An Operative Formulation of the Diagnosability of Discrete Event Systems Using a Single Logical Framework

Florent Peres
Univ. Lille Nord de France, F-59000 Lille,
IFSTTAR, COSYS/ESTAS
F-59650 Villeneuve d'Ascq
florent.peres@ifsttar.fr

Mohamed Ghazel
Univ. Lille Nord de France, F-59000 Lille,
IFSTTAR, COSYS/ESTAS
F-59650 Villeneuve d'Ascq
mohamed.ghazel@ifsttar.fr

Diagnosability is a procedure whose goal is to determine whether any failure – or a class of failures – can be determined in finite time after its occurrence. Earlier works on diagnosability of discrete event systems (DES) establish some intermediary models from the analyzed model and then call some procedures to check diagnosablity based on these models, while recent works try to give a diagnosability formulation as a model-checking problem. However, there still lacks a *single* framework able to handle both of the diagnosability issues: how to model the problem? and how to decide it? In this paper, we build on some existing works which have formally established necessary and sufficient conditions for diagnosability of DES and we propose a generic operative formulation of diagnosability using the μ-calculus logic, which allows resolving the diagnosability issue within a single formalism.

Diagnosis, Diagnosability, Monitoring, Discrete event system, μ-calculus

1. INTRODUCTION

Fault detection and isolation (FDI) is a crucial task, both for safety and productivity reasons. Moreover, systems become more and more complex, thus making monitoring/diagnosis a challenging task especially in automated systems. A typical issue to deal with when performing the diagnosis process is that of partial observability. Actually, it is often difficult and costly to detect all the changes that may occur within a complex system. Indeed, for technical and/or economic reasons, setting enough devices/sensors to catch all the information needed for control and supervision is generally unfeasible when dealing with large complex systems. Consequently, it becomes essential to develop efficient techniques to carry out FDI tasks. At a high level, discrete event models Cassandras (2008) are more convenient for diagnosis studies than continuous models, which are more appropriate for detailed levels Lin (1994). Basically, two main issues are tackled when dealing with diagnosis of discrete event models: examinating diagnosability and developing diagnosers. Diagnosability investigation is performed offline, and consists to determine whether every fault -or category of faults- can be detected and identified in finite time, consecutively to its occurrence. The diagnoser implementation issue comes next. The Diagnoser ensures online monitoring and determines whether the system behavior is faulty and which type is the fault.

Diagnosability of DES has been defined first in the seminal work of Sampath (1995). Some slight variations can also be found like for instance I-diagnosability which makes fault determination conditioned by the occurrence of some indicator events. Such definitions state *when* a system is said diagnosable and some procedures based on intermediate automata models are then needed to actually investigate diagnosability. Later, model-checking techniques were used in Cimatti (2003), Huang (2004) and Grastien (2009), coming closer and closer to an operative definition. Nevertheless, these works all have a common point: they must first build an intermediary model of the behavior, called *twin plant*, before being able to apply model-checking. Thus, none of those works actually gives a unified operative definition. Such a definition must use a language whose semantics describes how to achieve all the required steps (which is true for the latter cited works) *while* being able to express the problem as a whole.

In this paper, an operative definition of diagnosability is developed, while using a slight variant of μ-calculus[1], as it was initially proposed in Park (1976). This logic is basically a predicate calculus extended with traditional fix-point operators μ and ν. The benefits of such a logic is that it is extremely powerful from a theoretical point of view (even *modal* μ-calculus can be expressed using μ-calculus), but especially that it is decidable for finite DES. Last but not least, there exists a tool, MEC 5 Griffault (2004), Vincent (2003), (now incorporated within ARC), for checking μ-calculus formulas on Altarica Arnold (1999) systems. By *operative* definition, we mean a definition that can be used directly to perform the diagnosability analysis.

Using a single logical framework to give a formulation of diagnosability does not necessarily mean that the problem has been simplified. Indeed, we will see that some of the steps of our formulation are quite close to the cited works, especially those using a *twin plant/verifier*. Nevertheless, the benefits of using a single framework is twofold: from a theoretical point of view, this shows the existence of such a logical operative framework able to express the problem *as a whole*. Then from the practical point of view, this provides a way to quickly implement and experiment diagnosability only by looking at the semantics, and hopefully will it be useful to extend the diagnosability facilities, as will be shown in the sequel.
The paper is organized as follows: In section 2, we introduce diagnosability of DES as well as some related notations. Section 3 is devoted to give an overview on the related works. In section 4, we discuss our μ-calculus formulation for diagnosability of DES. Section 5 gives a brief discussion about complexity and finally section 6 concludes the paper while driving some perspectives for this work.

2. DIAGNOSABILITY

2.1. Definition

To properly give the definition of diagnosability that we consider, we need to introduce some concepts and notations relative to language theory.

An *alphabet* is a set of characters or symbols, usually denoted Σ. A word is a sequence –or string– of characters. The set of all finite length words, composed of characters in Σ is denoted Σ^*. A language over Σ is a subset of Σ^*. A word is *empty* if it contains no letter, and is denoted ϵ. If w is a word then $|w|$ denotes its length, i.e. the length of the character sequence constituting w ($|\epsilon| = 0$). Two words a and b can be *concatenated* to form a new word denoted $a.b$ (or ab for short). The concatenation operation complies with the property $|ab| = |a| + |b|$. Using concatenation, it can be helpful to confuse the 1-length words ($|w| = 1$) with characters, and to consider the empty word ϵ as a "hidden" word/character. For $w = ab$, a and b are called *prefix* and *suffix* of w, respectively. A language L is called prefix-closed iff each prefix of each word in L is in L, i.e. $(\forall w \in L)(\{a \mid \exists b,\ w = ab\} \subseteq L)$.

For $w \in L$, w^i denotes the i^{th} character of w. For $i \in \mathbb{N} \setminus \{0\}\ \forall\ i > |w|$, $w^i = \epsilon$ and w^1 is the first character. By abuse of notation, if $\alpha \in \Sigma$ and $w \in \Sigma^*$, then $\alpha \in w$ iff $(\exists i)(w^i = \alpha)$.

Now, we will define diagnosability as given in the seminal work of Sampath (1995). Let L be a prefix-closed language on the set of events Σ. Σ is partitioned into Σ_o, the set of observable events, and Σ_u the set of all unobservable events, which is in turn partitioned into Σ_f, the set of all faulty events and $\Sigma_h = \Sigma_u \setminus \Sigma_f$ denoting the set of unobservable events which are not faulty (harmless).

We consider the following assumption: once a fault has occurred, the system remains irreparably faulty, that is to say faults are permanent. More precisely, given a sequence of alternating states and transitions, once a fault has occurred, every subsequent state eventually reached is considered *faulty*. From the monitoring point of view, this means we do not consider any maintenance operation that may be performed on the system.

Definition 1 Π_f *is a partition of the set of faults* Σ_f*. Each subset* i *is denoted* Σ_{f_i}*, that is* $\Pi_f = \biguplus_i \Sigma_{f_i}$*.*

Definition 2 $\Psi(\Sigma_{f_i})$ *is the set of sequences ending with a fault in* Σ_{f_i}*:* $(\forall s \in L)(\forall f_i \in \Sigma_{f_i})(s^{|s|} = f_i \Leftrightarrow s \in \Psi(\Sigma_{f_i}))$

Definition 3 $L/s = \{t \in \Sigma^* | s.t \in L\}$ *is the set of all suffixes of* s *in* L*.*

Definition 4 $(\forall i)((s^i \in \Sigma_o \Rightarrow P(s)^i = s^i) \wedge (s^i \in \Sigma_u \Rightarrow P(s)^i = \epsilon))$*, defines the projection* $P(s)$ *of sequence* s *on the set of observable events* Σ_o*: if* s^i *is observable, then* $P(s)^i = s^i$*, but if* s^i *is unobservable then* $P(s)^i = \epsilon$*.*

Definition 5 $P_L^{-1}(y) = \{s \in L | P(s) = y\}$ *gives all the sequences* s *in* L *for which the projection on* Σ_o *gives* y*.*

Definition 6 (Diagnosability Sampath (1995))

$$(\forall i \in \Pi_f)(\exists n_i \in \mathbb{N})(\forall s \in \Psi(\Sigma_{f_i}))(\forall t \in L/s)$$
$$(|t| \geq n_i \wedge w \in P_L^{-1}(P(s.t)) \Rightarrow \exists (f \in \Sigma_{f_i})(f \in w))$$

[1] Please note the absence of *"modal"*

Informally speaking, this definition states that a given language is diagnosable iff for any sequence w with the same observable projection than a faulty sequence $s.t$ with $s^{|s|} \in \Sigma_{f_i}$, and t is sufficiently long, then w necessarily contains a fault from Σ_{f_i}. Reasoning directly on languages is not always possible, because they are often infinite. An alternative is to use labeled transition systems (LTS).

Definition 7 *A labeled transition system (LTS) is a tuple* $(Q, q_0, \Sigma, \rightarrow)$*, in which:*

- Q *is a set of* states
- q_0 *is the initial state*
- Σ *is a set of events*
- $\rightarrow$: $Q \times \Sigma \times Q$ *is the transition relation*

We say that an LTS *recognizes* a word $w = x_1 \dots x_n$ iff $q_0 \xrightarrow{x_1} \dots \xrightarrow{x_n}$, or in other words if the sequence of events given by w can occur from q_0. The set of recognizable words forms the language recognized by the LTS. By extension, we say that an LTS is diagnosable (resp. not diagnosable), iff the language it recognizes is diagnosable (resp. non-diagnosable).

2.2. Example

For the LTS in Figure 1, the set of observable events is $\{a, b\}$, while f is a fault (thus unobservable). The LTS is not diagnosable and the explanation is quite simple: any word of the language $(ab)*$, coming from an observable projection, can originate either from a recognized sequence containing a fault (any word in $f(ab)^*$ satisfies this), or from a non-faulty sequence (any word in $(ab)^*$ also verifies this). Thus, there is no length n, such that after this length, one can always distinguish faulty sequences from normal ones, based on observed events.

On the other hand, the language recognized by the automaton of Figure 2 is diagnosable.

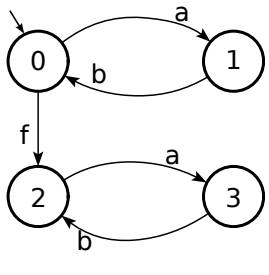

Figure 1: *A is not diagnosable*

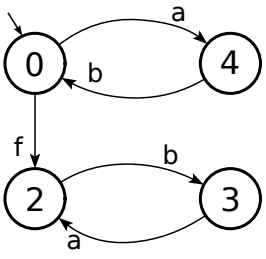

Figure 2: *B is diagnosable*

The difference between automata A and B is the appearance order of the observable events after the fault f. In B, a fault may appear either from the initial state, followed by an occurrence of b, or between two consecutive occurences of b. Therefore, when a fault f occurs, the projection on the set of observable events results in a sequence starting with b or it will include at least two consecutive b's. Both cases cannot be obtained without the occurrence of f.

3. RELATED WORKS

Several reference works in diagnosis of DES can be found in the literature, these works can be distinguished mainly according to the notations used, to the framework considered (centralized vs. distributed, untimed vs. timed), to the type of faults (permanent vs. transient) or also to the procedures adopted to investigate diagnosability. Sampath (1995) is a pioneer work in the DES diagnosis field, that has been improved in terms of computational complexity in Yoo (2002). Jiang (2001) proposes an efficient way to investigate diagnosability of DES modeled with state finite automata. In Basile (2010), Cabasino (2010), Genc (2007), Liu (2014), Ushio (1998) the authors analyze diagnosis on systems modeled by Petri nets. On the other hand, Tripakis (2002), Jiroveanu (2006), Ghazel (2009) and Basile (2013) deal with diagnosis within a timed framework, namely based on timed automata and petri net models. For a complete overview on the literature pertaining to the diagnosis of DES, the reader can refer to Zaytoon (2013) which offers a wide survey on the state of the art.

In this section, we will briefly discuss existing techniques on diagnosability of DES using a logical framework for which there is a well defined operational semantics. One may distinguish two sub-issues: the first is about developing a diagnosability decision procedure transformed into a *model-checking* problem; and the second is related to the specification of the faulty behavior.

3.1. Twin Plant

Although diagnosability is not defined using an "operative" logical framework, the originality of this work comes from its efficient decision procedure, which has been widely reused in several subsequent works. The authors of Jiang (2001) use the same definition of diagnosability as in Sampath (1995) and propose a polynomial algorithm, thus optimizing the diagnosability decision in comparison with that of Sampath (1995). The main idea is to drop the use of a diagnoser when checking whether a model is diagnosable.

The decision procedure is composed of three steps. From the original LTS $G = (X, q_0, \Sigma, \rightarrow)$, one proceeds as follows:

• Construction of $G_o = (X_o, q_0, \Sigma_o, \rightarrow_o)$, the "observable" version of G:

a) $X_o = \{(x, f) \mid x \in Q_1 \cup \{q_0\} \wedge f \subseteq \mathcal{F}\}$, where $Q_1 = \{x \in Q \mid (\exists x' \in Q)(\exists e \in \Sigma_o)(x' \xrightarrow{e} x)\}$, i.e. X_o is the set of states which are the destination of an observable transition (Q_1), plus the initial state q_0. Each state is labeled with a set of faults f, indicating which ones may have occurred before reaching this state.

b) $\rightarrow_o$: $X_o \times \Sigma_o \times X_o$ is such that $(x, f) \xrightarrow{e \in \Sigma_o}_o (x', f')$ iff $(\exists \sigma \in \Sigma_u^*, x'' \in X)\big(x \xrightarrow{\sigma} x'' \xrightarrow{e} x' \wedge f' = \{f_i \mid \sigma^j \in \Sigma_{f_i}\} \cup f\big)$. This is the observable reachability: if a state x' is reachable from a state x by an unobservable sequence, or the empty sequence, directly followed by an observable event e, then (x, e, x') is in $\rightarrow_o$.

• Construction of $G_d = (G_o||G_o) = (X_o \times X_o, (q_o, q_o), \Sigma_o, \Rightarrow)$, where $||$ is the usual parallel composition, for which only the following rule applies, because the two composed LTS are the same (clones):

$$\frac{q \xrightarrow{a \in \Sigma_o}_o q' \quad r \xrightarrow{a \in \Sigma_o}_o r'}{q||r \overset{a}{\Rightarrow} q'||r'}$$

• And finally, a cycle-checking pass is needed within G_d: for every cycle $q_1 \overset{\sigma}{\Rightarrow} q_n \overset{a}{\Rightarrow} q_1$, where $\sigma \in \Sigma_o^*$ and $a \in \Sigma_o$, if $(\forall i, j \in [1, n])(q_i = (s, f^1) \wedge q_j = (s', f^2) \Rightarrow f^1 = f^2)$ then G is diagnosable.

3.2. Verifier techniqueYoo (2002)

3.3. Twin plant + "LTL"

Several approaches use logical formulas to partially state diagnosability. All these works have in common the fact that they first build an intermediary LTS, in such a way to make the diagnosability decision a bit easier.

To the best of our knowledge, the authors of Cimatti (2003) are the first who have used *model-checking* in order to decide diagnosability: first, the model describing the system behavior is composed with itself (by the means of the usual parallel composition), giving a new structure called *twin-plant*. Two sets of states give the diagnosis conditions: the states in the first set have not to be confused with those in the second set. Then from the *twin-plant*, one may check whether there exists a path reaching a *critical state* q, such that $q = (x_1, x_2)$ and x_1 (from the first component in the parallel product) satisfies a diagnosis condition, whereas x_2 (from the second component in the parallel product) satisfies another diagnosis condition. If such a critical state q is reached, then the original model is not diagnosable.

In Huang (2004), the diagnosability problem has been dealt with using a CTL^* formula for the first time. Once again, it is still necessary to first build a *twin plant* G_d. Unlike the approach of Cimatti (2003), the authors use the same definition as in Sampath (1995). The model expressing the system dynamics is extended with a new variable denoting fault occurrence (Boolean). The *twin plant* is thus analyzed to find states corresponding to the composition of a "faulty" state with a "normal" state (information given by the added boolean variable) by *model-checking* techniques. In Grastien (2009), the authors use a substantially similar method.

3.4. Fault specification

Diagnosability can be extended using a more general fault specification: instead of pointing only faulty states or events, one may also express that a given behavior is faulty (resp. normal) if it satisfies (resp. does not satisfy) a certain formula specifying the faulty (resp. safe) behavior.

In Jiang (2004), the authors use an LTL formula f to specify the boundary of the normal (safe) behavior: each state outside this boundary is considered as to be faulty. In particular, usual faults are specified using safety properties: if e is a faulty event, then $\Box\neg e$ gives the normal behavior. However, what is particularly interesting in this approach lies in the fact that faults may be much more subtle, as for instance: a *deadlock* (a blocking preventing any further action of the system); a *livelock* (the blocking of certain functionalities in the system: technically speaking, the system executes some tasks, but does not meet its functional requirements); the repeated occurrence of a given event: taken individually, each occurrence is not a fault; but the recurrence of the event denotes a faulty behavior; etc. For example, a nominal behavior could be the following: after a request, the system must answer by a response (i.e the system is reactive). To specify this requirement one can use the following formula: $\Box(request \rightarrow response)$. Any run which does not satisfy this property is considered as to be faulty.

In Jéron (2006), Jeron et al. reuse, while generalizing them, the ideas discussed previously. The main idea is to use the LTS model to specify both the behavior to be diagnosed (i.e the "faulty" behavior), and the model of the system to be monitored. The idea is likely to be inspired from the well-known technique called as "verification with observers": when it is not possible to express a given property using a logic (or when using a logical formula leads to unsatisfying performances), it remains possible to *instrument* the behavioral model in order to facilitate the expression of the property. This is exactly what is suggested here: instead of giving a different logical formula

for each type of faults to be diagnosed, the system model is composed with the fault model, then a method, which is besides *generic*, is applied to check diagnosability. In Jéron (2006), several fault patterns are given.
Concretely, given $\mathcal{G}_f$ the fault model and $\mathcal{G}_M$ the model to be diagnosed, $\mathcal{G}_\Omega = \mathcal{G}_f \times \mathcal{G}_M$ is first computed, then a determinization is operated on this LTS. Thereafter, on this determinized LTS, the unobservable events are abstracted thus obtaining $\mathcal{G}_\Omega^{obs}$. Finally this obtained LTS is composed with itself: $\mathcal{G}_{diag} = \mathcal{G}_\Omega^{obs} \times \mathcal{G}_\Omega^{obs}$. The decision process is thereby reduced to checking whether there is not a sequence indefinitely "undetermined", i.e for which one component in $\mathcal{G}_{diag}$ is faulty whereas the other is not. The system is then diagnosable iff such a sequence does not exist.

4. μ-CALCUL FORMULATION OF DIAGNOSABILITY

The goal of our formulation is twofold: to establish a homogeneous formal logical framework to specify the diagnosability problem, and to give a "decision algorithm", directly deduced from the μ-calculus semantics.

The logic that we use here is typically a predicate calculation extended with two fix-point operators which have been proposed first in Park (1976).

μ-calcul syntax

$$E ::= \top \mid \bot \mid \neg E \mid E \wedge E \mid E \vee E \mid (\exists X)(E) \mid E = E \mid R'(x_1, \ldots, x_n)$$

$$R' ::= V \mid \lambda X^n.E \mid \mu V.R' \mid \nu V.R'$$

$$R ::= \lambda X^n.E \mid \mu V.R' \mid \nu V.R'$$

where V is the set of relational variables and X is a set of variables. Writing V or X is a shortcut to mean: "any element of these sets". Moreover, $n \in \mathbb{N}, n \geq 1$ and finally $V \cap X = \emptyset$.

There are two entry points for this grammar: E and R. Each of these entry points defines a particular type of formula: E denotes boolean formulas, while R defines relations. A relation is defined by the set of elements respecting a given boolean formula, therefore, E-expressions are necessary to define a relation. Note, however, that in the sequel E will not be used as an entry point in our formulation.

Semantics

Throughout the rest of the paper, we write $\phi_{[y/x]}$ to say that every occurrence of x in ϕ is substituted by y. The semantics of μ-calculus expressions is defined on the complete lattice $\langle \mathcal{O}, \subseteq \rangle$, as follows:

- $[\![\top]\!] =$ **true**
- $[\![\bot]\!] =$ **false**
- $[\![\neg\phi]\!] =$ **not** $[\![\phi]\!]$
- $[\![\phi = \gamma]\!] = \phi$ **equals** γ
- $[\![\phi \wedge \gamma]\!] = [\![\phi]\!]$ **and** $[\![\gamma]\!]$
- $[\![\phi \vee \gamma]\!] = [\![\phi]\!]$ **or** $[\![\gamma]\!]$
- $[\![(\exists x)(\phi)]\!] =$**OR**$_{i \in \mathcal{O}} [\![\phi_{[i/x]}]\!]$
- $[\![R(y_1, \ldots, y_n)]\!] = (y_1, \ldots, y_n) \in R$
- $[\![\lambda(x_1, \ldots, x_n).\phi]\!] = \{(y_1, \ldots, y_n) \in \mathcal{O}^n \mid [\![\phi_{[y_1/x_1, \ldots, y_n/x_n]}]\!] =$ **true**$\}$
- $[\![\mu x.\phi]\!] = \bigcup_{i=0}^{\infty} S^i$, with $S^0 = \emptyset$ and $S^{i+1} = [\![\phi_{[S^i/x]}]\!]$
- $[\![\nu x.\phi]\!] = \bigcap_{i=0}^{\infty} S^i$, with $S^0 = \mathcal{O}$ and $S^{i+1} = [\![\phi_{[S^i/x]}]\!]$

The terms **true** and **false** are the basis $\mathbb{B}$ of the boolean algebra whose operators are the usual boolean operators **and**, **or** and **not**; **equals**: $\mathcal{O} \times \mathcal{O} \rightarrow \mathbb{B}$ allows the comparison of two elements in $\mathcal{O}$; **OR** is the disjunction on a set of boolean terms and finally $\in$ is the usual membership operator.

In $[\![\mu x.\phi]\!]$ and $[\![\nu x.\phi]\!]$, ϕ must be monotone, i.e. each occurence of x must be "covered" by an even number of negations.

The fix-point operator is an infinite union. However and according to the Knaster-Tarski theorem Tarski (1955), since $\mathcal{O}$ is a complete lattice, we know that this fix-point will be reached upon a finite number of iterations.

4.1. Diagnosability

Let $\langle Q, q_0, \Sigma, \rightarrow \rangle$, with $\rightarrow$: $Q \times \Sigma \times Q$, the LTS modeling the system for which we want to check diagnosability. We do not want to handle any item other than states and events, thus $\mathcal{O} = Q \cup \Sigma$. We also assume the existence of sets Σ_f of faulty events and Σ_o of observable events.

Firstly, we will introduce diagnosability in the same way as defined in Sampath (1995). Thus, several relations will be defined. In these relations faulty states are those which are reached after a fault event has occurred (starting from the initial state). Secondly, we show how this definition can be easily extended.

Informally speaking, a triplet (a, b, f) is element of the *UOReach* relation, means that there exists an unobservable path between a and b. This path is labeled with a boolean f which is true when at least

UOReach $= \mu X.\lambda(s,t,f).(\exists s')(\exists f')(\exists e)(\exists r)(\exists e')$
$$\left(\begin{array}{l} \left(s \xrightarrow{e} t \wedge s = q_0 \wedge \neg\Sigma_o(e) \wedge f = \Sigma_f(e)\right) \vee \\ \left(s \xrightarrow{e} t \wedge r \xrightarrow{e'} s \wedge \Sigma_o(e') \wedge \neg\Sigma_o(e) \wedge f = \Sigma_f(e)\right) \vee \\ \left(\begin{array}{ll} X(s,s',f') \wedge s' \xrightarrow{e} t & \wedge \\ f = (f' \vee \Sigma_f(e)) \wedge \neg\Sigma_o(e) & \end{array}\right) \end{array}\right)$$

Figure 3: *Unobservable reachability + identification of faulty/normal paths*

one of the events of this path is a fault; conversely f is false if there exists an unobservable normal (without fault) path between a and b. In order to reduce the size of this relation, only the states destination of an observable event, or the initial state, are considered as an origin of the unobservable paths. A graphical representation of this relation applied to the model of Figure 4 is given in Figure 5.
In Figures 5, 7, 10, 11, 12, 13, 15, 16, 17 and 19, T tag stands for TRUE (faulty path) and F for FALSE (no fault).

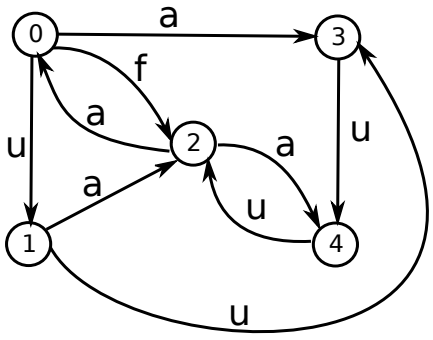

Figure 4: *The considered example*

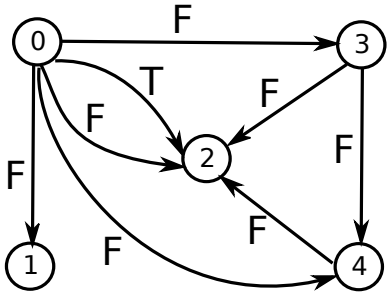

Figure 5: *Graphical representation of* UOReach *relation*

In the $UOReach$ formula, the μX fix-point operator indicates that the definition is recursive and that X denotes the set of elements in the relation at the previous step of the recursion. Since the samllest fix-point operator (μ) is used here, X is initially an empty set ($\emptyset$). The next operator, $\lambda(s,t,f)$, expresses that the relation *UOReach* is a ternary relation. The relation is defined by giving the valuation space that the three parameters, here (s,t,f) can have. *UOReach* is defined by three cases:

• Initially, s corresponds to the states issued from an observable transition from which an unobservable transition is possible, f indicates whether the event labelling the unobservable transition is faulty or not. Formally, this case can be written as: $(\exists r)(\exists e')(\exists e)(r \xrightarrow{e'} s \wedge \Sigma_o(e') \wedge s \xrightarrow{e} t \wedge \neg\Sigma_o(e) \wedge f = \Sigma_f(e))$

• By default, the initial state is considered as to be issued from an observable transition: $(s \xrightarrow{e} t \wedge s = q_0 \wedge \neg\Sigma_o(e) \wedge f = \Sigma_f(e))$

• The third case is the recursion operation: there exists a path between s and t if there is a triplet (s,s',f') in *UOReach* such that an unobservable transition links s' to t. The parameter f of the new triplet (s,t,f) is true if f' is true (a fault has already occurred between s and s'), or when the transition $s' \xrightarrow{e} t$ is faulty (i.e. $\Sigma_f(e)$): therefore we propagate the information that a fault is possible between s and s' to the new triplet. Formally, this case is expressed as follows: $(\exists s')(\exists f')(\exists e)(UOReach(s,s',f') \wedge s' \xrightarrow{e} t \wedge \neg\Sigma_o(e) \wedge f = f' \vee \Sigma_f(e))$

Nextobs $= \mu X.\lambda(s,e,t,f).(\exists s")(\exists e')(\exists f')(\exists s')$
$$\left(\begin{array}{ll} \left(\Sigma_o(e) \wedge s = q_0 \wedge s \xrightarrow{e} t \wedge \neg f\right) & \vee \\ \left(\Sigma_o(e) \wedge X(s",e',s,f') \wedge \neg f \wedge s \xrightarrow{e} t\right) & \vee \\ \left(\Sigma_o(e) \wedge UOReach(s,s',f) \wedge s' \xrightarrow{e} t\right) & \end{array}\right)$$

Figure 6: *Observable reachability + detection of faulty/normal paths*

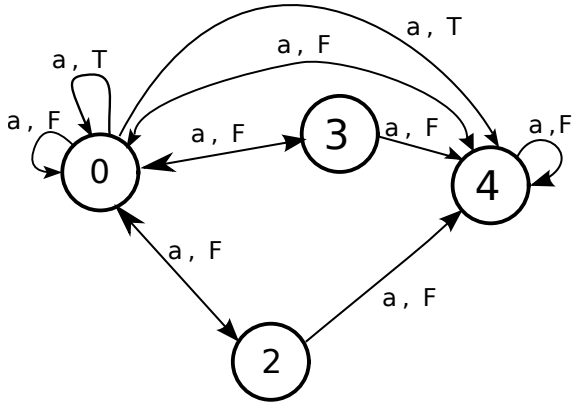

Figure 7: *Graphical representation of* Nextobs *relation*

The second step consists in determining the observable reachability of the LTS for which we examine dianosability. The observable reachability is an LTS which keeps only the observable events, and in which a transition links a state s to a state t iff it is possible to reach t from s through an unobservable sequence (may be ϵ) followed by an observable event. Here, the origin state s has to be either the initial state q_0, or a state destination of an observable event.

To each triplet (s, e, t) involved in an element of the observable reachability relation, we assign a boolean f denoting the existence of a faulty path (a path containing a fault) when f is true, and a normal path (without any fault) when f is false. The graphical representation of *Nextobs* relation is given in Figure 7. As for *UOReach*, *Nextobs* relation is defined according to two cases:

• Either $s \xrightarrow{e} t$ (with e observable) already exists, then we add (s, e, t, f) as is to *Nextobs* while putting f marker to false (since this is a faultless path from s to t). Formally, this can be written: $s \xrightarrow{e} t \wedge \Sigma_o(e) \wedge \neg f$. As for $UOReach$, the origin state must be either the initial state ($s = q_0$), or a state destination of an observable event ($X(s", e', s, f')$), which has been already captured in the $Nextobs$ relation, here.

• Or there exists an intermediary state s' such that $s' \xrightarrow{e} t$ where e is observable and s' is reachable from s through a sequence of unobservable events ($(s, s', f') \in UOReach$). In this case, the fault marker f is a copy of f': indeed only unobservable sequences containing a fault can turn f into true. Formally, this case can be expressed as follows: $UOReach(s, s', f) \wedge s' \xrightarrow{e} t \wedge \Sigma_o(e)$.

$$\textbf{Normal} = \mu X.\lambda(t).(\exists s)(\exists e) \begin{pmatrix} t = q_0 & \vee \\ (X(s) \wedge Nextobs(s, e, t, \bot)) & \end{pmatrix}$$

Figure 8: *Normal States*

Normal is a unary relation ($\lambda(t)$) containing the initial state q_0 as well as all the states, destination of an observable transition and reachable from q_0 by at least one normal path (cf. $Nextobs$). This relation will be useful in the sequel as if a given state t is reachable only through faulty paths, then the faults will propagate and all the subsequent reached states will be consequently faulty (no anymore ambiguity).

One may easily note that from the implementation point of view, sets $Nextobs$ and $Normal$ can be computed simultaneously using the same procedure, just by looking at the fault tagin $Nextobs$.

$$\textbf{Sameobs} = \mu X.\lambda(t, t').(\exists s)(\exists s')(\exists e) \begin{pmatrix} \begin{pmatrix} Normal(s) \wedge Nextobs(s, e, t, \bot) & \wedge \\ Nextobs(s, e, t', \bot) \wedge \neg(t = t') & \end{pmatrix} & \vee \\ \begin{pmatrix} X(s, s') \wedge Nextobs(s, e, t, \bot) & \wedge \\ Nextobs(s', e, t', \bot) \wedge \neg(t = t') & \end{pmatrix} & \end{pmatrix}$$

Figure 9: *Trace equivalence - Same observability*

This relation catches all the couples (t, t') such that states t and t' are different states that could be reached respectively by two normal (faultless) paths P_1 and P_2 having the same projection on Σ_o. $Sameobs$ allows us to keep all the states equivalent in terms of observation; the goal being to examine ambiguity in the system behavior starting from such pairs of states, as will be shown in the Amb relation. Two cases are considered:

• the first case holds when from the same "normal" state s, one can reach two different states t and t' respectively by two unobservable normal sequences, both followed by the same observable event e (cf. Figure 10). This case can be written as: $Normal(s) \wedge Nextobs(s, e, t, \bot) \wedge Nextobs(s, e, t', \bot) \wedge \neg(t = t')$.

• the second case corresponds to the recursion and consists in propagating the trace equivalence. Concretely from two indistinguishable states s and s' already in $Sameobs$ ($X(s, s')$), one can reach two different states t and t' while generating the same observable event e and without generating any fault for both paths (cf. Figure 11). This case can be expressed as: $X(s, s') \wedge Nextobs(s, e, t, \bot) \wedge Nextobs(s', e, t', \bot) \wedge \neg(t = t')$.

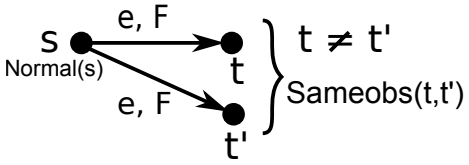

Figure 10: *First case*

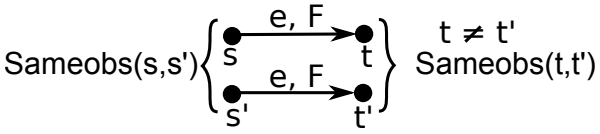

Figure 11: *Second case: propagation*

Proposition. $Sameobs(t, t') \implies \exists \sigma_1, \sigma_2 \in (\Sigma \setminus \Sigma_f)^*.\Sigma_o$ such that $\sigma_1 \neq \sigma_2 \wedge P_{\Sigma_o}(\sigma_1) = P_{\Sigma_o}(\sigma_2) \wedge q_0 \xrightarrow{\sigma_1} t \wedge q_0 \xrightarrow{\sigma_2} t'$

Proof. From the definition of $Sameobs$ given in Figure 9, (s, s') is in $Sameobs$ iff:

- either t and t' come from the same $Normal$ state s, by $Sameobs$ through a same observable event e, and without generating any fault (cf. Figure 12),

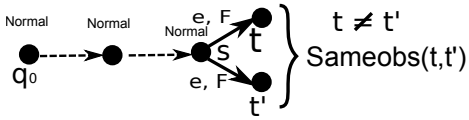

Figure 12: *Building (t, t') in the first case*

- or t and t' come respectively from s and s' by $Sameobs$ through a same observable event

e and without generating any fault such that (s, s') is also in $Sameobs$ (cf. Figure 13).

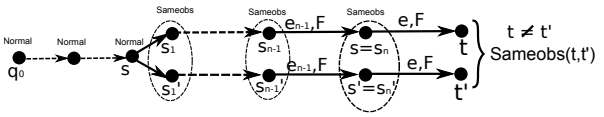

Figure 13: *Building (t, t') in the second case*

In the first case and given the definition of $Nextobs$, $\exists s \in Normal, \exists \sigma'_1, \sigma'_2 \in (\Sigma_u \setminus \Sigma_f)^*.\Sigma_o$ such that $\sigma'_1 \neq \sigma'_2$, $P_{\Sigma_o}(\sigma'_1) = P_{\Sigma_o}(\sigma'_2) = e \wedge s \xrightarrow{\sigma'_1} t \wedge s \xrightarrow{\sigma'_2} t'$. Besides, given the definition of $Normal$, it is easy to show (by induction) that $\exists \sigma' \in (\Sigma \setminus \Sigma_f)^*.\Sigma_o$ such that $q_0 \xrightarrow{\sigma'} s$. Then we have $q_0 \xrightarrow{\sigma'} s$, $s \xrightarrow{\sigma'_1} t$, $\sigma'_1 \neq \sigma'_2$ $s \xrightarrow{\sigma'_2} t'$. Hence, with $\sigma_1 = \sigma'.\sigma'_1$ and $\sigma_2 = \sigma'.\sigma'_2$, we have $\sigma_1, \sigma_2 \in (\Sigma \setminus \Sigma_f)^*.\Sigma_o \wedge \sigma_1 \neq \sigma_2 \wedge P_{\Sigma_o}(\sigma_1) = P_{\Sigma_o}(\sigma_2) \wedge q_0 \xrightarrow{\sigma_1} t \wedge q_0 \xrightarrow{\sigma_2} t'$.

In the second case, let us take back the computation of $Sameobs$ as given in definition 9. Assume there are n couples (s_i, s'_i) in $Sameobs$, $i \in [1, n]$ preceding (t, t') after having bifurcated from the same normal state s, as shown in Figure 13. Then, according to $Nextobs$ definition, $\exists \sigma_n, \sigma'_n \in (\Sigma_u \setminus \Sigma_f)^*.\Sigma_o$ such that $P_{\Sigma_o}(\sigma_n) = P_{\Sigma_o}(\sigma'_n) = e \wedge (s_n = s) \xrightarrow{\sigma_n} t \wedge (s'_n = s') \xrightarrow{\sigma'_n} t'$. This is also true for each pair of couples $(s_i, s'_i), (s_{i+1}, s'_{i+1})$ for $i \in [1, n-1]$. That is $\forall i \in$ [1,n-1], $\exists \sigma_i, \sigma'_i \in (\Sigma \setminus \Sigma_f)^*.\Sigma_o$ such that $P_{\Sigma_o}(\sigma_i) = P_{\Sigma_o}(\sigma'_i) \wedge s_i \xrightarrow{\sigma_i} s_{i+1} \wedge s'_i \xrightarrow{\sigma'_i} s'_{i+1}$.
Then by concatenating respectively σ_i sequences and σ'_i sequences, one can state that: $\exists \rho_1, \rho_2 \in (\Sigma \setminus \Sigma_f)^*.\Sigma_o, \rho_1 = \sigma_1 \ldots \sigma_n, \rho_2 = \sigma'_1 \ldots \sigma'_n$ such that $P_{\Sigma_o}(\rho_1) = P_{\Sigma_o}(\rho_2) \wedge s_1 \xrightarrow{\rho_1} t \wedge s'_1 \xrightarrow{\rho_2} t'$.
Moreover (s_1, s'_1) falls in the first case, then $\exists \tau_1, \tau_2 \in (\Sigma \setminus \Sigma_f)^*$, $\tau_1 \neq \tau_2$ such that $q_0 \xrightarrow{\tau_1} s_1 \wedge q_0 \xrightarrow{\tau_2} s'_1 \wedge P_{\Sigma_o}(\tau_1) = P_{\Sigma_o}(\tau_2) \wedge q_0 \xrightarrow{\tau_1} s_1 \wedge q_0 \xrightarrow{\tau_2} s'_1$.
Finally, by taking $\sigma_1 = \tau_1.\rho_1$ and $\sigma_2 = \tau_2.\rho_2$, we obtain: $\sigma_1, \sigma_2 \in (\Sigma \setminus \Sigma_f)^*.\Sigma_o$, $\sigma_1 \neq \sigma_2$, $P_{\Sigma_o}(\sigma_1) = P_{\Sigma_o}(\sigma_2) \wedge q_0 \xrightarrow{\sigma_1} t \wedge q_0 \xrightarrow{\sigma_2} t'$. □

$$\textbf{Amb} = \mu X.\lambda(t,t').(\exists s)(\exists s')(\exists e)(\exists f)$$
$$\left(\begin{array}{ll} \left(\begin{array}{l} Normal(s) \wedge Nextobs(s,e,t,\top) \wedge \\ Nextobs(s,e,t',\bot) \end{array} \right) & \vee \\ \left(\begin{array}{l} Sameobs(s,s') \wedge Nextobs(s,e,t,\top) \\ Nextobs(s',e,t',\bot) \end{array} \right) \wedge & \vee \\ \left(\begin{array}{l} X(s,s') \wedge Nextobs(s,e,t,f) \wedge \\ Nextobs(s',e,t',\bot) \end{array} \right) & \end{array} \right)$$

Figure 14: *Local Ambiguity*

Amb relation is quite simple and consists in identifying pairs of states t and t' locally ambiguous, i.e t and t' can be reached from q_0 by two sequences generating the same observation, but the first is faulty and not is the second. This can happen according to three cases:

• pairs (t, t') such that t and t' can be reached from the same normal state s, respectively through an unobservable faulty path followed by an observable event e in one hand, and on the other hand through a normal unobservable path (may be ϵ) followed by the same observable event e. This case can be expressed as:
$Normal(s) \wedge Nextobs(s, e, t, \top) \wedge Nextobs(s, e, t', \bot)$
(cf. Figure 15).

• when from two states s and s' such that $Sameobs(s, s')$, one can reach t and t' respectively through two unobservable sequences, σ_1 faulty and σ_2 normal, both followed by the same observable event e ($Sameobs(s, s') \wedge Nextobs(s, e, t, \top) \wedge Nextobs(s', e, t', \bot)$) as shown in Figure 16. This can be written:
$Sameobs(s, s') \wedge Nextobs(s, e, t, \top) \wedge Nextobs(s', e, t', \bot)$.

• the third case is when the ambiguity is obtained by "inheritence" from two ambiguous states s and s' repectively by a faulty unobservable sequence (which may be either normal or faulty, and possibly empty) followed by an observable event e; and on the other hand by a normal unobservable sequence followed by the same observable event e ($X(s, s') \wedge Nextobs(s, e, t, f) \wedge Nextobs(s', e, t', \bot)$). This case is depicted in Figure 17.

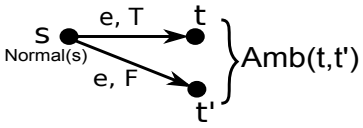

Figure 15: *Primitive ambiguity*

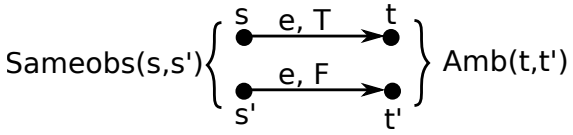

Figure 16: *Ambiguity from Sameobs states*

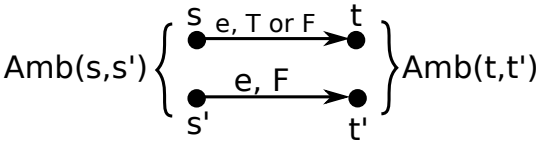

Figure 17: *Ambiguity by inheritance*

In order to examine diagnosability, one has to check whether there exists a cycle of ambiguous states. Searching such a cycle is not simple if we use the smallest fix-point operator μ. One possible way is to add, one by one, the elements that we are sure they do not make part of a cycle: if the obtained relation

(that we call *Noteveramb*) is equal to *Amb*, then there is no such a cycle. Conversely, the elements of *Amb* which are not in *Noteveramb* form at least one ambiguous cycle.

However, the μ-calculus offers another operator which will be very useful here: the greatest fix-point operator ν. Thanks to this operator, we will start from the maximal relation $Q \times Q$, then at each step we keep only the elements satisfying the equation until a fix-point is reached. Hence, defining $Everamb$ becomes simpler because only one case is possible (cf. Figure 18): a couple (s, s') is in $Everamb$ iff s and s' are ambiguous ($Amb(s, s')$) and iff there is at least one successor couple (t, t') by $Nextobs$ which fulfills both of the following conditions:

- is also ambiguous: $(Nextobs(s, e, t, \top) \wedge Nextobs(s', e, t', \bot))$, and
- is in $Everamb$ as well (recursivity): $(X(t, t'))$

Such a recursive definition implies that either of the following two cases holds:

1. the number of ambiguous couples in $Everamb$ is infinite, or
2. the ambiguous couples in $Everamb$ form a cycle.

Hence, since we deal with a finite state system, only the second case is possible. Thereby, each ambiguous couple (s, s') in $Everamb$ belongs to at least one cycle of ambiguous couples.

$$\textbf{Everamb} = \nu X.\lambda(s, s').(\exists t)(\exists t')(\exists e)(\exists f) \begin{pmatrix} Amb(s, s') \wedge Nextobs(s, e, t, \top) \wedge \\ Nextobs(s', e, t', \bot) \wedge X(t, t') \end{pmatrix}$$

Figure 18: *Cyclic ambiguity*

Figure 19 gives the $Everamb$ relation for the considered model. $Everamb$ is equal to Amb here, but for which we have shown in dotted line, some cycles in $Nextobs$ that satisfy Amb.

Definition 8 *An LTS is diagnosable according to a partition of faults Π_f iff for each part Σ_f in Π_f, $Everamb$ is empty.*

Theorem 1 *The previous definition of diagnosability and those of Sampath (1995) and Huang (2004) are equivalent.*

Proof.
It is proved in Huang (2004) that an LTS is diagnosable in the sense of Sampath (1995) iff the twin plant $G_d = (G_o||G_o)$ does not have any ambiguous cycle.

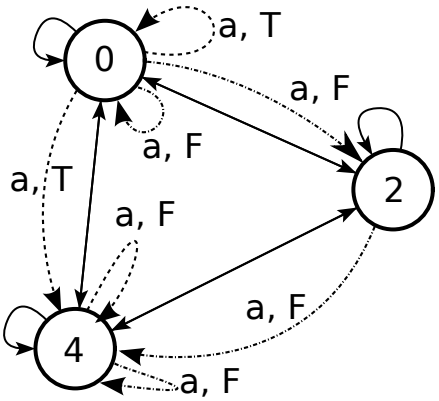

Figure 19: *Cyclic ambiguity*

On one hand, G_o of Huang (2004) is quite similar to $Nextobs$. The major difference being that $Nextobs$ holds fault information on transitions, while G_o holds them in states. This mostly impacts the number of states which would be more important in G_o, while $Nextobs$ may have more transitions.
On the other hand, the computation of $Sameobs$ together with Amb is equivalent to the composition procedure $G_d = (G_o||G_o)$. Actually, $Sameobs$ performs the composition between the normal paths which have an observational equivalence (both generate the same observation), whereas Amb performs the composition between a faulty path on one hand and a normal path on the other hand, when both paths have an observational equivalence.

Also, the definition of $Everamb$ does not allow multiple faults *directly*, but this does not break its generality: if there is more than one type of faults, i.e. if the partition Π_f contains more than one set, it is sufficient to check the emptiness of $Everamb$ for each of the sets individually, as stated in definition 8.

In G_d, a couple (x, y) is ambiguous iff, for a given fault f, $x = (q, F)$ and $f \in F$, while $y = (q', F')$ and $f \notin F'$. As (x, y) is in G_d, this means that q and q' are reachable by the same observable sequence. This corresponds to our definition of ambiguity. The couple of states (t, t') is ambiguous if either of the following conditions holds:

• there exists a state s belonging to $Normal$, i.e there exists a normal sequence $\sigma \in ((\Sigma_{uo} \setminus \Sigma_f)^* \Sigma_o)^*$ such that $q_0 \xrightarrow{\sigma} s$, and there exists an observable event e and two unobservable sequences σ' faulty and σ'' normal, such that $s \xrightarrow{\sigma'.e} t$ and $s \xrightarrow{\sigma''.e} t'$. This means we cannot decide, by only observing $P_{\Sigma_o}(\sigma).e$, whether a fault has occurred or not, hence the ambiguity.

• there exists a couple (s, s') of states in $Sameobs$, i.e there exist two normal sequences having the same

projection on Σ_o, $\sigma_1, \sigma_2 \in ((\Sigma_{uo} \setminus \Sigma_f)^*\Sigma_o)^*$ such that $q_0 \xrightarrow{\sigma_1} s$ and $q_0 \xrightarrow{\sigma_2} s'$, and (s, s') fulfills the following: from s there can be an observable step (in $NextObs$) $s \xrightarrow{e} t$ generating a fault and from s' there can be an observable step (in $NextObs$) $s' \xrightarrow{e} t'$ with no fault. Like previously, this means one cannot decide by only observing $P_{\Sigma_o}(\sigma).e$ whether a fault has occurred or not, thus the ambiguity.

• there exists a couple of states (s, s') in Amb, which means there exists a faulty sequence $\sigma_1 \in (\Sigma_{uo} * \Sigma_o)^*$ such that $q_0 \xrightarrow{\sigma_1} s$ and a normal sequence $\sigma_2 \in ((\Sigma_{uo} \setminus \Sigma_f)^*\Sigma_o)^*$ such that $q_0 \xrightarrow{\sigma_1} s'$ while σ_1 and σ_2 have the same observable projection, and (s, s') fulfills the following: there exist an observable event e, two unobservable sequences σ either faulty or not ($\in \Sigma_{uo}^*$), and σ' normal ($\in (\Sigma_{uo} \setminus \Sigma_f)^*$) such that $s \xrightarrow{\sigma.e} t$ and $s' \xrightarrow{\sigma'.e} t'$. Thereby, t can be reached from q_0 through a faulty path $\sigma_1\sigma e$, and t' can be reached from q_0 through a normal path $\sigma_2\sigma' e$ and both paths generate the same observation ($P_{\Sigma_o}(\sigma_1\sigma e) = P_{\Sigma_o}(\sigma_2\sigma' e)$). Like previously, this means we cannot decide based on the observed events if a fault has actually occurred, hence the ambiguity.

For every sequence of events, once a state is faulty, every successor is faulty as well. If a cycle is found in G_d such that it contains at least one composite state of a "faulty" and a "normal" states (i.e. ambiguity), then *every* state in that cycle is ambiguous too. This is why instead of finding cycles containing at least *one* ambiguous state as in Huang (2004), it is equivalent to say that each element of a cycle in $Everamb$ must be ambiguous. As shown when relation $Everamb$ has been introduced earlier, each element in this set belongs to at least in one cycle of ambiguous pairs. □

5. COMPLEXITY

We want to insist on the fact that the purpose of this article is *not* to propose a new algorithm. Indeed, even if the semantics is operational and allows computation, a direct implementation of that semantics would be far from being optimized. Nevertheless, we want to argue here that our formulation may be a good source for some new efficient algorithms. Indeed, where the other works *systematically* operate a product of the system to be diagnosed with itself (or a non-faulty version of itself), such a product is performed here according to some finer conditions. But before exploring what such an algorithm would look like, we wanted to explore the complexity of our approach, to see whether it is of the same complexity order, i.e. whether our formulation yields a polynomial complexity (if it is not, finding an efficient algorithm would have no sense!).

To estimate the complexity of the whole diagnosability decision process, we will seek for an upper bound on the complexity of each of the various formulas given in definitions 8 to 13.

Let us recall that these formulas are all based on fixed point operators. Because of their recursive nature and because the computation stops as soon as a new iteration does not change the result (the fixed point is reached), it is difficult to determine the worst case in a general way. Indeed, it is possible to find a worst case for a given (part of) formula, but it may very well be that this very worst case is at the same time the best case of another one (or at least, not its worst case), and the worst case of both formulas, when combined, generally cannot be the combination of the worst cases of each of the formulas.

Here we will take the worst case for each of the formulas for one single iteration and then, we will consider the maximum number of iterations leading to the fix point. This way, we are sure to get an upper bound of the complexity. Note that the result we will get is necessarily an overestimate of the real complexity. To picture that, let us consider the calculation of $UOReach$ (Definition 3). With the μ operator one starts with an empty set, and at the first iteration the triples (s, t, f) verifying the following conditions are added:

- s is either the initial state, or the target state of an observable transition; to check both conditions we must go through the whole transition relation (which has a -highly improbable- maximum of $|Q|.|\Sigma|.|Q|$ elements) and check for each triple (s, e, t) whether e is observable ($|\Sigma_o| \leq |\Sigma|$). At the end, these operations have the following complexity: $|Q|^2.|\Sigma|^2$.
- for all the states s found in the previous item (at most $|Q|$ states), (s, t, f) is in $UOReach$ at the first iteration if there exists (s, e, t) in the transition relation, such that e is not observable ($e \in \Sigma_u$). Like previously, we will consider Σ instead of Σ_u ($|\Sigma_u| \leq |\Sigma|$), to ease factorization. We obtain the following complexity: $|Q|^3.|\Sigma|^2$.
- f can be either **true** or **false**: the complexity of the previous item is then multiplied by 2.

This results in the following upper bound of the overall complexity: $|Q|^2.|\Sigma|^2 + 2|Q|^3|\Sigma|^2$ for the first iteration.

For the subsequent iterations, we start with the existing set of triples (s, s', f) (at most $2|Q|^2$ triples), and for all the (target) states s' of these triples, we determine the states t directly accessible by an unobservable transition e (complexity $2|Q|^2|\Sigma|^2$). So this gives the following upper bound of complexity: $2|Q|^4|\Sigma|^2$.

Regarding the number of iterations before reaching the fixed point, the worst case corresponds to the situation in which at each iteration, a single new triplet is added to $UOReach$: this means there are at most $|Q|^2|\Sigma|$ iterations.

We then obtain that an upper bound on the complexity of the $UOReach$ computation is $|Q|^6.|\Sigma|^3$. It is obvious that this bound is far from being optimal, but it allows us to determine the complexity class of our formulation. In the same way, we find that each of the formulas, has a polynomial complexity. Thus, we can say that analyzing the diagnosability on the basis of our formulation is of a polynomial computational complexity.

6. CONCLUSION

This work offers a new way to formulate diagnosability of DES using μ-calculus logic that we advocate to be a good formalism for a single formal framework to deal with diagnosability. While it theoretically defines the logical perimeter of the problem, it also allows the use of model checking techniques. This means taking advantage of already efficient tools and of mature techniques to circumvent, the best it can, the problem of combinatorial explosion, which is a well-known problem in *model-checking*, also called *the state space explosion* problem. The developed formulation is quite flexible and some extensions are being developed in order to tackle diagnosability issues under different contexts. Moreover, based on our logical formulation, we intend to develop diagnosers for online monitoring.

From a technical point of view, developing an on-the-fly algorithm to implement our formulation shall improve the efficiency of the computational complexity of the diagnosability analysis procedure Liu (2014). This issue will be investigated in our future works.
In addition, we believe that the flexibility of μ-calculus can be efficiently used for some other problems gravitating around diagnosability, as for fault specification of Jiang (2004), except that μ-calculus would be used instead of LTL. Its expressiveness, and the facilities it introduces are being studied.

The approach was tested with the MEC/ARC tools, but a prototype was also implemented to tackle the issue from the point of view of explicit exploration of the behavior (which is not allowed by MEC), but also to have a better understanding of the formulation complexity, and to explore optimization possibilities. As a side note, the prototype was relatively easily implemented (using Standard ML), and we think that this is in large part because of the simplicity of the μ-calculus semantics. Besides this *direct* implementation, we have also developed a second prototype based on a database management framework Ghazel (2012).

REFERENCES

A. Arnold, G. Point, A. Griffault and A. Rauzy (1999), The altarica formalism for describing concurrent systems, Fundam. Inf., 40(2-3):109-124.

F. Basile, P. Chiacchio and G. De Tommasi (2010), Petri nets via integer linear programming, Discrete Event Dynamic Systems, 10(1):71-77.

M.P. Cabasino, A. Giua and C. Seatzu (2010), Fault detection for discrete event systems using petri nets with unobservable transitions. Automatica, 46(9):1531-1539.

C.G. Cassandras and S. Lafortune (2008), Introduction to discrete event systems. Elsevier.

A. Cimatti, C. Pecheur and R. Cavada (2003), Formal verification of diagnosability via symbolic model checking, In IJCAI'03, pages 363369.

M. Cabasino F. Basile and C. Seatzu (2013), Marking estimation of time petri nets with unobservable transitions, In 18th IEEE Int. Conf. on Emerging Technologies and Factory Automation.

S. Genc and S. Lafortune (2007), Distributed diagnosis of place-bordered petri nets, IEEE Transactions on Automation Science and Engineering, 4(2):206-219.

M. Ghazel, A. Toguyéni and P. Yim (2009), State observer for des under partial observation with time petri nets, Discrete Event Dynamic Systems, 19(2):137-165.

M. Ghazel, F. Peres, A. Belhaj Alaya and A. Jemai (2012), A DBMS Framework for Diagnosability Analysis of Discrete Event Systems, The 42nd An- nual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'2012), Boston, USA.

A. Grastien (2009), Symbolic testing of diagnosability, International Workshop on Principles of Diagnosis, pages 131-138.

A. Griffault and A. Vincent (2004), The Mec 5 model-checker, in Rajeev Alur and Doron A. Peled, editors, Computer Aided Verification, volume 3114 of LNCS, pages 248-251.

Z Huang, S Bhattacharyya, V Chandra, S Jiang and R. Kumar (2004), Diagno- sis of discrete event systems in rules-based model using first-order linear temporal logic, in Proceedings of the American Control Conference.

T. Jéron, H. Marchand, S. Pinchinat and M-O. Cordier (2006), Supervision patterns in discrete event systems diagnosis, in 8th Workshop on Discrete Event Systems, WODES'06.

S. Jiang, Z. Huang, V. Chandra and R. Kumar (2001), A polynomial algorithm for testing diagnosability of discrete event systems, IEEE TAC, 46(8): 1318-1321.

S. Jiang and R. Kumar (2004), Failure diagnosis of discrete event systems with linear-time temporal logic fault specifications. IEEE TAC, 49:6:934-945.

G. Jiroveanu and R. Boel (2006), A distributed approach for fault detection and diagnosis based on time petri nets, Mathematics and Computers in Simulation, 70(5-6):287-313.

F. Lin (1994), Diagnosability of disrete event systems and its applications. JDEDS, 4:197-212.

B. Liu, M. Ghazel and A. Toguyéni (2014), Toward an efficient approach for diagnosability analysis of des modeled by labeled petri nets, in The 13th European Control Conference (ECC14), Strasbourg, France.

D. Park (1976), Finiteness is mu-ineffable. TCS, 3(2):173-181.

F. Peres, B. Berthomieu and F. Vernadat (2011), On the composition of time petri nets, Discrete Event Dynamic Systems, 21(3):395-424.

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen and D. Teneketzis (1995), Diagnosability of discrete-event systems. 40:1555-1575.

A. Tarski (1955), A lattice-theoretical fixpoint theorem and its applications, pa- cific journal of mathematics, Pacific Journal of Mathematics, 5(2):285- 309.

S. Tripakis (2002), Fault diagnosis for timed automata, in FTRTFT'02: Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, Springer-Verlag, pages 205-224, London, UK.

T. Ushio, I. Onishi and K. Okuda (1998), Fault detection based on petri net models with faulty behaviors, in Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pages 113-118.

A. Vincent (2003), Conception et réalisation d'un vérificateur de modles AltaRica - PhD thesis, Université des Sciences et Technologies - Bordeaux I.

T.S. Yoo and S. Lafortune (2002), Polynomial-time verification of diagnosability of partially observed discrete-event systems. IEEE Transactions on Automatic Control, 47(9):1491-1495.

J. Zaytoon and S. Lafortune (2013), Overview of fault diagnosis methods for discrete event systems. Annual Reviews in Control, 37(2):308-320.