Assistance for the design of a diagnosable component-based system

Yannick Pencolé
Computer Science Laboratory and NICTA Knowledge Representation and Reasoning,
RSISE, The Australian National University, Canberra, ACT 0200, Australia
Yannick.Pencole@anu.edu.au

Abstract

Diagnosability of component-based systems is a property that characterises the ability to diagnose fault events given a flow of observations. In this paper, we use model-based reasoning techniques and we propose a theoretical framework to analyse diagnosability in a decentralised way. We then introduce an algorithm that performs diagnosability analyses and provides useful information for the design of a diagnosable component-based system.

1. Introduction

Monitoring and diagnosing large systems like telecommunication networks, web services, business processes are challenging tasks. Such systems are component-based, each component communicating with other components of the system. When a system operates, some critical events or *faults* may occur. The system supervisor has to detect them and to make decisions to keep the system working, so an automatic monitoring is required. This problem has been studied for many years in both AI community [4, 9, 11] and Control community [6]. In these previous works, the objective is to model the system and to apply monitoring algorithms on it but they all share the same weakness: none of them takes into account any diagnosablility issue about the system. If a diagnosability analysis is performed on the system, then the diagnosis algorithm is more efficient and less costly because new information is taken into account before implementing it.

In this paper, we adopt the following point of view. Since diagnosability is a key issue for system monitoring, diagnosability analysis has to be performed during the system design. Analysing the diagnosability at the design stage has two consequences. Firstly, it guarantees that observation protocols have been specified so that any critical events or faults can be diagnosed with certainty after the deployment of the system. Secondly, such an analysis provides information for the design of the application in charge of monitoring the specified system.

Designing a component-based system is a difficult task because of the architectural and distributed nature of such a system. As a consequence, component-based systems cannot be designed without the use of assistant tools that perform automatic analyses like, for instance, model-checking tools [2] or testing tools [7]. In this paper, we present such an assistant to incorporate a diagnosability analysis at the design stage.

The paper is organised as follows. The first section presents the general architecture of the design assistant. Section 3 describes the kind of systems we tackle with and how they are specified; then the diagnosability problem on that kind of systems is introduced. Section 4 describes the theoretical framework of the decentralised diagnosability analysis and the notion of undiagnosable scenarios. Section 5 explains the diagnosability algorithm implemented in the design assistant and experimental results are provided in Section 6. The set of related works is presented in Section 7.

2. General architecture of the design assistant

The purpose of the diagnosability assistant is to provide an interactive design architecture of component-based systems which allows the specification of a diagnosable system (see Figure 1).

The *Designer* is a human agent who is responsible for the design of a component-based system. This Designer provides a description of the system (also called a specification) as a set of components. Each component has an event-driven behaviour and can interact with other components. For the sake of simplicity, we will consider in the following that the specification formalism is based on classical automata. This assumption is not restrictive because any high level specification language for component-based discrete event systems can be translated to automata and vice-versa.

Once the Designer has produced a specification of its system, the assistant operates as follows. The Designer provides the specification and queries about the diagnosability

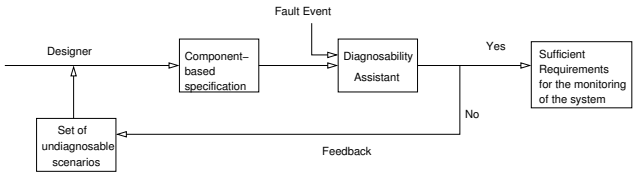

Figure 1. General architecture of the design assistant

analysis of a given event F described in the specification. If the assistant provides the answer *Yes*, then it means that the event F is diagnosable in a given subsystem and the assistant is able to provide some sufficient requirements for the monitoring and the diagnosis of this fault event. If the event F is not diagnosable, then the assistant provides a set of scenarios that could happen in the system and that are the reasons why the event F is not diagnosable. The objective of the Designer is then to modify the specification in order to eliminate these scenarios.

3. Background

3.1. System model

We study systems that evolve with the occurrence of events. The model a component-based system is based on classical automata: one automaton represents the behaviour (also called the *local model*) of one component [13]. This formalism is aimed at modelling any discrete event systems with multiple and permanent faults [11]. A fault occurs in one component and its consequences may propagate in other components.

Definition 1 (Local model) *A* local model Γ_i *is an automaton* $\Gamma_i = (Q_i, \Sigma_i, T_i, q_{0i})$ *where:*

- Q_i *is a finite set of states;* q_{0i} *is the initial state;*
- Σ_i *is the set of events and* $T_i \subseteq Q_i \times E_i \times Q_i$ *is the set of transitions.*

The set of events is divided in four disjoint subsets ($\Sigma_i = \Sigma_i^{obs} \oplus \Sigma_i^{com} \oplus \Sigma_i^{norm} \oplus \Sigma_i^{flt}$): Σ_i^{obs} the set of *observable events*, Σ_i^{com} the set of *communication events*, Σ_i^{flt} the set of *fault events* and Σ_i^{norm} the set of *normal events*. We also consider that the empty event ϵ as a label of a transition $s \xrightarrow{\epsilon} s$ for each state s of the local model. Such a transition expresses asynchronism between two local models.

Figure 2 presents a system composed of three local models defined as above. The events f_i are the fault events, the events c_i are the communication events and the events o_i are

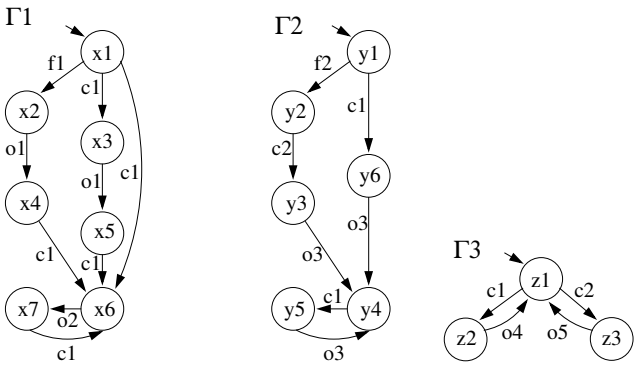

Figure 2. Component-based system modelled with three automata

the observable events.[1] In this example, f_1 and then o_1 can occur on Γ_1. Then, a communication event c_1 can occur, this event being shared by all the components. After the occurrence of this sequence of events, Γ_1 is in state x_6, Γ_2 is in state y_6 and Γ_3 is in state z_2. The behaviour of the global system is formally described by the set of local models and the composition operator $\|_{com}$ (see appendix A).

We call a *subsystem* γ any non-empty set $\{\Gamma_{i_1} \dots \Gamma_{i_m}\}$ of components of the system. The *global model* of γ (denoted $\|\gamma\|$) is the automaton: $\|\gamma\| = \Gamma_{i_1} \|_{com} \dots \|_{com} \Gamma_{i_m}$. The global model of the system Γ is: $\|\Gamma\| = \Gamma_1 \|_{com} \dots \|_{com} \Gamma_N$. If γ contains one component, the global model $\|\gamma\|$ is the local model of the component.

3.2. Diagnosability

Diagnosability of discrete-event systems is a crucial problem. Diagnosability is a property that measures the ability of a monitoring application to diagnose faults inside the supervised system. The deployment of a monitoring application (algorithms, sensor placements, integration) strongly depends on how diagnosable the system is. Before the definition of diagnosability, we introduce the notion of scenario.

Definition 2 (Scenario) *A* scenario *in a subsystem* γ *is a transition path of* $\|\gamma\|$ *whose initial state is the initial state of* $\|\gamma\|$.

There are several definitions for diagnosability, we focus on the classical and general definition from [13].[2]

[1] For the sake of simplicity, normal events are not considered in this example. Empty events are not shown on the figures.

[2] Another definition is also given in [3] or in [1], the second definition is too much restrictive for the kind of systems we study.

Definition 3 (Diagnosability) *A fault F occurring in a subsystem γ is* diagnosable *in γ iff, after the occurrence of F, a finite sequence of events from γ is sufficient to diagnose F with certainty.*

Formally, let $p_F s_F$ be a scenario of the global model $\|\Gamma\|$ such that p_F ends with the occurrence of F to the state x_F and s_F is a transition path from x_F, let Σ_γ^{obs} be the set of observable events from γ, F is diagnosable in γ iff:

$\forall$ scenario $p_F s_F, \exists l \in \mathbb{N} : |s_F| \geq l \Rightarrow (\forall s \in \|\Gamma\|$ such that

$$P_{\Sigma_\gamma^{obs}}(s) = P_{\Sigma_\gamma^{obs}}(p_F s_F), F \text{ occurs in scenario } s).$$

This diagnosability definition is equivalent to the one of [13] when the considered subsystem γ is the system Γ itself.

4. Diagnosability analysis: theory

We now turn to the description of the theoretical model-based reasoning framework for solving the diagnosability problem in a subsystem γ for a given fault F. This reasoning is based on the computation of a special finite-state machine called the *twin* diagnoser. Before introducing this machine, we present another machine called the *interactive diagnoser*.

4.1. Interactive diagnoser

We consider the global model of a subsystem γ. The *interactive diagnoser* $\Delta_\gamma(F)$ is an abstraction of $\|\gamma\|$ based on the observable events and the communication events of $\|\gamma\|$ which is able to detect the potential occurrence of the event F relying on the observable events and the communication events. The interactive diagnoser is formally defined as follows. The global model of γ is denoted $\|\gamma\| = (Q_\gamma, \Sigma_\gamma, T_\gamma, x_0)$ with $\Sigma_\gamma = \Sigma_\gamma^{com} \oplus \Sigma_\gamma^{obs} \oplus \Sigma_\gamma^{flt} \oplus \Sigma_\gamma^{norm}$.

Definition 4 (Interactive diagnoser) *The* interactive *F-diagnoser $\Delta_\gamma(F)$ is a finite-state machine $\Delta_\gamma(F) = (Q^\Delta, \Sigma^\Delta, T^\Delta, x_0^\Delta)$ where:*

- $Q^\Delta \subseteq Q_\gamma \times \{F, \neg F\}$; $x_0^\Delta = (x_0, \neg F)$;
- $\Sigma^\Delta = \Sigma_\gamma^{com} \cup \Sigma_\gamma^{obs}$; $T^\Delta \subseteq Q^\Delta \times \Sigma^\Delta \times Q^\Delta$.

The transition $t = (x, f) \xrightarrow{e} (x', f')$ belongs to the diagnoser iff:

1. (x, f) is reachable from x_0^Δ, and
2. there exists a path $x \xrightarrow{uo_1} x_1 \ldots x_n \xrightarrow{e} x'$ in $\|\gamma\|$ with $\{uo_1, \ldots, uo_n\} \subseteq \Sigma_\gamma^{flt} \oplus \Sigma_\gamma^{norm}$ such that: $(F \in \{uo_1, \ldots, uo_n\} \Rightarrow f' = F) \wedge (F \notin \{uo_1, \ldots, uo_n\} \Rightarrow f' = f)$.

Each state of the diagnoser contains a diagnosis information about the occurrence of the fault event F. This automaton represents all the possible behaviours (communication and observable behaviours) of a subsystem. Figure 3 shows the interactive diagnoser computed from the local model Γ_1 for the fault f_1 (see Figure 2), the diagnoser state (x_i, f_1) denotes the fact that Γ_1 in state x_i and f_1 has occurred and the diagnoser state $(x_i, \neg f_1)$ denotes the fact that Γ_1 in state x_i and f_1 has not occurred.

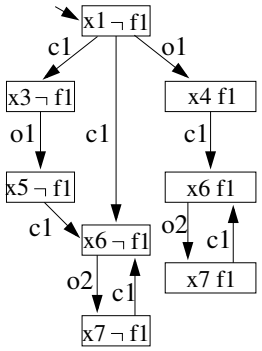

Figure 3. Interactive diagnoser $\Delta_{\Gamma_1}(F)$ based on the local model of the component Γ_1 for the fault f_1.

4.2. Twin diagnoser

Given one unique sequence of events (communication events and observable events), the interactive diagnoser provides a set of diagnoser states (a *belief state*)[3] and informs if the fault F has not happened, may have happened or has definitely happened. If the event sequence produces a belief state where F may have happened, then this sequence cannot provide a diagnosis about F with certainty. The computation of the twin diagnoser for the diagnosability analysis is based on this property of the interactive diagnoser. Formally, the twin diagnoser is obtained by composing this interactive diagnoser with itself.

Definition 5 (Twin diagnoser) *The* twin F-diagnoser *of γ is $\Theta_\gamma(F) = \Delta_\gamma(F) \|_{obs} \Delta_\gamma(F)$.*

Intuitively, the twin diagnoser is obtained by cloning the interactive diagnoser $\Delta_\gamma(F)$ (clone 1 and clone 2) and by using the composition operator $\|_{obs}$ (see appendix **??** for details) to synchronise the observable events of these two automata. Each state x of the twin diagnoser is then composed of two diagnoser states (x_1, f_1) and (x_2, f_2) having their own diagnosis information about F (see Figure 4).

[3] The interactive diagnoser is non-deterministic.

This computation is a way to check if two different event sequences from the interactive diagnoser (communication events and observable events), having the same observable signature (i.e the projection of those sequences on the observable events are identical), provide the same diagnosis information about the occurrence of F.

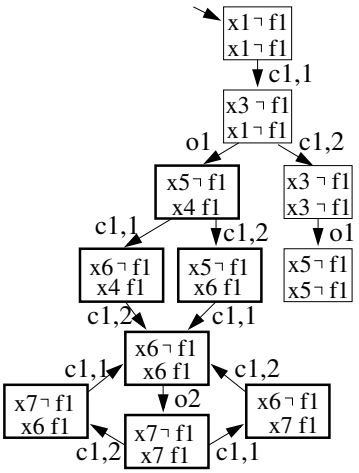

Figure 4. Part of $\Theta_{\Gamma_1}(f_1) = \Delta_{\Gamma_1}(f_1) \|_{obs} \Delta_{\Gamma_1}(f_1)$. Communication event c_i from clone $j \in \{1,2\}$ is denoted ci,j.

Definition 6 (Ambiguity) *A state x of $\Theta_\gamma(F)$ is* ambiguous *iff $x = ((x_1, f_1), (x_2, f_2))$ and $f_1 \wedge f_2$ is not satisfiable.*

Figure 4 shows ambiguous states in bold. Every transition path from the initial state of $\Theta_\gamma(F)$ to an ambiguous state represents two possible scenarios from γ that have the same observable behaviour. Only one of these scenarios contains the fault F, the other one can be either normal (no fault event at all) or faulty (but in this case, fault events of this behaviour are not F).

4.3. Undiagnosable scenarios

When a model of a system (or specification) is not diagnosable, the purpose of the assistant is to provide the reasons of the non-diagnosability. The purpose of our approach is then to provide the *scenarios* that are the causes of the diagnosability problem.

Considering two scenarios with the same observable behaviour, if a fault only occurs in one of the scenarios then the observable behaviour from the scenarios is not sufficient to diagnose the fault with certainty. Two scenarios that have these characteristics are said to be *undiagnosable*. Undiagnosable scenarios are formally defined as follows, with help of a projection operation (see appendix A for details).

Definition 7 (Undiagnosable scenarios) *A couple of scenarios σ_1, σ_2 in a subsystem γ are* undiagnosable *iff: $obs_1 = P_{\Sigma_\gamma^{obs}}(\sigma_1)$ and $obs_2 = P_{\Sigma_\gamma^{obs}}(\sigma_2)$ are such that $obs_1 = obs_2$ and $F \in \sigma_1 \cup \sigma_2 \wedge F \notin \sigma_1 \cap \sigma_2$.*

Figure 5 presents such scenarios. These *undiagnosable scenarios* are strongly linked to the twin diagnoser. This link is given by the following property.

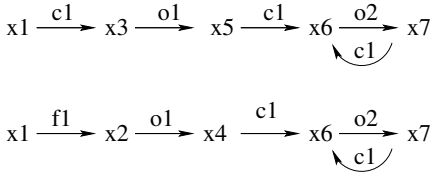

Figure 5. Two undiagnosable scenarios of Γ_1.

Theorem 1 *Two infinite scenarios σ_1, σ_2 in a subsystem γ are* undiagnosable *iff there exists a transition path τ in $\Theta_\gamma(F)$ containing an infinite set of ambiguous states and such that $P_{\Sigma_\gamma^{com} \cup \Sigma_\gamma^{obs}}(\sigma_1) = P_1(\tau)$ and $P_{\Sigma_\gamma^{com} \cup \Sigma_\gamma^{obs}}(\sigma_2) = P_2(\tau)$ where $P_i(\tau)$ is the projection of τ on the events of the clone i in $\Theta_\gamma(F)$.*

Proof:

($\Rightarrow$) σ_1, σ_2 are *undiagnosable* so $P_{\Sigma_\gamma^{obs}}(\sigma_1) = P_{\Sigma_\gamma^{obs}}(\sigma_2)$. By definition of the twin diagnoser$\Theta_\gamma(F)$, there exists a transition path τ such that $P_{\Sigma_\gamma^{com} \cup \Sigma_\gamma^{obs}}(\sigma_1) = P_1(\tau)$ and $P_{\Sigma_\gamma^{com} \cup \Sigma_\gamma^{obs}}(\sigma_2) = P_2(\tau)$. $F \in \sigma_1 \cup \sigma_2 \wedge F \notin \sigma_1 \cap \sigma_2$ therefore τ contains an infinite set of ambiguous states.

($\Leftarrow$) Let τ be a transition path of $\Theta_\gamma(F)$ containing an infinite set of ambiguous states. Let $\mathcal{S}_1 = \{\sigma, P_{\Sigma_\gamma^{com} \cup \Sigma_\gamma^{obs}}(\sigma) = P_1(\tau)\}$ and $\mathcal{S}_2 = \{\sigma, P_{\Sigma_\gamma^{com} \cup \Sigma_\gamma^{obs}}(\sigma) = P_2(\tau)\}$. Let $\sigma_1 \in \mathcal{S}_1$ and $\sigma_2 \in \mathcal{S}_2$, σ_1 and σ_2 are infinite and we have $P_{\Sigma_\gamma^{obs}}(\sigma_1) = P_{\Sigma_\gamma^{obs}}(\sigma_2)$. Moreover, the infinite set of ambiguous states in τ guarantees that $F \in \sigma_1 \cup \sigma_2 \wedge F \notin \sigma_1 \cap \sigma_2$, so σ_1 and σ_2 are undiagnosable. □

Theorem 1 means that, given a subsystem γ, the twin diagnoser $\Theta_\gamma(F)$ is a way to detect the infinite undiagnosable scenarios for F in the subsystem γ. Figures 4 and 5 illustrate the relationship between the undiagnosable scenarios and the twin diagnoser. The scenarios of Figure 5 are both represented by a transition path in the twin diagnoser of Figure 4. This path contains an infinite set of ambiguous states.

If two scenarios σ_1, σ_2 in γ are undiagnosable in γ, it does not mean yet, that F is not diagnosable inside the

subsystem γ. Those two scenarios are based on the local behaviours, they are *locally admissible*. The reason comes from the fact that behaviours in $\Theta_\gamma(F)$ are actually local and do not take into account the interactions with the neighbourhood (the components that communicate with this subsystem). Some behaviours described in the interactive diagnoser may not happen because the communication events cannot be synchronised with the neighbourhood. For this reason, behaviours involved in ambiguous cycles, may never globally happen.

Proposition 1 *Every scenario represented in $\Theta_\gamma(F)$ is* locally admissible.

The existence of σ_1, σ_2 in γ is based on the fact that the communication events between γ and its *neighbourhood* (i.e. the components that communicate with γ) in σ_1, σ_2 effectively occur. Those scenarios σ_1 and σ_2 exist only if each of them is *globally admissible*.

Definition 8 (Globally admissible scenario) *A scenario σ in a subsystem γ is* globally admissible *iff there exists σ_{glob} a scenario of the global model $\|\Gamma\|$ such that*

$$\sigma = P_{\Sigma_\gamma}(\sigma_{glob}).$$

Intuitively, if there exists a scenario σ_{glob} from $\|\Gamma\|$ whose projection on the subsystem γ is σ, it means that all the communication events from σ can be globally synchronised, the existence of σ is consistent with the system.

If we prove that there exists at least two infinite undiagnosable scenarios in γ that are globally admissible, then we prove that F is not diagnosable in γ.

Theorem 2 *F is diagnosable in the subsystem γ iff there is no couple of infinite undiagnosable scenarios inγ that are globally admissible.*

The result is a direct consequence of the definitions 2, 7 and 8.

4.4. Feedbacks from the diagnosability analysis

From this theoretical framework, we can deduce some results that the assistant can provide to the Designer.

1. If, inside the component-based model, we are able to find a subsystemγ in which every existing undiagnosable scenario is not globally admissible, then the assistant can provide the fact that F is diagnosable on that subsystem γ. From a design point of view, it means that for the given specification, F is diagnosable and there exists a subsystem γ whose monitoring is sufficient to diagnose any occurrence of F. From a monitoring point of view, it is very interesting because the smaller the subsystem is, the less costly the monitoring is.
2. If, for a given subsystem γ, we detect undiagnosable scenarios that are globally admissible, in order to make F diagnosable on that subsystem, the Designer has to respecify only part of this subsystem in order to eliminate the undiagnosable scenarios from the specification.

5. Algorithm

The design assistant algorithm performs the diagnosability analysis in a decentralised way (see Algorithm 1).

5.1. Decentralised algorithm

The algorithm takes as input the current component-based specification, a fault F and a set of components $\gamma_\mathcal{N}$. The subsystem $\gamma_\mathcal{N}$ is given as an input for two main reasons.

1. The Designer may specify a particular subsystem on which the analysis of the diagnosability of F has to be checked.
2. It is a way to limit the computation in case that the system is large. If the diagnosability analysis is not complete after the analysis of $\gamma_\mathcal{N}$, the algorithm provides an approximation of the result that is useful for the Designer.

The main idea of the algorithm is to firstly check the diagnosability of F locally (in Γ_1) (lines 4-6). For this purpose, the twin diagnoser of Γ_1 is computed and we extract from it only the states and transitions that are involved in infinite undiagnosable scenarios (line 5), so Uds_{Γ_1} becomes just a subpart of $\Theta_{\Gamma_1}(F)$. Inside Uds_{Γ_1}, there are maybe couples of infinite undiagnosable scenarios that have no communication events at all, in that case, we already know they are globally admissible since no further synchronisation operations can remove them: those scenarios are thus removed from Uds_{Γ_1} and put in $Gauds_{\Gamma_1}$ (line 6). In $Gauds_{\Gamma_1}$ is empty and Uds_{Γ_1} is not, then we cannot decide yet that F is not diagnosable in Γ_1. Uds_{Γ_1} then contains infinite undiagnosable scenarios whose global admissibility is not certain yet, a checking is required. To check the global admissibility of the scenarios Uds_{Γ_1}, the algorithm is based on the following result.

Theorem 3 *Let γ_1 and γ_2 be two disjoint subsystems, $\Theta_{\gamma_1\cup\gamma_2}(F) = \Theta_{\gamma_1}(F)\|_{com}\Theta_{\gamma_2}(F)$.*

Proof idea: This result is based on the fact that $\|$ is an associative and commutative operation.

Theorem 3 allows the computation of a twin diagnoser based on two other twin diagnosers. The consequence is

that, if we select a component Γ_i (line 9) and we compute the twin diagnoser of Γ_i, then $\Theta_{\Gamma_i}(F)||_{com}Uds_{\Gamma_1}$ is a part of $\Theta_{\{\Gamma_1,\Gamma_i\}}(F)$ that is generally smaller than $\Theta_{\{\Gamma_1,\Gamma_i\}}(F)$ so its computation is more efficient. All the undiagnosable scenarios of Uds_{Γ_1} are checked with Γ_i because of the synchronisation. Moreover, by construction, all the infinite undiagnosable scenarios that are locally admissible in $\Theta_{\{\Gamma_1,\Gamma_i\}}(F)$ are included in $\Theta_{\Gamma_i}(F)||_{com}Uds_{\Gamma_1}$.

Once $\Theta_{\Gamma_i}(F)||_{com}Uds_{\Gamma_1}$ is computed (line 11), the algorithm proceeds as before by extracting the scenarios (line 12) that can be automatically detected as globally admissible in $\Theta_{\Gamma_i}(F)||_{com}Uds_{\Gamma_1}$ (such scenarios do not contain communication events that are shared with components other that Γ_1 and Γ_i).

The algorithm stops for several reasons:

1. $Gauds_\gamma \neq \emptyset$: in that case, the algorithm founds out that F is not diagnosable in $\gamma \subseteq \gamma_\mathcal{N}$ because of the presence of globally admissible scenarios described in $Gauds_\gamma$. In that situation, we know that F is not diagnosable in any subsystem that contains γ.
2. $Gauds_\gamma = \emptyset \wedge Uds_\gamma = \emptyset$: in that case, the algorithm founds out that F is diagnosable in γ. Every occurrence of F can be diagnosed after a finite sequence of events from γ.
3. $\gamma = \gamma_\mathcal{N} \wedge Uds_\gamma \neq \emptyset \wedge Gauds_\gamma = \emptyset$: in that case, the algorithm is incompletely conclusive. F may be diagnosable because no global admissible scenario has been found yet. Other components, not included in $\gamma_\mathcal{N}$, have to be taken into account.

5.2. Complexity discussion

Computing the twin diagnoser $\Theta_\gamma(F)$ of a subsystem γ is exponential to the number of components in γ in the worst case.In [8], the diagnosability analysis consists in computing $\Theta_\Gamma(F)$ so it is $O(2^N)$ where N is the number of components in the system.In Algorithm 1, the complexity is the worst case is $O(2^M)$ where M is the number of components in $\gamma_\mathcal{N}$. So computing the undiagnosable scenarios is still, in the worst case, an exponential task in the number of components. Nevertheless, our approach is more efficient in practice. In the case where F is diagnosable, only a few components in the neighbourhood make F diagnosable. Then, with the neighbourhood selection (line 9), it is likely that the algorithm will find such a neighbourhood if $\gamma_\mathcal{N}$ is big enough. Moreover, the algorithm uses a decentralised approach, therefore, in practice, it computes only parts of $\Theta_\gamma(F)$ (the undiagnosable parts). Another advantage of our algorithm is its incrementality. Even if we cannot compute a subsystem for which F is diagnosable, we are still able to provide for smaller subsystems a set of undiagnosable scenarios that are *locally admissible*.

Algorithm 1 Undiagnosable scenarios of fault F

1: **Input:** Component-based specification $\{\Gamma_1, \ldots, \Gamma_N\}$
2: **Input:** F a fault event occurring in the component Γ_1
3: **Input:** Neighbourhood $\gamma_\mathcal{N}$
4: Compute the twin diagnoser $\Theta_{\Gamma_1}(F)$
5: $Uds_{\Gamma_1} \leftarrow$ UndiagScenarios($\Theta_{\Gamma_1}(F)$)
6: $Gauds_{\Gamma_1} \leftarrow$ RemoveGlobAdmissibleScenarios(Uds_{Γ_1})
7: $\gamma \leftarrow \{\Gamma_1\}$
8: **while** $Gauds_\gamma = \emptyset \wedge Uds_\gamma \neq \emptyset \wedge \gamma \neq \gamma_\mathcal{N}$ **do**
9: $\Gamma_i \leftarrow$ SelectComponent($\gamma, \gamma_\mathcal{N}$)
10: Compute the twin diagnoser $\Theta_{\Gamma_i}(F)$
11: $Uds_{\gamma\cup\{\Gamma_i\}} \leftarrow$ UndiagScenarios($\Theta_{\Gamma_i}(F)\|_{com}Uds_\gamma$)
12: $Gauds_{\gamma\cup\{\Gamma_i\}} \leftarrow$ RemoveGlobAdmissibleScenarios($Uds_{\gamma\cup\{\Gamma_i\}}$)
13: $\gamma \leftarrow \gamma \cup \{\Gamma_i\}$
14: **end while**
15: **if** $Uds_\gamma = \emptyset$ **then**
16: **if** $Gauds_\gamma \neq \emptyset$ **then**
17: Print "F is not diagnosable in any subsystem including γ. See $Gauds_\gamma$."
18: **else**
19: Print "F is diagnosable in γ."
20: **end if**
21: **else**
22: **if** $Gauds_\gamma \neq \emptyset$ **then**
23: Print "F is not diagnosable in any subsystem including $\gamma_\mathcal{N}$. See $Gauds_\gamma$ and Uds_γ"
24: **else**
25: Print "F may be diagnosable in $\gamma_\mathcal{N}$. See Uds_γ. Other computation required"
26: **end if**
27: **end if**

6. Results

6.1. Running example

In the running example (see Figure 2), two faults can occur: f_1 and f_2. f_2 is not diagnosable in the subsystem $\{\Gamma_2\}$ because every infinite scenario inΓ_2 has the same observable behaviour. However, f_2 is diagnosable in $\{\Gamma_2, \Gamma_3\}$ because Γ_3 discriminates the occurrence of the events c_1 and c_2 of Γ_2. The observation of the sequences $o3, o5$ or $o5, o3$ in the subsystem $\gamma = \{\Gamma_2, \Gamma_3\}$ is sufficient to diagnose the occurrence of $f2$ with certainty, whatever the observations of Γ_1 are. As far as the fault f_1 is concerned, there is no subsystem in which it is diagnosable. The scenarios of Figure 5 are undiagnosable and globally admissible in Γ_1, they can be extracted from $Gauds_\Gamma$ generated by Algorithm 1. The main problem of the non-diagnosability of f_1 is that the future of f_1 is exactly the same as the future of c_1 (see Figure 5). In order to make f_1 diagnosable, it is sufficient to

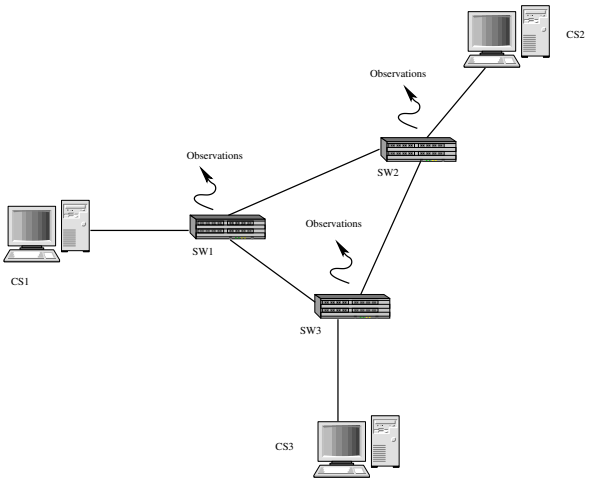

Figure 6. Example of component-based system: telecommunication network

change this fact in the new specification so that this couple of undiagnosable scenarios do not happen any more.

6.2. Experimental evaluation

We have implemented a prototype to evaluate the improvement of our decentralised algorithm compared to the centralised method of [8]. We have run it on an example extracted from a telecommunication network model (see [12] for details). This model is an extract from a real-world system. It defines communication protocols between station controls CS and switches SW in order to manage flows of data between switches. Only the switches are observable. A topology of this network is presented in Figure 6.

This example contains 4 types of faults (each component of the same type has the same types of fault) and is modelled with 9 components (2 components are required to model a switch). The global model $\|\Gamma\|$ of this system contains 124128 states and 532748 transitions. Even if the system is simple, computing the diagnosability of a fault with a centralised approach has not been possible with our computational resources[4] due to the lack of memory and time. The interactive diagnoser for one fault contains on average 300000 states and 1000000 transitions, so the state space of the corresponding twin diagnoser is 9×10^{10} in the worst case. With our approach, we were able to find out that three of the four fault types are diagnosable. This result has only been obtained by computing only 1701 states and 4212 transitions. The small number of states and transitions comes from the fact that the subsystem in which each fault is diagnosable, is at the most composed of 3 components (a CSi components and the 2 components of a switch SWi). Another consequence of this analysis is the fact that, to diagnose this type of fault, the observations from one switch are sufficient so the monitoring of the observations from the other switches are not required. As far as the undiagnosable fault is concerned, we were able to compute the undiagnosable scenarios for a $\gamma_{\mathcal{N}}$ containing 6 components in a few seconds which is an excellent response time for a design assistant. This result has been obtained by only computing only 22734 states and 115278 transitions. From the resulting undiagnosable scenarios that are all just locally admissible, we have noticed that, in that particular case, only two components of the neighbourhood bring diagnosability information. The other ones do not remove undiagnosable scenarios at all in the component where the fault occur. As a consequence, it is very likely that the scenarios we have computed are actually globally admissible.

[4] PIII 1Ghz with 512Mb RAM.

7. Related work

Diagnosability of discrete event systems has been studied for ten years and there is a significant amount of work dealing with this issue, both in the AI community [3], [1], [10] and in the Control community [13], [8], [5], [14]. In most of the previous works, the diagnosability analysis share the same weaknesses. First, they just detect if a system is diagnosable or not, no other informative feedback, like undiagnosable scenarios, is provided. The second weakness of the previous works comes from the fact that the diagnosability analysis is performed globally without taking into account the distributed nature of component-based system. The first diagnosability method is based on the classical diagnoser approach [13]. This diagnoser is a finite-state machine based on the global model of the system which is able to efficiently perform on-line diagnosis. The diagnosability checking problem is then equivalent to the detection of ambiguous paths in this diagnoser. The main problem of this approach is that, given the global model $\|\Gamma\|$, the computation of the diagnoser has an exponential complexity. [8] and [14] propose to improve the classical approach by the use of a global twin-plant method (also called global verifier) whose computation is only polynomial but still based on the global model. [1] proposes an identical technique to check a more restrictive diagnosability property on dynamical systems that is implemented with a symbolic model checker. In [3], the diagnosability problem is defined as a consistency problem between relevant observations and the global system description using process algebra.

8. Conclusion and Perspectives

Diagnosability analysis cannot be reduced to find a *yes-or-no* answer. The systems become larger and larger and

we need to get more information from the diagnosability analysis of those systems: this is the motivation of this approach. To our knowledge, this method is the first attempt to solve this crucial problem by providing granular information about the reasons why a system specification is not diagnosable. Since the algorithm is decentralised, the diagnosability analysis takes into account the fault location first and can detect undiagnosable scenarios faster than any centralised approaches. The algorithm is incremental and can be used interactively during the design process to build a diagnosable system. In order to reduce the cost of fault diagnosis, the Designer has to specify a system so that the fault can be diagnosed in a small subsystem. This specification depends on sensor placements (what is observable? what is the sensor cost?) and on communication protocols between components.

The main perspective of this approach is to extend this assistant to provide more accurate requirements for the system monitoring in order to automatically generate an optimal fault diagnosis algorithm for a particular system. Firstly, if the fault is diagnosable in a subsystem, we can use this information to implement a specialised and light diagnosis algorithm for that particular fault based on that particular subsystem. Secondly, in case of a specification where faults are not diagnosable, it is still possible to implement a diagnosis algorithm that takes into account this fact. If a fault is not diagnosable, then there exist some situations when the fault will never be diagnosed with certainty and the diagnosis process will detect it and will have the possibility to stop definitely. The second perspective is to use this assistant to provide an active monitoring system. Active monitoring means that, given a fault diagnosis at a given time that is not certain yet, the monitoring system will be able to automatically activate tests on the system (if the system allows such actions) in order to remove any ambiguity. Activating tests is a way to control the system so that undiagnosable scenarios do not occur.

A. Composition/Projection operators

Composition: Let $\{\mathcal{A}_i\}_{i\in\{1,\dots,m\}}$ be m automata on the m set of events $\{\mathcal{E}_i\}$ and $\mathcal{E}$ a set of events (s.t. $\epsilon \notin \mathcal{E}$) from the automata. The composition $\|_{\mathcal{E}}$ is such that the automaton $\mathcal{A} = \mathcal{A}_1 \|_{\mathcal{E}} \dots \|_{\mathcal{E}} \mathcal{A}_m$ is the subpart of the cartesian product containing the synchronised transitions. A transition $t = (x_1 \xrightarrow{e_1} x'_1, \dots, x_m \xrightarrow{e_m} x'_m)$ is synchronised according to $\mathcal{E}$ iff

$$(\exists j \in \{1,\dots,m\} : e_j \notin \mathcal{E} \wedge \forall i \in \{1,\dots,m\} \setminus \{j\} : e_i = \epsilon)$$
$$\vee((\exists j \in \{1,\dots,m\} : e_j \in \mathcal{E}) \wedge (\forall i \in \{1,\dots,m\} :$$
$$(e_j \in \mathcal{E}_i \Rightarrow e_i = e_j) \wedge (e_j \notin \mathcal{E}_i \Rightarrow e_i = \epsilon))).$$

$\mathcal{A}$ represents the behaviour of $\{\mathcal{A}_i\}_{i\in\{1,\dots,m\}}$ where only the events of $\mathcal{E}$ are synchronised. By definition, $\|_{\mathcal{E}}$ is an associative and commutative operator. We denote by $\|_{com}$ (resp. $\|_{obs}$) the composition operator synchronised on the communication events (resp. observable events) involved in the automata to compose.

Projection: Let $\mathcal{A}$ be an automaton based on the set of events Σ and Σ' be another set of events. $\mathcal{A}$ represents the prefix-closed language L. The projection $P_{\Sigma'}(L)$ of L on Σ' is defined as follows. $\forall w \in L : P_{\Sigma'}(w) = \epsilon$ if $w \in \Sigma \setminus \Sigma', w$ if $w \in \Sigma' P_{\Sigma'}(u) P_{\Sigma'}(v)$ if $w = uv \wedge u \in \Sigma$. We denote by $P_{\Sigma'}(\mathcal{A})$ an automaton which represents $P_{\Sigma'}(L)$.

References

[1] A. Cimatti, C. Pecheur, and R. Cavada. Formal verification of diagnosability via symbolic model checking. In *Proc. IJCAI'03*, Acapulco, Mexico, 2003.

[2] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[3] L. Console, C. Picardi, and M. Ribaudo. Diagnosis and diagnosibility analysis using pepa. In *Proc. ECAI'2000*, pages 131–135, Germany, 2000.

[4] L. Console, C. Picardi, and M. Ribaudo. Process algebra for systems diagnosis. *Artificial Intelligence*, 142:19–51, November 2002.

[5] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *JDEDS: Theory and Application*, 10(1–2):33–86, 2002.

[6] E. Fabre, A. Benveniste, S. Haar, and C. Jard. Distributed monitoring of concurrent and asynchronous systems. *Journal of Discrete Event Dynamic Systems*, 15:33–83, March 2005.

[7] C. Jard and T. Jéron. Tgv: theory, principles and algorithms, a tool for the automatic synthesis of conformance test cases for non-deterministic reactive systems. *Software Tools for Technology Transfer (STTT)*, 6, October 2004.

[8] 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.

[9] G. Lamperti and M. Zanella. *Diagnosis of active systems*. Kluwer Academic Publishers, 2003.

[10] Y. Pencolé. Diagnosability analysis of distributed discrete event systems. In *Proc. ECAI'04*, pages 43–47, 2004.

[11] Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164:121–170, May 2005.

[12] L. Rozé and M.-O. Cordier. Diagnosing discrete-event systems: extending the "diagnoser approach" to deal with telecommunication networks. *JDEDS*, 12(1):43–81, 2002.

[13] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete event system. *IEEE Trans. on Automatic Control*, 40(9):1555–1575, 1995.

[14] T. Yoo and S. Lafortune. Polynomial-time verification of diagnosability of partially-observed discrete-event systems. *IEEE Trans. of Automatic Control*, 47(9):1491–1495, 2002.