_{12}||C_2$ are shown in figures 3 and 4, respectively.

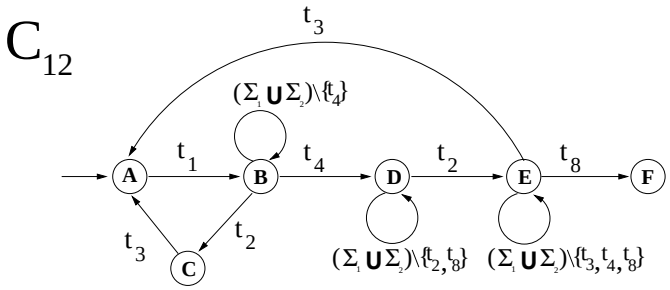

Figure 2: The interaction between the two components in figure 1.

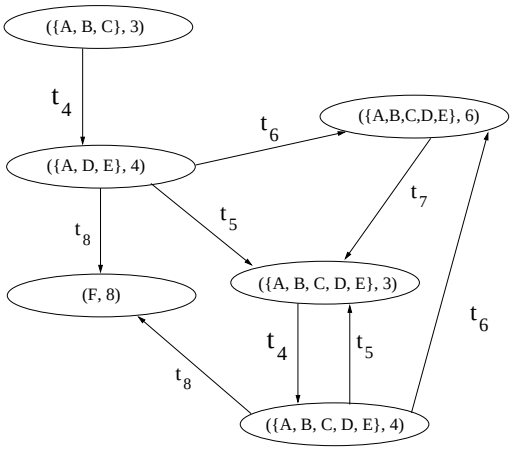

Figure 4: $Obs(C_2)$ for the components in figures 1 and 2.

3.2. Constructing modular diagnosers

We depart from some of the specifics of the formulation of the diagnosers introduced in [9]. We incorporate the idea that diagnosers are constructed from observers by augmenting states with labels that indicate whether or not a failure has occurred. In this paper, we are not so much interested in the exact flavor of the failure as much as we want to know if the system has passed through a faulty state.

We indicate the possibility of the system having passed through a faulty state at state by appending a "Y" after a state estimate if a fault in component i has occurred and "N" after a state estimate if a fault has not occurred [2].

A diagnoser for $C_i||C_{ij}$, denoted $Diag(C_i)$, is an automaton that is constructed from $Obs(C_i)$. Formally, $Diag(C_i) = (D_i, \Sigma_{i,o}, \delta_{diagi}, d_0)$. Recall that a state in $Obs(C_i)$ has the form $\{(x, x',) \mid x \in X_i \text{ and } x' \in X_{ij}\}$. A state in the corresponding diagnoser augments these states with labels to indicate whether or not a given local state $x \in X$ and/or global state $x' \in X_{ij}$ is reachable from a faulty state.

To describe the construction of the states of the diagnoser, we adopt the algorithm for constructing diagnosers from [2]:

Procedure 1 *Constructing Diagnosers*

1. Given: $C_i||C_{ij}||C_j$.
2. Begin with the initial state of q_0 of $Obs(C_i)$.
   (a) Attach the label N to all $q' \in q_0$ that can be reached from the initial state in $C_i||C_{ij}||C_j$ by unobservable sequences in $(\Sigma \setminus \Sigma_{i,o})^*$ such that the sequence does not pass through a state $(a, b, *)$ where in $a \in X_{i,f}$ and/or $b \in X_{ij,f}$ and * represents "any state" in X_j.
   (b) Attach the label Y to all $q' \in q_0$ that can be reached from the initial state of $C_i||C_{ij}||C_j$ by unobservable sequences in $(\Sigma \setminus \Sigma_{i,o})^*$ such that the sequence does pass through a state $(a, b, *)$ where in $a \in X_{i,f}$ and/or $b \in X_{ij,f}$ and * represents "any state" in X_j.
   (c) If $q' \in q_0$ can be reached via a faulty state and a non-faulty state, create two entries for inclusion in d_0, namely xN and xY.
3. The remaining states of the diagnoser are constructed by appending labels to observer states (following the transition function of $Obs(C_i)$) and propagating the label Y. That is, any state z reachable from state xY receives the label Y to note that a faulty state was reached on some path to z.

3.3. Composing modular diagnosers

We want to ensure that our modular diagnosers for $C_1||C_{12}$ and $C_2||C_{12}$ produce equivalent information to a "centralized" diagnoser for $C_1||C_{12}||C_2$. By centralized diagnoser, we mean a diagnoser that is constructed for the complete behavior of components 1 and 2. We denote the composition of the two local

diagnosers as the automaton $Diag_{gbl}$. The construction of $Diag_{gbl}$ is described in the following procedure.

Procedure 2 *Composition of local diagnosers* $Diag(C_1)$ *and* $Diag(C_2)$

1. Let $\mathcal{DL} = \{Y, N\}$ be a set of labels for the diagnoser.

2. The initial state of $Diag(C_1)$ is $d_{1,0}$ and is a set of states of the form $(a\ L, d\ L, *)$, where $a \in X_i$, $d \in X_{ij}$ and $L \in \mathcal{DL}$.

3. The initial state of $Diag(C_2)$ is $d_{2,0}$ is a set of states of the form $(*, d'\ L, b\ L)$, where $d' \in X_{ij}$, $b \in X_j$ and $L \in \mathcal{DL}$.

4. The function merge(ℓ_1, ℓ_2), is a partial function merge:$D_1 \times D_2 \rightarrow (X_1 \times \mathcal{DL}) \times (X_{12} \times \mathcal{DL}) \times (X_2 \times \mathcal{DL})$ and is defined as follows:

   (a) Let $D = \{d \mid (a\ L, d\ L, *) \in \ell_1\}$ and $D' = \{d' \mid (*, d'\ L, b\ L) \in \ell_2\}$

   (b) The output of *merge*:

$$merge(\ell_1, \ell_2) := \begin{cases} \{(a\ L, \hat{d}\ L, b\ L) \mid \\ (a\ L, \hat{d}\ L, b\ L) \in \ell_1 \cup \ell_2 \\ \text{ and } \hat{d} \in D \cap D'\} \\ \text{if } D \cap D' \neq \emptyset; \\ \textit{undefined} \text{ otherwise.} \end{cases}$$

5. The initial state of $Diag_{gbl}$ is the result of performing the *merge* function on $d_{1,0}$ and $d_{2,0}$. Subsequent states are constructed by applying the transition function (defined below) to the initial state.

6. The transition function is defined as follows where $\sigma \in \Sigma$ and $merge(\ell_1, \ell_2)$, where $\ell_1 \in D_1$ and $\ell_2 \in D_2$, is defined:

$$\delta^{Diag_{gbl}}(merge(\ell_1, \ell_2), \sigma) := \begin{cases} merge(\delta_{obs1}(\ell_1, \sigma), \ell_2) & \text{if } \sigma \in \Sigma_{1,g} \text{ and } \delta_1(a, \sigma), \delta_{12}(c, \sigma) \text{ are defined;} \\ merge(\delta_{obs1}(\ell_1, \sigma), \ell_2) & \text{if } \sigma \in \Sigma_{1,\ell} \text{ and } \delta_1(a, \sigma) \text{ is defined;} \\ merge(\ell_1, \delta_{obs2}(\ell_2, \sigma)) & \text{if } \sigma \in \Sigma_{2,g} \text{ and } \delta_{12}(c', \sigma), \delta_2(b, \sigma) \text{ are defined;} \\ merge(\ell_1, \delta_{obs2}(\ell_2, \sigma)) & \text{if } \sigma \in \Sigma_{2,\ell} \text{ and } \delta_2(b, \sigma) \text{ is defined;} \\ \textit{undefined} & \text{otherwise.} \end{cases}$$

4. Discussion

We were able to formulate modular diagnosers for our network management system because there are interactions between the components in the distributed system. The *a priori* assignment of this interaction may not be possible to describe in other application domains. In addition, we have assumed here that all alarms arising from faults in the system are observable, although it is possible that there is uncertainty surrounding the identity of the alarm. A necessary extension of the strategy must allow for the loss of alarms.

Another question to address in future work is how this strategy for modular diagnosis can be applied to systems where failure events are unobservable and hence no observable alarms herald the occurrence of a fault (e.g., [9, 3]). We conjecture that as long as the overall behavior of each component satisfies the standard notions of diagnosability, the total ordering of access to common resources may be enough to ensure that the modular diagnosers produce equivalent diagnoses to a centralized diagnoser.

References

[1] P. Baroni, G. Lamperti, P. Pogliano, and M. Zanella. Diagnosis of large active systems. *Artificial Intelligence*, 110:135–183, 1999.

[2] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer, 1999.

[3] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *J. DEDS*, 10(1/2):33–86, 2000.

[4] E. Fabre, A. Aghasaryan, A. Benveniste, R. Boubour, and C. Jard. Fault detection and diagnosis in distributed systems: an approach by partially stochastic petri nets. *J. DEDS*, 8:203–231, 1998.

[5] E. Fabre, A. Benveniste, C. Jard, L. Ricker, and M. Smith. Distributed state reconstruction for discrete-event systems. In *Proc. 39th Conf. Decision and Cont.*, 2000. Forthcoming.

[6] N.B. Hadj-Alouane, S. Lafortune, and F. Lin. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *J. DEDS*, 6:379–427, 1996.

[7] F. Lin. Diagnosibility of discrete-event systems and its applications. *Journal of Discrete Event and Dynamical Systems*, 4(2):197–212, 1994.

[8] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Contr. Optim.*, 25(1):206–230, 1987.

[9] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosibility of discrete event systems. *IEEE Trans. Automat. Contr.*, 40(9):1555–1575, 1995.