Project no.: **PIRSES-GA-2011-295261**

Project full title: **Mobility between Europe and Argentina applying Logics to Systems**

Project Acronym: **MEALS**

Deliverable no.: 4.2 / 1

Title of Deliverable: Distributed Analysis for Diagnosability in Concurrent Systems

Contractual Date of Delivery to the CEC:	**30-Sep-2013**
Actual Date of Delivery to the CEC:	**30-Sep-2013**
Organisation name of lead contractor for this deliverable:	**ULEIC**
Author(s):	**Hernán Ponce de León,** **Gonzalo Bonigo,** **Laura Brandán Briones**
Participants(s):	**IMP, UBA, ULEIC, TUE, UNC**
Work package contributing to the deliverable:	**WP4**
Nature:	**R**
Dissemination Level:	**Public**
Total number of pages:	**15**
Start date of project:	1 Oct. 2011 **Duration:** 48 month

Abstract:

Complex systems often exhibit unexpected faults that are difficult to handle. Such systems are desirable to be diagnosable, i.e. faults can be automatically detected as they occur (or shortly afterwards), enabling the system to handle the fault or recover. A system is diagnosable if it is possible to detect every fault, in a finite time after they occurred, by only observing the available information from the system. Complex systems are usually built from simpler components running concurrently. We study how to infer the diagnosability property of a complex system (distributed and with multiple faults) from a parallelized analysis of the diagnosability of each of its components synchronizing with fault free versions of the others. In this paper we make the following contributions: (1) we address the diagnosability problem of concurrent systems with arbitrary faults occurring freely in each component. (2) We distribute the diagnosability analysis and illustrate our approach with examples. Moreover, (3) we present a prototype tool that implements our techniques showing promising results.

Note:

This deliverable is based on material that has been published in 24th International Workshop on Principles of Diagnosis.

This project has received funding from the European Union Seventh Framework Programme (FP7 2007-2013) under Grant Agreement Nr. 295261.

Contents

1 Introduction

As systems become larger, their behavior becomes more complex. Several things may go wrong, resulting in faults occurring. It is then crucially important to design our systems in a way that we can detect or recover from such faults when they occur. A system is diagnosable when its design allows the detection of faults, for instance a system that has sensors specially dedicated to detect them. Sometimes the detection of faults is more involved and the diagnosability property is harder to establish, specially in systems with several components.

A sound software engineering rule for building complex systems is to divide the whole system in smaller and simpler components, each solving a specific task. Moreover, usually they are built by different groups of people and may be in different places. This means that, in general, complex systems are actually collections of simpler components running in parallel.

In order to model such systems and formally prove results, there are several formalisms like Finite State Machines (FSMs) [11, 8], Petri Nets [7, 9] and Labeled Transition Systems (LTSs) [4, 5, 3]. In this paper, we model each component by a LTS, so the whole system is a collection of LTSs synchronizing in all their shared observable actions (see Section 2).

In the diagnosability analysis of distributed systems it is usually assumed that a fault can occur in exactly one of the different components. We relax this assumption allowing the same fault to occur in several components.

Also, the diagnosability analysis is usually iterative (i.e., sequential): the information from local diagnosers is combined until a global verdict is reached. We propose a method to distribute this analysis.

Finally, we developed a tool that implements all our research. The DADDY tool (Distributed Analysis for Distributed Discrete sYstems) [2] is a prototype based on the results presented in [3] and this paper. The tool does not only implement the method we presents but also the classic one allowing us to compare both approaches. We present a comparative analysis of their performance obtained from the experimental running of several examples.

Related Work Diagnosability was initially developed in [11] under the setting of discrete event systems. In that paper, necessary and sufficient conditions for testing diagnosability are given. In order to test diagnosability, a special diagnoser is computed, whose complexity of construction is shown to be exponential in the number of states of the original system, and double exponential in the number of faults. Later, in [8], an improvement of this algorithm is presented, where the so-called twin plant method is introduced and shown to have polynomial complexity in the number of states and faults. Afterwards, in [13], an improvement to the twin plant method is presented where the system is reduced before building the twin plant.

None of the methods presented there (i.e., [11, 8]) consider the problem when the system is composed of components working in parallel. An approach to this consideration is addressed in [13, 6, 10, 12] where the diagnosability problem is performed by either local diagnosers or twin plants communicating with each other, directly or through a coordinator, and by that means pooling together the observations. [14] shows that when considering only local observations, diagnosability becomes undecidable when the communication between component is unobservable. An algorithm is proposed to check a sufficient but not necessary condition of diagnosability.

However, their results are based in the assumption that a fault can only occur in one of the components, an assumption that can not always be made.

Several mechanisms such as interleaving, shared variables and handshaking have been described in [1] to provide operational models for distributed systems. In the handshaking method, the communication is made by the synchronization on actions or events. These actions must be specified a priori in the model, so the different components can be synchronized at execution time. In [3] the authors study how different kinds of synchronizations (via all the shared actions, some of them or none) impact in the diagnosis analysis.

Motivation Suppose different groups of people are commanded to build different components of a system. Even if each component is diagnosable, it is not always the case that the resulting system has such property[1]. In [3] the authors show that with different kinds of synchronizations, the diagnosability of the global system can not be inferred directly from the diagnosability of each component.

We propose a framework where each component only shares with the rest a fault free version of its own, maybe the specification of its ideal behavior. Then, each component should not only be diagnosable, but also its interaction with the fault free version of the others, i.e. its synchronous product with fault free version of the other components. Therefore, our diagnosability analysis can be distributed.

Paper organization Section 2 presents the formal model that we use for modeling each component, the parallel composition between them and the notion of diagnosability. In Section 3, we develop our analysis method, showing how the diagnosability of each component synchronizing with fault free versions of the other components influences the diagnosability property of the overall system. Section 4 presents our tool DADDY and some experimental results. We conclude and discuss about future work in Section 5.

2 Diagnosability Analysis

2.1 Model of the system

We consider a distributed system composed of two autonomous components G_1, G_2 that communicate with each other by all their shared observable actions. The local model of a component is defined as a Labeled Transition System.

Definition 1. A Labeled Transition System (LTS) is a tuple $G = (Q, \Sigma, \delta, q_0)$ where

- Q is a finite set of states,
- Σ is a finite set of actions,
- δ is a partial transition function, and

[1] See for example C, D and $C \times D$ in Figures 1 and 2.

- q_0 the initial state, with $q_0 \in Q$.

As usual in diagnosability analysis, some of the actions of Σ are observable while the rest are unobservable. Thus the set of actions Σ is partitioned as $\Sigma = \Sigma_o \uplus \Sigma_{uo}$ where Σ_o represents the observable actions and Σ_{uo} the unobservable ones.

The faults to diagnose are considered unobservable, i.e. $\Sigma_F \subseteq \Sigma_{uo}$, as faults that are observable can be easily diagnosable.

As usual in diagnosability analysis, we made the following assumptions about our systems.

Assumption 1. We only consider (live) systems where there is a transition defined at each state, i.e. the system cannot reach a point at which no action is possible.

Assumption 2. The system does not contain cycles of unobservable actions.

Note that, these assumptions together assure that all our systems are free of observation starvation.

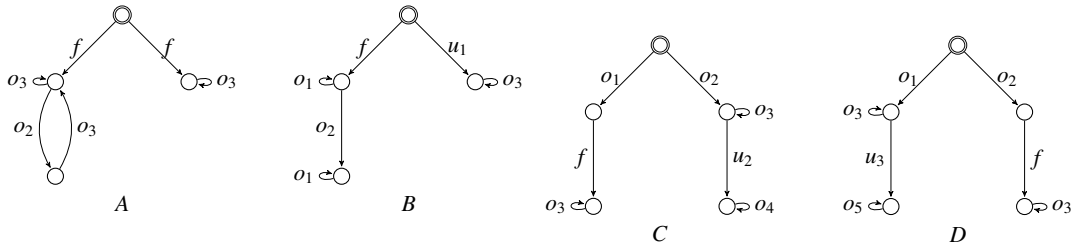

Figure 1: Specification of four components modeled by LTSs

Figure 1 shows four components modeled by the LTSs A, B, C and D where $o_1, o_2, o_3, o_4, o_5 \in \Sigma_o$ and $u_1, u_2, u_3 \in \Sigma_{uo}$. The special action $f \in \Sigma_F$ is the fault to be diagnosable.

A path from state q_i to state q_j in G is a sequence $q_i{\cdot}a_i{\cdot}q_{i+1} \ldots a_{j-1}{\cdot}q_j$ such that $(q_k, a_k, q_{k+1}) \in \delta$ for $i \leq k \leq j-1$. The set of paths in G is denoted by paths(G).

The trace associated with any given path consists of its sequence of actions (i.e., for a path $\rho = q_0 \cdot a_0 \cdot q_1 \ldots a_{n-1} \cdot q_n$ we have trace(ρ) $= a_0 \cdot a_1 \ldots a_n$). Given a trace, $\sigma = a_0 \cdot a_1 \ldots a_n$, we denote as $f \in \sigma$ when there exists i such that $f = a_i$. As our systems are live, we only consider infinity traces where the infinite repetition of an actions a is denoted by $\widehat{a}$. The set of all traces starting in q_0 is denoted by traces(G). As we consider nondeterministic systems, the same trace can belong to several paths. The set of possible paths of a trace σ in G are: path(σ) = $\{\rho \in \text{paths}(G) \mid \text{trace}(\rho) = \sigma\}$.

The observation of a trace is given by the following definition.

Definition 2. Let $\sigma \in \Sigma^*$, then

$$obs(\sigma) \quad = \quad \begin{cases} \epsilon & \text{if } \sigma = \epsilon \\ a{\cdot}obs(\sigma') & \text{if } \sigma = a{\cdot}\sigma' \wedge a \in \Sigma_o \\ obs(\sigma') & \text{if } \sigma = a{\cdot}\sigma' \wedge a \notin \Sigma_o \end{cases}$$

The communication between two components is given by their synchronous product where the synchronizing actions are all the shared observable ones.

Definition 3. Given two local components $G_1 = (Q^1, \Sigma^1, \delta^1, q_0^1)$ and $G_2 = (Q^2, \Sigma^2, \delta^2, q_0^2)$, the behavior of the global system is given by their synchronous product $G_1 \times G_2 = (Q^1 \times Q^2, \Sigma^1 \cup \Sigma^2, \delta^{1\times2}, (q_0^1, q_0^2))$ where $\delta^{1\times2}$ is defined as follows

$$\delta^{1\times2}((q_i^1, q_j^2), a) = \begin{cases} (\delta^1(q_i^1, a), \delta^2(q_j^2, a)) & \text{if } a \in \Sigma_o^1 \cap \Sigma_o^2 \\ (\delta^1(q_i^1, a), q_j^2) & \text{if } a \in \Sigma^1 \wedge a \notin \Sigma^2 \\ (q_i^1, \delta^2(q_j^2, a)) & \text{if } a \in \Sigma^2 \wedge a \notin \Sigma^1 \end{cases}$$

Given a path in the global system, we can project it to a single component.

Definition 4. Let $\rho \in \text{paths}(G_1 \times G_2)$, its projection in G_i is

$$\begin{aligned} P_i((q^1, q^2)) &= q^i \\ P_i((q^1, q^2) \cdot a \cdot \rho') &= \begin{cases} q^i \cdot a \cdot P_i(\rho') & \text{if } \exists\, \delta^i(q^i, a) \\ P_i(\rho') & \text{otherwise} \end{cases} \end{aligned}$$

For a trace in the global system, we define the projections to know which actions belong to a certain component.

Definition 5. Let σ be a trace in traces$(G_1 \times G_2)$, σ' is its projection in G_i, denoted $P_i(\sigma) = \sigma'$, iff

$$\exists\, \rho \in \text{path}(\sigma) : \text{trace}(P_i(\rho)) = \sigma'$$

Example 1. Let $\sigma = o_1 f o_3 u_3 \widehat{o_5}$ be a trace in traces$(C \times D)$ from Figure 2, its projection in component C is given by $P_C(\sigma) = o_1 f o_3$ and its projection in component D is given by $P_D(\sigma) = o_1 o_3 u_3 \widehat{o_5}$. These projections are traces of the corresponding components C and D from Figure 1. Note that projections of an infinite trace from the global system can be finite in one of the components.

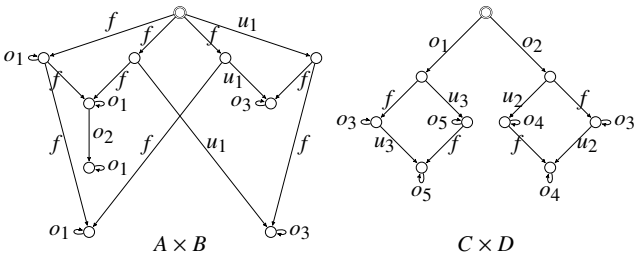

Figure 2: Synchronous product of components A, B and C, D

As the projection operator only erases actions in a trace, it is easy to see that every fault belonging to a projection of such a trace, also belongs to the trace in the global system as it is shown by the following result.

Proposition 1. *For every trace σ in traces$(G_1 \times G_2)$ with $P_i(\sigma) = \sigma_i$, we have*

$$\textit{if} \quad f \in \sigma_i \quad \textit{then} \quad f \in \sigma$$

When two components synchronize in all their shared actions, if two traces of the global system have the same observability and we project them to the same component, the resulting projections will also have the same observability. This result is captured by Proposition 2.

Proposition 2. *Given two traces σ and α in traces$(G_1 \times G_2)$ with $P_i(\sigma) = \sigma_i$ and $P_i(\alpha) = \alpha_i$, we have*

$$\textit{if} \ \ obs(\sigma) = obs(\alpha) \quad \textit{then} \quad obs(\sigma_i) = obs(\alpha_i)$$

This result is proved by double induction in the structure of σ and α. We analyze several cases depending on the existence of the projections. One of the most critical cases is when $\sigma = a{\cdot}\sigma', \alpha = b{\cdot}\alpha', a \in \Sigma_o^1 \cap \Sigma_o^2$, but $b \notin \Sigma_o^1 \cap \Sigma_o^2$ as it has several particular sub-cases. Note that this result only holds when the synchronization is done in all the set of shared actions.

2.2 Diagnosability condition

We present now the notion of diagnosability. Informally, a fault $f \in \Sigma_F$ is diagnosable if it is possible to detect, within a finite delay, occurrences of such a fault using the record of observed actions. In other words, a fault is not diagnosable if there exist two infinite paths from the initial state with the same infinite sequence of observable actions but only one of them contains a fault.

Definition 6. Let f be a fault in Σ_F, f is diagnosable in G iff

$$\forall \sigma, \alpha \in \text{traces}(G) : \text{if } obs(\sigma) = obs(\alpha)$$

$$\text{and } f \in \sigma \text{ then } f \in \alpha$$

The system G is diagnosable, denoted by **diag**(G), if and only if every fault $f \in \Sigma_F$ is diagnosable.

The previous definition introduced in [4] is a reformulation of the one presented in [11].

Example 2. Let consider the components A and B from Figure 1. The only pair of traces in A with the same observability are of the form $f\widehat{o_3}$ (one for each branch from the initial state), as both traces contain the fault f, system A is diagnosable. In the case of B, each observable trace corresponds to a unique path, therefore B is diagnosable.

Now, consider system $A \times B$ from Figure 2, we can see that every trace contains a fault, therefore $A \times B$ is diagnosable. On the contrary, in system $C \times D$ we have two traces, $o_2u_2\widehat{o_4}$ and $o_2fu_2\widehat{o_4}$ that have the same observability, but one of them contains a fault and the other does not, therefore the system $C \times D$ is not diagnosable.

3 Distributing the diagnosability analysis

The notion of diagnosability is introduced in [11] assuming a centralized architecture of the system. In order to check the diagnosability property in distributed systems, the synchronous product of components is computed and such a product is given as an input to an algorithm that tests its diagnosability (usually based on the twin plant method). The size of such a product grows exponentially with respect of the size of the components, resulting in an inefficient algorithm. When dealing with real applications, such as telecommunication networks or power distribution networks, the centralized approach is clearly unrealistic because of the size of those applications. Moreover, this approach does not exploit the fact that such systems are distributed.

In [13, 10] the authors distribute the search for non-distinguishable behaviors based on local verifiers and local twin plants. The local information is propagated until a verdict is made or, in the worst case, the global system is built. Their result is based on the assumption that a fault can occur in exactly one component.

In this section we present a method that allows to decide the diagnosability of a distributed system in terms of the diagnosability of each faulty component synchronizing with fault free versions of the remaining ones. Basically, we compose each component with a fault free version of the other components and analyze their diagnosability in parallel. To the best of our knowledge, it is the first method that allows to parallelize the diagnosability analysis.

Algorithm 1

Require: A LTS $G = (Q, \Sigma, \delta, q_0)$
Ensure: An f-fault free version of G
1: $Q' := \{q_0\}$, $\delta' := \emptyset$, $Q := Q \setminus \{q_0\}$
2: **while** $\exists (q',x,q) : q' \in Q' \wedge (q',x,q) \in \delta \wedge (q',x,q) \notin \delta'$ **do**
3: **if** $x \neq f$ **then**
4: $Q' := Q' \cup \{q\}$
5: $\delta' := \delta' \cup (q', x, q)$
6: **end if**
7: $\delta := \delta \setminus (q', x, q)$
8: **end while**
9: **return** $G^f = (Q', \Sigma, \delta', q_0)$

For testing the diagnosability of a fault $f \in \Sigma_F$ in the global system, instead of computing the whole composition, we consider one component and compose it with the fault free versions of the others. These fault free versions may be taken as the specification of each component, when provided, or can be computed by removing the fault f in the component using Algorithm 1 and considering such as the correct behavior of the system.

If we compose a component G_i with the fault free version of G_j, meaning G_j^f, clearly the traces of the resulting system are those of $G_i \times G_j$ such that its projections in G_j are fault free.

Proposition 3. *Let G_i and G_j be two LTSs, then $\sigma \in traces(G_i \times G_j^f)$ iff*

$$\sigma \in traces(G_i \times G_j) \wedge \forall \sigma_j : P_j(\sigma) = \sigma_j \Rightarrow f \notin \sigma_j$$

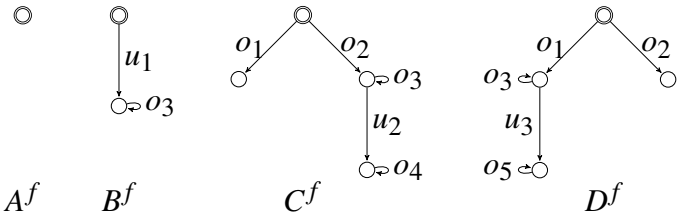

Figure 3: Components A, B, C and D after removing their faults

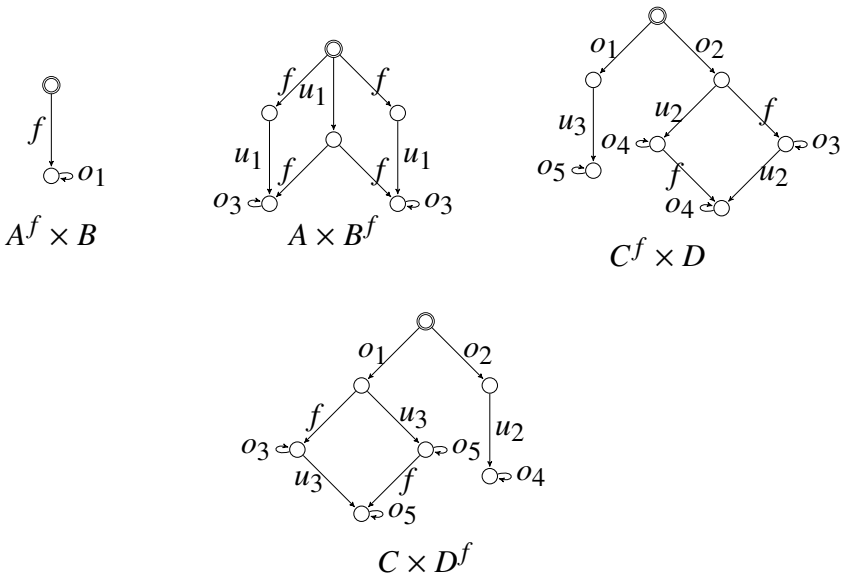

Figure 4: Composed systems after removing the faults in one of the components

Figure 3 shows components A, B, C and D after removing fault f and Figure 4 shows them synchronizing with the faulty components.

Example 3. Let us consider the systems from Figure 4. System $A^f \times B$ is trivially diagnosable. In the case of system $A \times B^f$, it is easy to see that the observable traces are of the form $\widehat{o_3}$, but all traces containing o_3 also contain f and therefore $A \times B^f$ is also diagnosable. In system $C^f \times D$, traces $\sigma = o_2 u_2 \widehat{o_4}$ and $\alpha = o_2 f u_2 \widehat{o_4}$ have the same observability, but α contains a fault and σ does not. So, we can conclude that $C^f \times D$ is not diagnosable.

The following result states necessary conditions for the diagnosability of the global system, i.e. the non diagnosability of $G_1^f \times G_2$ or $G_1 \times G_2^f$ implies the non diagnosability of $G_1 \times G_2$.

Theorem 1. *Let G_1 and G_2 be two LTSs, then*

$$\textbf{\textit{diag}}(G_1 \times G_2) \;\Rightarrow\; \textbf{\textit{diag}}(G_1^f \times G_2) \wedge \textbf{\textit{diag}}(G_1 \times G_2^f)$$

Proof. Lets assume that $\neg\textbf{diag}(G_1^f \times G_2)$, then there exist two traces $\sigma, \alpha \in \text{traces}(G_1^f \times G_2)$ and f such that $obs(\sigma) = obs(\alpha)$ with $f \in \sigma$, but $f \notin \alpha$. We know from Proposition 3 that every trace in $G_1^f \times G_2$ is a trace in $G_1 \times G_2$, so we have found two traces of the global system with the same observability, one containing a fault and the other one not. Therefore $(G_1 \times G_2)$ is non-diagnosable. An analogous analysis can be made if $\neg\textbf{diag}(G_1 \times G_2^f)$. □

Example 4. We see in Example 3 that $C^f \times D$ is non diagnosable. Using Theorem 1 we can conclude that $C \times D$ is non diagnosable. This result is consistent with the diagnosability analysis made in Example 2.

As explained above, the idea is to build a diagnosable component and to test that its interaction with another fault free component is also diagnosable. We can then decide the diagnosability of $G_1 \times G_2$ in term of the diagnosability of $G_1, G_2, G_1^f \times G_2$ and $G_1 \times G_2^f$.

Theorem 2. *Let G_1 and G_2 be two LTSs, then*

$$\left.\begin{array}{l} \textbf{\textit{diag}}(G_1) \wedge \textbf{\textit{diag}}(G_1 \times G_2^f) \\ \textbf{\textit{diag}}(G_2) \wedge \textbf{\textit{diag}}(G_1^f \times G_2) \end{array}\right\} \Rightarrow \textbf{\textit{diag}}(G_1 \times G_2)$$

Proof. Let assume that we have a fault $f \in \Sigma_F$ and two traces $\sigma, \alpha \in \text{traces}(G_1 \times G_2)$ with $f \in \sigma$ and $obs(\sigma) = obs(\alpha)$, we need to prove that $f \in \alpha$. Consider the following cases:

1. if $\sigma, \alpha \in \text{traces}(G_i^f \times G_j)$ we can prove by $(G_i^f \times G_j)$'s diagnosability that $f \in \alpha$ and then $G_1 \times G_2$ is diagnosable,

2. if $\alpha \notin \text{traces}(G_i^f \times G_j)$, using the hypothesis that $\alpha \in \text{traces}(G_i \times G_j)$, we can apply Proposition 3 and obtain that $\exists \alpha_i : P_i(\alpha) = \alpha_i \wedge f \in \alpha_i$. By Proposition 1 we know that every fault belonging to a projection also belongs to the trace in the global system, then $f \in \alpha$ and $G_1 \times G_2$ is diagnosable,

3. if $\alpha \in \text{traces}(G_i^f \times G_j)$ and $\sigma \notin \text{traces}(G_i^f \times G_j)$ we know by Proposition 3 that $\forall \alpha_i : P_i(\alpha) = \alpha_i$ and $f \notin \alpha_i$ and also that $\exists \sigma_i : P_i(\sigma) = \sigma_i$ with $f \in \sigma_i$. As $obs(\sigma) = obs(\alpha)$ we have that $obs(\sigma_i) = obs(\alpha_i)$ by Proposition 2. Finally as G_i is diagnosable and $f \in \sigma_i$, the fault should belong to α_i, leading to a contradiction. We can conclude that $G_1 \times G_2$ is diagnosable.

□

Example 5. From Example 2 and Example 3 we know that $A, B, A^f \times B$ and $A \times B^f$ are diagnosable. If we apply Theorem 2 we can conclude that $A \times B$ is diagnosable, which is consistent with the analysis made in Example 2.

3.1 Generalization

Until now we only consider systems composed by only two components. However, real examples are usually more complex and are composed of several components. Therefore we need to generalize the previous results to global systems composed of n different components running in parallel.

In order to generalize all our results, the associativity and commutativity property of synchronous product become essential. Note that in a general case the set of synchronizing actions is not necessarily the intersection of all their observable actions. Suppose that a system is composed by three components, G_1, G_2 and G_3, where two of them synchronize via an action a that does not belong to a third component, i.e. $a \in \Sigma_o^1 \cap \Sigma_o^2$, but $a \notin \Sigma_o^3$. We expect that G_1 and G_2 still synchronize in a. Fortunately, despite its apparent complications, the synchronous product is associative and commutative. The proof of such result can be found in previous work [3].

The following results generalized Theorems 1 and 2 respectively, giving necessary and sufficient conditions for the diagnosability of the global system.

Theorem 3. *Let $G_1, G_2, \ldots, G_n$ be n components modeled by LTSs, then*

$$\begin{aligned} &\boldsymbol{diag}(G_1 \times G_2 \times \ldots \times G_n) \\ &\Downarrow \\ &\overbrace{\boldsymbol{diag}(G_1 \times G_2^f \times \ldots \times G_n^f) \wedge} \\ &\boldsymbol{diag}(G_1^f \times G_2 \times \ldots \times G_n^f) \wedge \\ &\vdots \\ &\boldsymbol{diag}(G_1^f \times G_2^f \times \ldots \times G_n) \end{aligned}$$

Theorem 4. *Let $G_1, G_2, \ldots, G_n$ be n components modeled by LTSs, then*

$$\begin{aligned} &\boldsymbol{diag}(G_1) \wedge \boldsymbol{diag}(G_1 \times G_2^f \times \ldots \times G_n^f) \wedge \\ &\boldsymbol{diag}(G_2) \wedge \boldsymbol{diag}(G_1^f \times G_2 \times \ldots \times G_n^f) \wedge \\ &\vdots \\ &\underbrace{\boldsymbol{diag}(G_n) \wedge \boldsymbol{diag}(G_1^f \times G_2^f \times \ldots \times G_n)} \\ &\Downarrow \\ &\boldsymbol{diag}(G_1 \times G_2 \times \ldots \times G_n) \end{aligned}$$

Their proofs can be inferred directly from results that can be found in [3].

When the faults can occur in every component and $G_1^f \times G_2^f \times \ldots \times G_n \neq G_1 \times G_2 \times \ldots \times G_n$, our approach shows important advantages, however in the cases where $G_1^f \times G_2^f \times \ldots \times G_n = G_1 \times G_2 \times \ldots \times G_n$, the whole product is analyzed and the computation time of our method is equal to the classic one. Nevertheless, when a diagnosability analysis is performed it is because it is known

that several faults can occur in different components and it is more likely that $G_1^f \times G_2^f \times \ldots \times G_n$ is smaller than $G_1 \times G_2 \times \ldots \times G_n$.

Moreover, the diagnosability analysis of each component and $G_1^f \times G_2^f \times \ldots \times G_n$ can be tested in parallel, allowing parallel analysis of diagnosability.

4 The DADDY tool

In the previous section we try to minimize the information that components needs to share to be able to decide the diagnosability property of the whole system. We now present our tool, called DADDY (from Distributed Analysis for distributed Discrete sYstems). DADDY implements the method presented above and the classic one (where the synchronous product is computed before the diagnosability analysis is performed). The tool is written in Python and has GNU GPL v3 license. It uses a standard format (.aut) for the description of each component and it also allows to see a graphical representation of the system. It can be downloaded from [2].

The tool receives as inputs the components of the system. These inputs are assumed to be diagnosable, if not, an alert message is returned. If the specifications, meaning the non faulty components, are not given, systems G_j^f, for $j \neq i$, are computed following Algorithm 1. Hence G_j^f is synchronized with G_i, and its diagnosability is checked using the twin plant method from [8]. Also, time t_i of such computation is registered.

As soon as it is known that a component interacting with fault free versions of the other ones is non diagnosable, applying Theorem 3, a non diagnosable verdict is returned. Moreover, using the fact that it is a distributed computation, when we find a non diagnosable component, the computation of all others components can be stopped. So, the resulting time of such computation is $min(t_i)$ with $1 \leq i \leq n$.

On the other hand, if every component interacting with the fault free version of the other ones is diagnosable, using the assumption that every G_i is diagnosable by its own, we can conclude that $G_1 \times \ldots \times G_n$ is diagnosable applying Theorem 4. In this case, the diagnosability of every component is computed (in parallel) and the required time is $max(t_i)$ with $1 \leq i \leq n$.

We can see in table from Figure 5 that the diagnosability analysis results obtained by DADDY are consistent with the ones presented in our previous examples. We can also see that our method can be almost ten times faster than the classical one. If we consider systems n_1, n_2, n_3 from exaples/sample5 in [2], a non diagnosable result is obtained (as $n_1^f \times n_2 \times n_3^f$ is not diagnosable) in 0.16974902153 seconds with our method while the classical one does not reach a result after more than 24 hours. This shows an important improvement with respect to the classical method when the number of components grows.

5 Conclusions and Future Work

We have presented a new framework for the distributed diagnosability analysis of concurrent systems. We remove the assumption that a fault can only occur in a single component (which is usually made in distributed systems) and allow to analyze more general systems. The method

System	Diagnosable	Our method	Classic method
$A \times B$	yes	0.0027251243	0.024051904
		0.0028400421	0.023932933
		0.0028848648	0.024003028
		0.0029160976	0.025793075
		0.0032229423	0.023809194
$C \times D$	no	0.0041198730	0.015272855
		0.0040440559	0.015629053
		0.0042178630	0.015436887
		0.0040760040	0.009753942
		0.0047080516	0.015598058

Figure 5: Diagnosis results in seconds unit

presented in this paper parallelized the analysis leading, in general, to an important reduction in the computing time. The theoretical results are illustrated by several examples and supported by experimental results obtained with the DADDY tool.

We plan to continue trying to keep reducing the system in order to obtain minimal components from which we can infer the diagnosability of the original global system. In addition, we intend to relax the assumption that the communicating (synchronizing) events are observable.

Furthermore, even if the framework presented in this paper allows the distribution of the analysis, the formalism to model the systems is still sequential (product of LTSs) and can suffer of state space explosion making the twin plant method to check its diagnosability still prohibitive. We are working to extend such analysis to concurrent models such as Petri Nets.

Bibliography

[1] C. Baier and J-P. Katoen. *Principles of model checking*. MIT Press, 2008.

[2] G. Bonigo. Daddy. `https://code.google.com/p/daddy/`, 2012.

[3] G. Bonigo and L. Brandán-Briones. *Trabajo Final para la Licenciatura en Ciencias de la Computación: Análisis de Diagnosticabilidad en Sistemas Distribuidos*. http://www.famaf.unc.edu.ar/institucional/biblioteca/ trabajos/638/16627.pdf, 2012.

[4] L. Brandán-Briones, A. Lazovik, and P. Dague. Optimizing the system observability level for diagnosability. In *ISoLA*, pages 815–830, 2008.

[5] L. Brandán-Briones and A. Madalinski. Bounded predictability for faulty discrete event systems. In *SCCC*, pages 815–830, 2011.

[6] R. Debouk, S. Lafortune, and D. Teneketzis. Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems*, 10(1-2):33–86, 2000.

[7] S. Genc and S. Lafortune. Distributed diagnosis of discrete-event systems using petri nets. In *ICATPN*, pages 316–336, 2003.

[8] S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A polynomial algorithm for testing diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 46:1318–1321, 2000.

[9] A. Madalinski, F. Nouioua, and P. Dague. Diagnosability verification with petri net unfoldings. *Int. J. Know.-Based Intell. Eng. Syst.*, 14(2):49–55, April 2010.

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

[11] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of Discrete-Event Systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, September 1995.

[12] A. Schumann and J. Huang. A scalable jointree algorithm for diagnosability. In *AAAI*, pages 535–540, 2008.

[13] A. Schumann and Y. Pencolé. Scalable diagnosability checking of event-driven system. In *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI07*, pages 575–580, 2007.

[14] L. Ye and P. Dague. Diagnosability analysis for self-observed distributed discrete event systems. In *VALID*, pages 93–98, 2012.

MEALS Partner Abbreviations

SAU: Saarland University, D

RWT: RWTH Aachen University, D

TUD: Technische Universität Dresden, D

INR: Institut National de Recherche en Informatique et en Automatique, FR

IMP: Imperial College of Science, Technology and Medicine, UK

ULEIC: University of Leicester, UK

TUE: Technische Universiteit Eindhoven, NL

UNC: Universidad Nacional de Córdoba, AR

UBA: Universidad de Buenos Aires, AR

UNR: Universidad Nacional de Río Cuarto, AR

ITBA: Instituto Técnológico Buenos Aires, AR