Using Incremental SAT for Testing Diagnosability of Distributed DES

Hassan IBRAHIM[1] and **Philippe DAGUE**[1] and **Laurent SIMON**[2]
[1]LRI, Univ. Paris-Sud and CNRS, Orsay, France
hassan.ibrahim@lri.fr, philippe.dague@lri.fr
[2]LaBRI, Univ. Bordeaux and CNRS, Bordeaux, France
lsimon@labri.fr

Abstract

We extend in this work the existing approach to analyse diagnosability in discrete event systems (DES) using satisfiability algorithms (SAT), in order to analyse the diagnosability in distributed DES (DDES) and we test this extension. For this, we handle observable and non observable communication events at the same time. We also propose an adaptation to use incremental SAT over the existing and the extended approaches to overcome some of the limitations, especially concerning the length and the distance of the cycles that witness the non diagnosability of the fault, and improve the process of dealing with the reachability limit when scaling up to large systems.

1 Introduction

Diagnosis task is mainly using the available observations to explain the difference between the expected behavior of a system and its real behavior which may contain some faults. Many works have been done to study the automatic approaches to system fault diagnosis. They all try to deal with the main problem, i.e. the compromise between the number of possible diagnoses to the considered faults and the number of observations which must be given to make the decision. Diagnosis problem is NP-hard and one always needs to cope with an explosion in the number of system model states. Moreover, the diagnosis decision is not always certain, and thus running a diagnosis algorithm may not be accurate. For example, two sets of observations provided by different sets of sensors or at different times may lead to different diagnoses. This uncertainty raises the problem of diagnosability which is essential while designing the system model. After that, the model based diagnosis will be used in applications to explain any anomaly, with a guarantee of correctness and precision at least for anticipated faults.
Diagnosability of the considered systems is a property defined to answer the question about the possibility to distinguish any possible faulty behavior in the system from any other behavior without this fault (i.e., correct or with a different fault) within a finite time after the occurrence of the fault. A fault is diagnosable if it can be surely identified from the partial observation available in a finite delay after its occurrence. A system is diagnosable if every possible fault in it is diagnosable. This property provides information before getting into finding the explanations of the fault. It also helps in designing a robust system against faults and in positioning the sensors to manage the observation requirements. The main difficulty in diagnosability algorithms is related to the states number explosion. Another difficulty appears when checking diagnosability of a system which is actually diagnosable, i.e. the inexistence of a counter-example witnessing non diagnosability. Thus all possibilities need to be tested as for proving the non existence of a plan in a planning problem, and usually in this case some approximations are used to avoid exploring all the search space.
The paper is structured as follows. Section 2 will introduce the system transition models for centralized DES and recall the traditional definition of the diagnosability in those models and the state of the art of encoding this definition as a satisfiability problem in propositional logic. Section 3 will present our first contribution, an extension of this state of the art to DDES with observable and non observable communication events in the same model, and will give experimental results of this extension. Section 4 is devoted to our second contribution, using incremental SAT calls to overcome the limitation when the number of steps required to check diagnosability, i.e., the length of possible paths with cycles witnessing non diagnosability, is large, and will present experimental results showing how the method scales up. Section 5 will present related works and section 6 will conclude and give our perspectives for future work.

2 Using SAT in Diagnosability Analysis of Centralized Systems

We recall first the definitions of DES models we use and of diagnosability for these models.

2.1 Preliminaries

We will use finite state machines (FSM) to model systems. We define labeled transition systems following [1].

Definition 1. A **Labeled Transition System** (LTS) is a tuple $T = \langle X, \Sigma_o, \Sigma_u, \Sigma_f, \delta, s_0 \rangle$ where:

- X is a finite set of states,
- Σ_o is a finite set of observable correct events,
- Σ_u is a finite set of unobservable correct events,
- Σ_f is a finite set of unobservable faulty events,
- $\delta \subseteq X \times (\Sigma_o \cup \Sigma_u \cup \Sigma_f) \times X$ is the transition relation,
- s_0 is the initial state.

In [2] the authors used an equivalent but more compact representation than LTS for modeling systems in order to analyze their diagnosability: succinct transition systems, that exploit the regularity in the systems structures and are expressed in terms of propositional variables, which allowed them to translate more easily to a SAT problem the twin plant method proposed by [3] for checking diagnosability.

As we aim at studying the diagnosability of DDES using SAT solvers, we will follow the model of [2] who studied the same problem in centralized DES. It represents the system states by the valuations of a finite set A of Boolean state variables where valuation changes reflect the transitions between states according to the events. The set of all literals issued from A is $L = A \cup \{\neg a | a \in A\}$ and $\mathcal{L}$ is the language over A that consists of all formulas that can be formed from A and the connectives $\vee$ and $\neg$. We use the standard definitions of further connectives $\Phi \wedge \Psi \equiv \neg(\neg\Phi \vee \neg\Psi), \Phi \rightarrow \Psi \equiv \neg\Phi \vee \Psi$ and $\Phi \leftrightarrow \Psi \equiv (\Phi \rightarrow \Psi) \wedge (\Psi \rightarrow \Phi)$. The transition relation is defined to allow two or more events to take place simultaneously. Thus each event is described by a set of pairs $\langle \phi, c \rangle$ which represent its possible ways of occurrence by indicating that the event can be associated with changes $c \in 2^L$ in states that satisfy the condition $\phi \in \mathcal{L}$.

Definition 2. A **Succinct Transition System** (SLTS) is described by a tuple $T = \langle A, \Sigma_o, \Sigma_u, \Sigma_f, \delta, s_0 \rangle$ where:

- A is a finite set of state variables,
- Σ_o is a finite set of observable correct events,
- Σ_u is a finite set of unobservable correct events,
- Σ_f is a finite set of unobservable faulty events,
- $\delta : \Sigma = \Sigma_o \cup \Sigma_u \cup \Sigma_f \rightarrow 2^{\mathcal{L} \times 2^L}$ assigns to each event a set of pairs $\langle \phi, c \rangle$,
- s_0 is the initial state (a valuation of A).

It is straightforward to show that any LTS can be represented as an SLTS (one takes $\lceil log(|X|) \rceil$ Boolean variables and represents states by different valuations of these variables; one assigns to each occurence of an event e labeling a transition (x, e, y) a pair $\langle \phi, c \rangle$, with ϕ expressing the valuation of x and c the valuation changes between x and y). And reciprocally any SLTS can be mapped to an LTS (see Definition 2.4 in [2]).

The formal definition of diagnosability of a fault f in a centralized system modeled by (an LTS or SLTS) T was proposed by [1] as follows:

Definition 3. Diagnosability. A fault f is diagnosable in a system T iff

$$\exists k \in N, \forall s^f \in L(T), \forall t \in L(T)/s^f, |t| \geq k \Rightarrow$$
$$\forall p \in L(T), (P(p) = P(s^f.t) \Rightarrow f \in p).$$

In this formula, $L(T)$ denotes the prefix-closed language of T whose words are called trajectories, s^f any trajectory ending by the fault f, $L(T)/s$ the post-language of $L(T)$ after s, i.e., $\{t \in \Sigma^* | s.t \in L(T)\}$ and P the projection of trajectories on observable events. The above definition states that for each trajectory s^f ending with fault f in T, for each t that is an extension of s^f in T with enough events, every trajectory p in T that is equivalent to $s^f.t$ in terms of observation should contain in it f. As usual, it will be assumed that $L(T)$ is live (i.e., for any state, there is at least one transition issued from this state) and convergent (i.e., there is no cycle made up only of unobservable events).

A system T is said to be diagnosable iff any fault $f \in \Sigma_f$ is diagnosable in T. In order to avoid exponential complexity in the number of faults during diagnosability analysis, only one fault at a time is checked for diagnosability. It will thus be assumed in the following that there exists only one fault event f ($\Sigma_f = \{f\}$), without restriction on the number of its occurrences. Diagnosability checking has been proved in [3] to be polynomial in the number $|X|$ of states for LTS, so exponential in the number $|A|$ of state variables for SLTS (actually the problem is NLOGSPACE-complete for LTS and PSPACE-complete for SLTS [4]).

2.2 SLTS Diagnosability as Satisfiability

An immediate rephrasing of the definition 3 shows that T is non diagnosable iff it exists a pair of trajectories corresponding to cycles (and thus to infinite paths), a faulty one and a correct one, sharing the same observable events. Which is equivalent to the existence of an ambiguous (i.e. made up of pairs of states respectively reachable by a faulty path and a correct path) cycle in the product of T by itself, synchronized on observable events, which is at the origin of the so called *twin plant* structure introduced in [3]. This non diagnosability test was formulated in [2] as a satisfiability problem in propositional logic. We recall below this encoding with the variables and the formulas used, where superscripts t refer to time points and (e_o^t) and $(\hat{e}_o^t)$ refer respectively to the faulty and correct events occurrences sequences (corresponding states being described by valuations of (a^t) and $(\hat{a}^t)$) of a pair of trajectories witnessing non diagnosability (so sharing the same observable events represented by (e^t) and forming a cycle). The increasing of the time step corresponds to the triggering of at least one transition and the extension by an event of at least one of the two trajectories. $T = \langle A, \Sigma_u, \Sigma_o, \Sigma_f, \delta, s_0 \rangle$ being an SLTS, the propositional variables are thus:

- a^t and $\hat{a}^t$ for all $a \in A$ and $t \in \{0, \ldots, n\}$,
- e_o^t for all $e \in \Sigma_o \cup \Sigma_u \cup \Sigma_f$, $o \in \delta(e)$ and $t \in \{0, \ldots, n-1\}$,
- $\hat{e}_o^t$ for all $e \in \Sigma_o \cup \Sigma_u$, $o \in \delta(e)$ and $t \in \{0, \ldots, n-1\}$,
- e^t for all $e \in \Sigma_o$ and $t \in \{0, \ldots, n-1\}$.

The following formulas express the constraints that must be applied at each time step t or between t and $t+1$.

1. The event occurrence e_o^t must be possible in the current state:

$$e_o^t \rightarrow \phi^t \quad \text{for } o = \langle \phi, c \rangle \in \delta(e) \tag{2.1}$$

and its effects must hold at the next time step:

$$e_o^t \rightarrow \bigwedge_{l \in c} l^{t+1} \quad \text{for } o = \langle \phi, c \rangle \in \delta(e) \tag{2.2}$$

We have the same formulas with $\hat{e}_o^t$.

2. The present value ($True$ or $False$) of a state variable changes to a new value ($False$ or $True$, respectively) only if there is a reason for this change, i.e., because of an event that has the new value in its effects (so, change without reason is prohibited). Here is the change from

$True$ to $False$ (the change from $False$ to $True$ is defined similarly by interchanging a and $\neg a$):

$$(a^t \wedge \neg a^{t+1}) \rightarrow (e^t_{i_{1_{o_{j_1}}}} \vee \cdots \vee e^t_{i_{k_{o_{j_k}}}}) \quad (2.3)$$

where the $o_{j_l} = \langle \phi_{j_l}, c_{j_l} \rangle \in \delta(e_{i_l})$ are all the occurrences of events e_{i_l} with $\neg a \in c_{j_i}$.
We have the same formulas with $\hat{a}^t$ and $\hat{e}^t_{i_{l_{o_{j_l}}}}$.

3. At most one occurrence of a given event can occur at a time and the occurrences of two different events cannot be simultaneous if they interfere (i.e., if they have two contradicting effects or if the precondition of one contradicts the effect of the other):

$$\neg(e^t_o \wedge e^t_{o\prime}) \quad \forall e \in \Sigma, \forall \{o, o\prime\} \subseteq \delta(e), o \neq o\prime \quad (2.4)$$

$$\neg(e^t_o \wedge e'^t_{o\prime}) \quad \forall \{e, e\prime\} \subseteq \Sigma, e \neq e\prime, \forall o \in \delta(e),$$
$$\forall o\prime \in \delta(e\prime) \text{ such that } o \text{ and } o\prime \text{ interfere} \quad (2.5)$$

We have the same formulas with $\hat{e}^t_o$.

4. The formulas that connect the two events sequences require that observable events take place in both sequences whenever they take place (use of e^t):

$$\bigvee_{o \in \delta(e)} e^t_o \leftrightarrow e^t \text{ and } \bigvee_{o \in \delta(e)} \hat{e}^t_o \leftrightarrow e^t \quad \forall e \in \Sigma_o \quad (2.6)$$

The conjunction of all the above formulas for a given t is denoted by $\mathcal{T}(t, t+1)$.
A formula for the initial state s_0 is:

$$\mathcal{I}_0 = \bigwedge_{a \in A, s_0(a)=1} (a^0 \wedge \hat{a}^0) \wedge \bigwedge_{a \in A, s_0(a)=0} (\neg a^0 \wedge \neg \hat{a}^0) \quad (2.7)$$

At last, the following formula can be defined to encode the fact that a pair of executions is found with the same observable events and no fault in one execution (first line), but one fault in the other (second line), which are infinite (in the form of a non trivial cycle, so containing at least one observable event, [1] at step n; third line), witnessing non diagnosability:

$$\Phi^T_n = \quad \mathcal{I}_0 \wedge \mathcal{T}(0,1) \wedge \cdots \wedge \mathcal{T}(n-1, n) \quad \wedge$$
$$\bigvee_{t=0}^{n-1} \bigvee_{e \in \Sigma_f} \bigvee_{o \in \delta(e)} e^t_o \quad \wedge$$
$$\bigvee_{m=0}^{n-1} \Big(\bigwedge_{a \in A} ((a^n \leftrightarrow a^m) \wedge (\hat{a}^n \leftrightarrow \hat{a}^m)) \wedge \bigvee_{t=m}^{n-1} \bigvee_{e \in \Sigma_o} e^t \Big)$$

From this encoding in propositional logic, follows the result (theorem 3.2 of [2]) that an SLTS T is not diagnosable if and only if $\exists n \geq 1, \Phi^T_n$ is satisfiable. It is also equivalent to $\Phi^T_{2^{2|A|}}$ being satisfiable, as the twin plant states number is an obvious upper bound for n, but often impractically high (see in [2] some ways to deal with this problem).

3 Using SAT in Diagnosability Analysis of Distributed Systems

We extend from centralized systems to distributed systems the satisfiability framework of subsection 2.2 for testing diagnosability and we provide some experimental results.

[1] This verification that the cycle found is not trivial was not done in [2]; it is why the authors had to add for each time point a formula, not needed here, guaranteeing that at least one event took place, to avoid silent loops with no state change.

3.1 DDES Modeling

In order to model DDES with SLTS, we need to extend these ones by adding communication events *to each component*. So we use the following definition for a distributed SLTS with k different components (sites):

Definition 4. A **Distributed Succinct Transition System** (DSLTS) with k components is described by a tuple $T = \langle A, \Sigma_o, \Sigma_u, \Sigma_f, \Sigma_c, \delta, s_0 \rangle$ where (subscripts i refer to component i):

- A is a union of disjoint finite sets $(A_i)_{1 \leq i \leq k}$ of component own state variables, $A = \cup_{i=1}^k A_i$,
- Σ_o is a union of disjoint finite sets of component own observable correct events, $\Sigma_o = \cup_{i=1}^k \Sigma_{oi}$,
- Σ_u is a union of disjoint finite sets of component own unobservable correct events, $\Sigma_u = \cup_{i=1}^k \Sigma_{ui}$,
- Σ_f is a union of disjoint finite sets of component own unobservable faulty events, $\Sigma_f = \cup_{i=1}^k \Sigma_{f_i}$,
- Σ_c is a union of finite sets of (observable or unobservable) correct communication events, $\Sigma_c = \cup_{i=1}^k \Sigma_{ci}$, which are the only events shared by at least two different components (i.e., $\forall i, \forall c \in \Sigma_{ci}, \exists j \neq i, c \in \Sigma_{cj}$),
- $\delta = (\delta_i)$, where $\delta_i : \Sigma_i = \Sigma_{oi} \cup \Sigma_{ui} \cup \Sigma_{f_i} \cup \Sigma_{ci} \rightarrow 2^{\mathcal{L}_i \times 2^{L_i}}$, assigns to each event a set of pairs $\langle \phi, c \rangle$ in the propositional language of the component where it occurs (so, for communication events, *in each component separately* where they occur),
- $s_0 = (s_{0i})$ is the initial state (a valuation of each A_i).

In this distributed framework, synchronous communication is assumed, i.e., communication events are synchronized such that they all occur simultaneously in all components where they appear. More precisely, a transition by a communication event c may occur in a component iff a simultaneous transition by c occurs in all the other components where c appears (has at least one occurrence). In particular, all events before c in trajectories in all these components necessarily occur before all events after c in these trajectories. The global model of the system is thus nothing else that the product of the models of the components, synchronized on communication events. Notice that we allow in whole generality communication events to be, partially or totally, unobservable, so one has in general to wait further observations to know that some communication event occurred between two or more components. On the other side, assuming these communications to be faultless is not actually a limitation. If a communication process or protocol may be faulty, it has just to be modeled as a proper component with its own correct and faulty behaviors (the same that, e.g., for a wire in an electrical circuit). In this sense, communications between components are just a modeling concept, not subject to diagnosis. It will be also assumed that the observable information is global, i.e. centralized (when observable information is only local to each component, distributed diagnosability checking becomes undecidable [5]), allowing to keep definition 3 for diagnosability.

3.2 DSLTS Diagnosability as Satisfiability

Let T be a DSLTS made up of k components denoted by indexes i, $1 \leq i \leq k$. In order to express the diagnosability analysis of T as a satisfiability problem, we have to extend

the formulas of subsection 2.2 to deal with communication events between components. Let $\Sigma_c = \Sigma_{co} \cup \Sigma_{cu}$ be the communication events, with $\Sigma_{co} = \cup_{i=1}^{k} \Sigma_{coi}$ the observable ones and $\Sigma_{cu} = \cup_{i=1}^{k} \Sigma_{cui}$ the unobservable ones.

The idea is to treat each communication event as any other event in each of its owners and, as it has been done with events e^t for $e \in \Sigma_o$ for synchronizing observable events occurrences in the two executions, to introduce in the same way a global reference variable for each communication event at each time step, in charge of synchronizing any communication event occurrence in any of its owner with occurrences of it in all its other owners. We use one such reference variable for each trajectory, e^t and $\hat{e}^t$, for unobservable events $e \in \Sigma_{cu}$, and only one for both trajectories, e^t, for observable events $e \in \Sigma_{co}$ as it will also in addition play the role of synchronizing observable events between trajectories exactly as the e^t for $e \in \Sigma_o$. So, we add to the previous propositional variables the new following ones:

- $e_o^t, \hat{e}_o^t$ for all $e \in \Sigma_c, o \in \delta(e) = \cup_i \delta_i(e)$ and $t \in \{0, \ldots, n-1\}$,
- e^t for all $e \in \Sigma_c$, $\hat{e}^t$ for all $e \in \Sigma_{cu}$ and $t \in \{0, \ldots, n-1\}$.

Formulas in $\mathcal{T}(t, t+1)$ are extended as follows.

1. Formulas (2.1), (2.2), (2.3) and (2.5) extend unchanged to e_o^t and $\hat{e}_o^t$ $\forall e \in \Sigma_c$, expressing that a communication event must be possible and has effects in each of its owner components and that two such different events cannot be simultaneous if they interfere.
2. Formulas (2.4) extend to prevent two simultaneous occurrences of a given communication event in the same owner component, i.e. apply $\forall e \in \Sigma_c, \forall i, \forall \{o_i, o_i\prime\} \subseteq \delta_i(e), o_i \neq o_i\prime$ and the same with $\hat{e}$ (obviously they do not apply to different owner components, by the very definition of communication events).
3. Finally, the new following formulas express the communication process itself, i.e. the synchronization of the occurrences of any communication event e in all its owners components ($S(e)$ being the set of indexes of the owners components of e) and extend also formulas (2.6) to observable communication events:

$$\bigvee_{o_i \in \delta_i(e)} e_{o_i}^t \leftrightarrow e^t \text{ and } \bigvee_{o_i \in \delta_i(e)} \hat{e}_{o_i}^t \leftrightarrow \hat{e}^t \quad \forall e \in \Sigma_{cu}\ \forall i \in S(e)$$

$$\bigvee_{o_i \in \delta_i(e)} e_{o_i}^t \leftrightarrow e^t \text{ and } \bigvee_{o_i \in \delta_i(e)} \hat{e}_{o_i}^t \leftrightarrow e^t \quad \forall e \in \Sigma_{co}\ \forall i \in S(e)$$

The formula Φ_n^T is unchanged except that, in the verification that the found cycle (third line) is not trivial, any observable event can be used, so the final disjunct of events e^t is extended to all $e \in \Sigma_o \cup \Sigma_{co}$. We have thus the result that a DSLTS T is not diagnosable if and only if $\exists n \geq 1, \Phi_n^T$ is satisfiable.

3.3 Implementation and Experimental Testing

We have implemented the above extension in Java. We used the well designed API of the SAT solver Sat4j [6]. If more efficient solvers could have been chosen, it fitted well our clause generator written in Java and only a limited speed up can be awaited from C++ solvers (a speed up of 4, i.e. reduction of 75% of the runtime is often observed).

We have tested our tool on small examples with several communication events with multiple occurrences (three communicating components) with global communication (all components share the same event) or partial communication (only some components share the same event), as in Figure 1, which was the running example in [7].

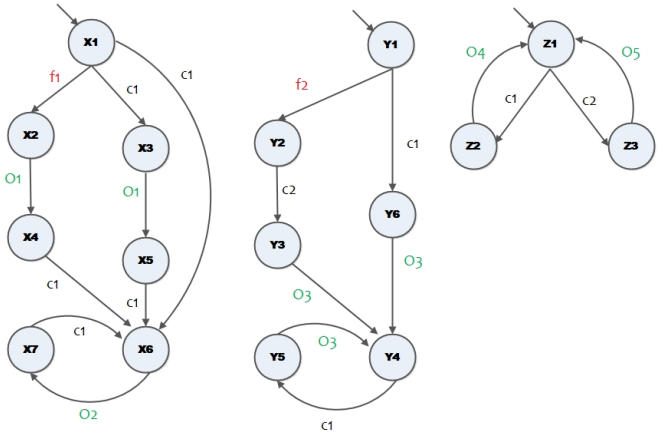

Figure 1: A DDES made up of 3 components C1, C2 and C3 from left to right. $c_{i,1 \leq i \leq 2}$ are unobservable communication events, $o_{i,1 \leq i \leq 5}$ are observable events and $f_{i,1 \leq i \leq 2}$ are faulty events.

The total number of propositional variables $VarsNum$ in the generated formula Φ_n^T after n steps is:
$VarsNum = n \times (2|A| + 3\sum_{i=1}^{Obs} ObOcc_i + \sum_{i=1}^{Faults} FaultOcc_i + 2\sum_{i=1}^{Unobs} UnobOcc_i)$, where:
$|A|$ is the total number of state variables,
Obs the total number of observable events,
$ObOcc_i$ the total number of occurrences of the observable event e_i,
$Faults$ the total number of faults,
$FaultOcc_i$ the total number of occurrences of the faulty event e_i,
$Unobs$ the total number of unobservable correct events,
$UnobOcc_i$ the total number of occurrences of the unobservable correct event e_i.
The results are in Table 1, where the columns show the system and the fault considered (3 cases), the steps number n, the numbers of variables and clauses and the runtime.

System	Fault	|Steps|	SAT?	|Variables|	|Clauses|	runtime(ms)
$C2$	$f2$	4	No	106	628	27
$C2$	$f2$	5	Yes	131	783	15
$C2, C3$	$f2$	5	No	225	1157	28
$C2, C3$	$f2$	32	No	1386	7340	641
$C2, C3$	$f2$	64	No	2762	14668	1422
$C2, C3$	$f2$	128	No	5514	29324	5061
$C2, C3$	$f2$	256	No	11018	58636	18970
$C2, C3$	$f2$	512	No	22026	117260	130164
$C2, C3$	$f2$	1024	No	44042	234508	548644
$C1, C2, C3$	$f1$	8	No	576	3546	91
$C1, C2, C3$	$f1$	9	Yes	646	3987	110

Table 1: Results on the example of Figure 1.

Which means that $f2$ is not diagnosable in $C2$ alone while it becomes diagnosable when synchronizing $C2$ and $C3$. For this last result, we have increased the steps number until reaching $2^{2|A|}$, which is the theoretical upper bound of the twin plant states represented in the logical formula. As in general it is not always possible to reach this bound in practice, we propose in section 4 using incremental SAT to improve the management of increasing steps number. While

$f1$ is not diagnosable even after synchronizing all three components together. Numbers of variables and clauses are small in comparison to what SAT solvers can handle (up to hundred thousands propositional variables and millions of clauses). These tests are mentioned as a proof of concept. However, to test the tool on larger systems and because of the absence of benchmark in the literature, we have created in subsection 4.2. an example that can be scaled up.

4 Adaptation to Incremental SAT Diagnosability Checking

We adapt satisfiability algorithms for checking diagnosability of both centralized (subsection 2.2) and distributed (subsection 3.2) DES in order to incrementally process the maximum length of paths with cycles searched for witnessing non diagnosability and we provide experimental results.

4.1 Diagnosability as Incremental Satisfiability

Two cases have to be distinguished while testing diagnosability using SAT solvers to verify the satisfiability of the logical formula Φ_n^T for a given n [2]. The first case is when we find a model for Φ_n^T, which *definitely* indicates the non diagnosability of the studied fault. The second case is when we do not find such a model: this result indicates just that the studied fault has not been found non diagnosable according to the value of n. In other words, after testing all the possible first n steps, we did not find a pair of executions of length at most n containing cycles such that one of them contains the fault and not the other and such that the two executions are equivalent in terms of observation. However, as the theoretical upper bound $n = 2^{2|A|}$ which would guarantee that the fault is actually diagnosable is often in practice unreachable, such a pair may exist for a greater value of n. Testing it means increasing n and *rebuilding the logical formula* Φ_n^T then recalling the SAT solver.

Instead, we propose to adapt the formula Φ_n^T in order to be tested in an *incremental* SAT mode by multiple calls to a Conflict Driven Clause Learning (CDCL) solver. Using CDCL solvers in a specialized, incremental, mode is relatively new but already widely used [8] in many applications. In this operation mode, the solver can be called many times with different formulas. However, solvers are designed to work with similar formulas, where clauses are removed and added from calls to calls. Learnt clauses can be kept as soon as the solver can ensure that clauses used to derive them are not removed. This is generally done by adding specialized variables, called *assumptions*, to each clause that can be removed. By assuming the variable to be $False$, the clause is activated and by assuming the variable to be $True$, the clause is trivially satisfied and no longer used by the solver. What is interesting for our purpose is that the CDCL solver can save clauses learnt during the previous calls and test multiple assumptions in each new call. This means that after n steps we hope that the solver will have learnt some constraints about the behavior of the system. Although we are interested in testing the diagnosability property on a defined system, this property is independent from the system behavior which can be learnt by the solver from the previous calls.

In order to extend the clauses representation given in subsections 2.2 and 3.2 to this mode of operation, we propose to divide the formula Φ_n^T in two parts. The first part $\mathcal{T}_n$ describes the first n steps, synchronized on the observations, of the behavior of both trajectories (represented by the conjunction of formulas $\mathcal{T}(t, t+1), 0 \leq t \leq n-1$, representing the $(t+1)$th step). The second part $\mathcal{D}_n$ describes the diagnosability property at step n, i.e., the occurrence of a fault in the n previous steps of the faulty trajectory (given by the formula $\mathcal{F}_n$) and the detection of a cycle at step n (given by the formula $\mathcal{C}_n$). So we obtain, for $n \geq 1$:

$$\Phi_n^T = \mathcal{T}_n \wedge \mathcal{D}_n$$

$$\mathcal{T}_n = \mathcal{I}_0 \wedge \bigwedge_{t=0}^{n-1} \mathcal{T}(t, t+1) \qquad \mathcal{D}_n = \mathcal{F}_n \wedge \mathcal{C}_n$$

$$\mathcal{F}_n = \bigvee_{t=0}^{n-1} \bigvee_{e \in \Sigma_f} \bigvee_{o \in \delta(e)} e_o^t$$

$$\mathcal{C}_n = \bigvee_{m=0}^{n-1} \Big(\bigwedge_{a \in A} ((a^n \leftrightarrow a^m) \wedge (\hat{a}^n \leftrightarrow \hat{a}^m)) \Big) \wedge \bigvee_{t=m}^{n-1} \bigvee_{e \in \Sigma_o} e^t \Big)$$

Add now at each step j a control variable h_j allowing to disable (when its truth value is $False$) or activate (when its truth value is $True$) the formulas $\mathcal{F}_j$ and $\mathcal{C}_j$ and keep at step n all these controlled formulas for $1 \leq j \leq n$. We obtain the following Ψ_n^T formula, for $n \geq 1$:

$$\Psi_n^T = \mathcal{T}_n \wedge \bigwedge_{j=1}^{n} \mathcal{D}_j{}' \quad \mathcal{D}_j{}' = \mathcal{F}_j{}' \wedge \mathcal{C}_j{}' \quad 1 \leq j \leq n$$

$$\mathcal{F}_j{}' = \neg h_j \vee \mathcal{F}_j \qquad \mathcal{C}_j{}' = \neg h_j \vee \mathcal{C}_j \quad 1 \leq j \leq n$$

We have thus the equivalence, for all $n \geq 1$:

$$\Phi_n^T \equiv \Psi_n^T \wedge h_n \wedge \bigwedge_{j=1}^{n-1} \neg h_j$$

This allows one, for all $n \geq 1$, to replace the SAT call on Φ_n^T by a SAT call on Ψ_n^T under the control variables setting given by $H_n = \{\neg h_1, \ldots, \neg h_{n-1}, h_n\}$ (indicated in a second argument of the call):

$$SAT(\Phi_n^T) = SAT(\Psi_n^T, H_n)$$

The idea is now to consider the control variables h_j as assumptions and use incremental SAT calls $IncSAT_j$ under varying assumptions, for $1 \leq j \leq n$. For this, we use the following recurrence relationships for both formulas Ψ_j^T and assumptions H_j:

$$\Psi_0^T = \mathcal{I}_0 \quad \Psi_{j+1}^T = \Psi_j^T \wedge \mathcal{T}(j, j+1) \wedge \mathcal{D}_{j+1}{}' \quad j \geq 0$$
$$H_1 = \{h_1\} \quad H_{j+1} = H_j[\{\neg h_j, h_{j+1}\}] \quad j \geq 1$$

where the notation $H_j[\{ass_i\}]$ means updating in H_j assumptions h_i by their new settings ass_i, i.e., in the formula above, replacing the truth value of h_j, which was $True$, by $False$, and adding the new assumption h_{j+1} with truth value $True$. From these relationships, the unique call to SAT under given assumptions $SAT(\Psi_n^T, H_n)$ can be replaced, starting with the set of clauses $\mathcal{I}_0$, by multiple calls, $0 \leq j \leq n-1$, to an incremental SAT under varying assumptions:

$$IncSAT_{j+1}(NewClauses_{j+1}, NewAssumptions_{j+1})$$
$$= IncSAT_{j+1}(\mathcal{T}(j, j+1) \wedge \mathcal{D}_{j+1}{}', \{\neg h_j, h_{j+1}\}) \quad (4.1)$$

If $IncSAT_j$ answers SAT, the search is stopped as non diagnosability is proved, if it answers UNSAT, then $IncSAT_{j+1}$ is called.

Notice that we used a unique assumption h_j for controlling both $\mathcal{F}_j$ and $\mathcal{C}_j$ as non diagnosability checking requires the presence of both a fault occurrence in the faulty trajectory and of a cycle. But the same framework allows the independent control of formulas by separate assumptions. For sake of simplicity, we also assumed we called $IncSAT$ at each step, but this is not mandatory and indexes j for the successive calls can be decoupled from indexes t for steps. We should also say that, even if $IncSAT$ allows us to reactivate an already disabled clause, we are sure in our case to never use this function (when h_k has been set to $False$, it always remains so) and we can thus force the solver to do a hard simplification process that removes the forgotten clauses permanently. As a result of our adaptation we will be able to scale up the size of the tested system and the distance and length of a cycle witnessing non diagnosability.

4.2 Experimental Results

We show in this subsection a comparison between our adapted version of subsection 4.1, that uses incremental SAT, and the previous versions, for centralized model (subsection 2.2 following [2]) and for distributed model (subsection 3.2). We have created the example in Figure 2 which contains $2k + 1$ components: one faulty component and two sets of k neighboring components. The faulty component has two separated paths, each one containing k different successive unobservable events c_i and ending with the same observable cycle of length 1, but only one of them contains the fault. The centralized model will be limited to this faulty component alone and thus in this case the events c_i, $1 \leq i \leq 2k$, are just unobservable events as is u. In the distributed model, these events c_i are communication events and the faulty component is considered with the other two sets of components, where each component in both sets shares one event c_i with the faulty component to ensure a number $2k$ of communications before arriving to the cycles that will witness the non diagnosability of the fault. Each set of components will be synchronized with only one path, either the faulty path or the correct one. This allows us to study the effect of the cycle distance in both models.

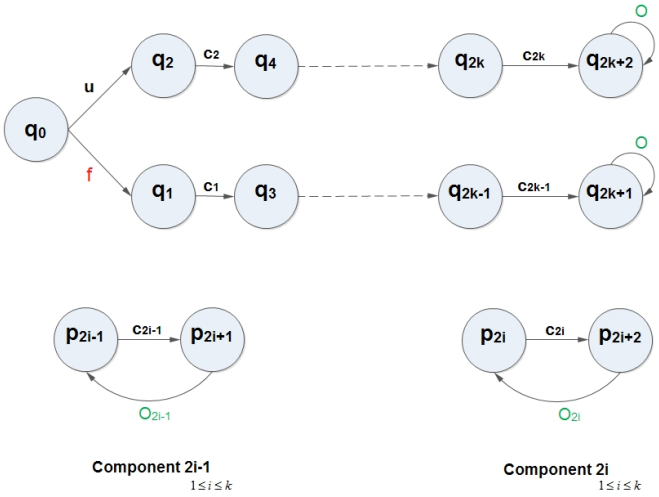

Figure 2: One faulty component that communicates with two sets of k components. Each set communicates with one path (resp. faulty and correct) in the faulty component.

The results are in Table 2 for the centralized model (for k = 18, 28, 38, 48, 58 and 98) and in Table 3 for the distributed model (for k = 3, 13, 23, 33, 43 and 63). The length of a pair of executions with cycles witnessing the non diagnosability of f in each example is $k + 2$ and we consider the satisfiability of the formula Φ^T_{k+2}, so the number of steps required for SAT to provide the answer *Yes* is: $|Steps] = k + 2$. In order to obtain a fair comparison between $IncSAT$, which manages internally by handling assumptions the successive satisfiability checks of increasing formulas for $j = 1, \ldots, k+2$, and SAT, for which $k+2$ successive calls are made to the solver with respective formulas Φ^T_n for $n = 1, \ldots, k+2$, the sum of the $k + 2$ runtimes of the SAT solver calls are considered in this case (last column in the tables).

|Steps|	|Clauses|	Inc. SAT(s)	SAT(s)
20	42,614	1.5	1.3
30	131,714	10.3	13.1
40	303,736	49.3	77.8
50	576,466	106	223
60	970,156	320	699
100	4,334,018	9410	13040

Table 2: Results on the faulty component of Figure 2.

|Steps|	|Comps|	|Clauses|	Inc. SAT(s)	SAT(s)
5	7	1,962	0.04	0.06
15	27	30,313	0.8	0.5
25	47	113,906	6.5	4.8
35	67	277,873	33.8	33.7
45	87	542,033	111	132
65	127	1,490,590	967	1090

Table 3: Results on the whole system of Figure 2.

Although these examples remain relatively simple and do not reflect any potential constraint that could be resumed by some learnt clauses (e.g. no interfering events), we can already notice the difference in runtime in favor of our incremental version in the centralized case and for the two largest values of k in the distributed case. This difference could be explained by the fact that generating all variables from the beginning for all time steps and for all events imply many meaningless clauses that would add a load on the solver in the version in [2], this load being avoided in our incremental version because of the clauses learnt by the CDCL SAT solver. From another side, we should say that generating in both versions all variables from the beginning has two main advantages: firstly, it allows the system description without unfolding it (even if this description is verbose); secondly, it allows the ordering of these variables by their time step in order to generate the constraints for only one time step and then get next steps constraints by just shifting the numbers (as we are representing the clauses in DIMACS format). One last point could help to a more efficient description of the system: in the succinct systems we represent all the occurrences of an event together, but in its SAT encoding we "unfold" this succinctness by generating for *each* occurrence n variables (for n time steps), even though logically only one of them will be assigned to *True*. We could thus mark this relation among these n copies by introducing a global cardinality constraint to express that these copies belong to only one occurrence of an event.

5 Selection of Related Works

The first introduction to the notion of diagnosability was by [1]. The authors studied diagnosability of FSM, as defined in definition 1. Their formal definition of diagnosability is the one we mentioned in definition 3. They introduced an approach to test this property by constructing a deterministic diagnoser. However, in the general case, this approach is exponential in the number of states of the system, which makes it impractical.

In order to overcome this limitation [3] introduced the *twin plant* approach, which is a special structure built by synchronizing on their observable events two identical instances of a nondeterministic fault diagnoser, and then searched for a path in this structure with an observed cycle made up of ambiguous states, i.e. states that are pairs of original states, one reached by going through a fault and the other not. Thus faults diagnosability is equivalent to the absence of such a path, called a *critical path*. This approach turns the diagnosability problem in a search for a path with a cycle in a finite automaton, and this reduces its complexity to be polynomial of degree 4 in the number of states (and exponential in the number of faults, but processing each fault separately makes its linear in the number of faults).

Let us mention here that the two previous works were interested in centralized systems with simple faults modeled as distinguished events. The first studies about fault patterns were introduced in [9] and [10] which generalize the simple fault event in a centralized DES to handle a sequence of events considered together as a fault, or handle multiple occurrences of the same fault or of different faults. More generally, a fault pattern is given as a suffix-closed rational events language (so by a complete deterministic automaton with a stable subset of final states).

The first work that addressed diagnosability analysis in DDES was [7]. A DDES is modeled as a set of communicating FSM. Each FSM has its own events set, communication events being the only ones shared by at least two different FSM. In [7] was introduced an incremental diagnosability test which avoids to build the twin plant for the whole distributed system if not needed. Thus one starts by building a local twin plant for the faulty component to test the existence of a local critical path. If such a path exists one builds the local twin checkers of the neighboring components. Local twin checker is a structure similar to local twin plant, i.e., where each path in it represents a pair of behaviors with the same observations, except that there is no fault information in it since it is constructed from non-faulty component. After constructing local twin checkers, one tries to solve the ambiguity resulting from the existence of a critical path in the local twin plant. This is done by synchronizing *on their communication events* this local twin plant with the local twin checker of one neighboring component. In other words, one tries to distinguish the faulty path from the correct one by exploiting the observable events in the neighboring components, because theses events occurrences that are consistent with the occurrences of the communication events could solve the ambiguity. The process is repeated until the diagnosability is answered, so only in the worst case has the whole system to be visited. Another important contribution in this work was to delete the unambiguous parts after each synchronization on the communication events, reducing thus the amount of information transferred to next check (if needed). The approach assumed simple faults.

The work by [11] has optimized the construction of local twin plants, by exploiting the fact that one distinguishes two behaviors (faulty and correct) and one synchronizes at two levels (observations first and communications later). It improved the construction of the twin plants proposed by [7] by exploiting the different identifiers given to the communication events at the observation synchronization level (depending on which instance, left or right, they belong to) to assign them directly to the two behaviors studied (left copy assigned to the faulty behavior, right copy to the correct one). This helped in deleting the redundant information, then in abstracting the amount of information to be transferred later to next steps if the diagnosability was not answered. The generalization to fault patterns in DDES was introduced by [12].

After the reduction of diagnosability problem to a path finding problem by [3], it became transferable to a satisfiability problem like it is the case for planning problems [13]. This was done by [2] which formulated the diagnosability problem (in its twin plant version) into a SAT problem, assuming a centralized DES with simple fault events. The authors represented the studied transition system by a succinct representation (cf. definition 2). This allows both a compact representation of the system states and a maximum amount of non interfering events to be fired simultaneously. Thus, they represented the system states by the valuation of a set of Boolean state variables ($\lceil log(q) \rceil$ state variables for q states) and the interference relation between two events according to the consistency among their effects and preconditions, one versus the other. They distinguished between an occurrence of an event in the faulty sequence or in the correct sequence by introducing two versions of it and constructed the logical formula expressing states transitions for each possible step in the system. Each step may contain simultaneous events that belong to faulty and correct sequences but must synchronize the occurrence of observable events whenever they take place. For a given bound n of paths length, they made the conjunct of these formulas for n steps and added the logical formula that represents the occurrence of the fault in the faulty sequence and the occurrence of a cycle in both sequences. The satisfiability of the obtained formula is equivalent to finding a critical path, i.e. to the non diagnosability of the fault (see subsection 2.2 for a summary of this approach). Although this approach allows one to test diagnosability in large systems, it has a limitation which is that we cannot dynamically increase n to ensure reaching more states while scaling up the size of the system where the cycles that witness non diagnosability can be very long. However the authors notice that we are not always forced to test all reachable states in many cases where an approximation for the reachable states can be applied, but without explaining explicitly how such an approximation can be found.

6 Conclusion and Future Works

By extending the state of the art works for centralized DES, we have expressed diagnosability analysis of DDES as a satisfiability problem by building a propositional formula whose satisfiability, witnessing non diagnosability, can be checked by SAT solvers. We allow both observable and non observable communication events in our model. Our expression of these communication events, which avoids

merging all their owner components, helps in reducing the number of clauses used to represent them and this reduction is proportional to the number of their occurrences. We have also proposed an adaptation of the logical formula in order to use incremental SAT calls helping managing the scaling up of the distance and the length of the intended cycles witnessing non diagnosability and thus the size of the tested system. Thus we exploited the clauses learnt about the system behavior in the previous calls. This approach is more practical and more efficient for complex systems than existing ones, as it avoids starting from scratch at each call.

We are now considering the extension of this work to fault patterns diagnosability [12]. We will use the same approach to express predictability analysis [14] as a satisfiability problem, for DES and DDES [15] and both for simple fault events and fault patterns [16] . Although our representation can be easily extended to deal with local observations (i.e., observable events in one component are observed only by this component), we know that in general diagnosability checking becomes then undecidable, e.g. when communication events are unobservable (obviously it remains decidable when these events are observable in all their owners) [5]. A future work will be to study decidable cases of diagnosability checking in DDES with local observations, e.g. assuming some well chosen communication events being observable. Another natural question is to study if the methods used in [7] and refined in [11] to check diagnosability in DDES in an incremental way in terms of the system components could be transposed as guiding strategies for some component incremental SAT based approach for testing diagnosability in DDES. Transposing in SAT these methods, based on building a local twin plant and local twin checkers for gaining efficiency with regards to a global checking, seems difficult. Basically, at any step k, corresponding to considering a subsystem made up of k components, these methods build all critical paths witnessing non diagnosability at the level of this subsystem and the incremental step, when adding a $(k+1)^{th}$ neighboring component, consists in checking the consistency of these pairs with the observations in the new component: only those pairs which can be consistently extended are kept, if any. In addition, in [11], only useful and abstracted information is kept from one step to the next one. With SAT, only one critical pair witnessing non diagnosability of the subsystem (i.e., a model for the formula) will be built. If it is not consistent, and thus disappears, when adding the $(k+1)^{th}$ component, diagnosability is not proven for all that: other critical pairs in the subsystem, not completely computed at step k, may exist and be extendible to step $(k+1)$. So, they have to be computed now, which limits the incremental characteristic of the approach. In the same way, abstracting some information is difficult to achieve with SAT. So, there is no evidence a priori that efficiency gain could be obtained by trying to develop a component incremental SAT based approach for testing DDES diagnosability.

References

[1] 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, 1995.

[2] J. Rintanen and A. Grastien. Diagnosability testing with satisfiability algorithms. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 532–537, 2007.

[3] 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(8):1318–1321, 2001.

[4] J. Rintanen. Diagnosers and diagnosability of succinct transition systems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 538–544, 2007.

[5] L. Ye and P. Dague. Undecidable case and decidable case of joint diagnosability in distributed discrete event systems. *International Journal On Advances in Systems and Measurements*, 6(3 and 4):287–299, 2013.

[6] D. Le Berre and A. Parrain. The sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.

[7] Y. Pencolé. Diagnosability analysis of distributed discrete event systems. In *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04)*, 2004.

[8] A. Nadel and V. Ryvchin. Efficient SAT solving under assumptions. In *Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing (SAT'12)*, 2012.

[9] T. Jéron, H. Marchand, S. Pinchinat, and M.-O. Cordier. Supervision patterns in discrete event systems diagnosis. In *Proceedings of the 8th International Workshop on Discrete Event Systems*, 2006.

[10] S. Genc and S. Lafortune. Diagnosis of patterns in partially-observed discrete-event systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 422–427. IEEE, 2006.

[11] L. Ye and P. Dague. An optimized algorithm for diagnosability of component-based systems. In *Proceedings of the 10th International Workshop on Discrete Event Systems (WODES'10)*, 2010.

[12] L. Ye, Y. Yan, and P. Dague. Diagnosability for patterns in distributed discrete event systems. In *Proceedings of the 21st International Workshop on Principles of Diagnosis (DX'10)*, 2010.

[13] H. Kautz and B. Selman. Planning as satisfiability. In *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI'92)*, pages 359–363, 1992.

[14] S. Genc and S. Lafortune. Predictability of Event Occurrences in Partially-observed Discrete-event Systems. *Automatica*, 45(2):301–311, 2009.

[15] L. Ye, P. Dague, and F. Nouioua. Predictability Analysis of Distributed Discrete Event Systems. In *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC-13)*, pages 5009–5015. IEEE., 2013.

[16] T. Jéron, H. Marchand, S. Genc, and S. Lafortune. Predictability of Sequence Patterns in Discrete Event Systems. In *Proceedings of the 17th World Congress*, pages 537–453. IFAC., 2008.