A Predictability Algorithm for Distributed Discrete Event Systems

Lina Ye, Philippe Dague, Farid Nouioua

▶ **To cite this version:**

Lina Ye, Philippe Dague, Farid Nouioua. A Predictability Algorithm for Distributed Discrete Event Systems. The 17th International Conference on Formal Engineering Methods, Nov 2015, Paris, France. 10.1007/978-3-319-25423-4_13 . hal-01274813

HAL Id: hal-01274813
https://hal.science/hal-01274813v1

Submitted on 16 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Predictability Algorithm for Distributed Discrete Event Systems

Lina Ye[1,2], Philippe Dague[1], and Farid Nouioua[3]

[1] Université Paris-Sud, LRI, F-91405, Orsay, France
[2] CentraleSupélec, F-91192, Gif sur Yvette, France
[3] Université Aix-Marseille, LSIS, France
lina.ye@lri.fr,philippe.dague@lri.fr,farid.nouioua@lsis.org

Abstract. Predictability is considered as a crucial system property that determines with certainty the future occurrence of a fault based on a sequence of observations on system model. There are very few works done on the predictability problem for discrete event systems, which is however extremely important for developing critical complex systems. In this paper, we propose a formal sufficient and necessary condition for this property before presenting a new algorithm based on it, which is extendible from a centralized framework to a distributed one. Both are formally presented, as well as experimental results that show the efficiency of our approach.

1 Introduction

Fault diagnosis is a crucial and challenging task in the automatic control of complex systems [15, 19, 5, 14, 4, 8, 1, 9], whose very possibility depends on a system property called diagnosability. The diagnosability problem has received considerable attention in the literature. Diagnosability describes the system ability to determine whether a fault has effectively occurred based on the observations. In a given system, the existence of two infinite behaviors, with the same observations but exactly one containing the considered fault, violates diagnosability. The existing works search for such ambiguous behaviors both in centralized [17, 12, 16, 3, 7] and distributed [13, 18, 20] ways. The most classical method is to construct a structure called twin plant that captures all pairs of observationally equivalent behaviors to directly check the existence of such ambiguous pairs. However, sometimes it is very expensive to recover the system after fault, which motivates the work on predictability problem, i.e., the system ability to predict with certainty future faults when this system is still in a normal state.

Predictability is an important system property that determines at design stage whether the considered fault can be correctly predicted before its occurrence based on available observations. If a fault is predicted, the system operator can be warned and may decide to halt the system or otherwise take preventive measures. However, up to now, very few works have been done on this subject for discrete event systems (DESs). The authors of [6] proposed a deterministic diagnoser approach with exponential complexity as well as a polynomial method

that checks predictability directly on a twin plant. Both of them were established in a centralized way and are difficult to be extended for distributed systems due to combinatorial explosion. The first distributed method handling this problem was proposed in [21], which however has the same search space as the centralized one in the worst case.

In this paper, we propose a new efficient algorithm of predictability for DESs. First, we propose and then prove a sufficient and necessary condition for predictability, i.e., characterizing pairs of behaviors violating predictability as two trajectories, exactly one containing the fault, with the same observations before the fault and the normal one being infinite. Totally different from the polynomial method proposed in [6] that reused twin plant, we construct another structure that captures all pairs of trajectories with the same observations only before the fault while preserving the normal trajectories, where the existence of violating pairs can be directly checked. More importantly, we show how to extend this method in a distributed framework with smaller state space even in the worst case. Our distributed algorithm is different from that proposed for checking diagnosability described in [13]. For diagnosability, it suffices to synchronize local twin plants based on communication events in a unique way since the same observations are imposed both before and after the fault. While for predictability, we have to check the same observations only before the fault as well as the infinity of the corresponding normal trajectory, both in an incremental way.

The organization of the rest of the paper is as follows. The next section recalls the definitions and gives a sufficient and necessary condition for predictability. Section 3 proposes a new predictability algorithm before extending it to a distributed framework in Section 4. Section 5 gives some experimental results. Then we conclude in Section 7 after a discussion in Section 6.

2 Preliminaries

In this section, we show how to model a DES, recall the definition of its predictability, and propose a sufficient and necessary condition with a formal proof.

2.1 Models of DESs

We model a DES as a Finite State Machine (FSM), denoted by $G = (Q, \Sigma, \delta, q^0)$, where Q is the finite set of states, Σ is the finite set of events, $\delta \subseteq Q \times \Sigma \times Q$ is the set of transitions (the same notation will be kept for its natural extension to words of Σ^*), and q^0 is the initial state. The set of events Σ is divided into three disjoint parts: $\Sigma = \Sigma_o \uplus \Sigma_u \uplus \Sigma_f$, where Σ_o is the set of observable events, Σ_u the set of unobservable normal events and Σ_f the set of unobservable fault events.

Example 1. The left part of Figure 1 shows an example of a system model G, where $\Sigma_o = \{O1, O2, O3\}, \Sigma_u = \{U1, C1, C2\}$, and $\Sigma_f = \{F\}$.

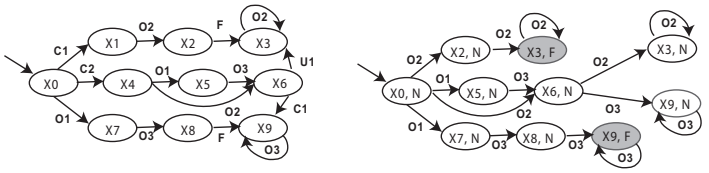

Fig. 1. A system example (left) and its diagnoser (right).

Given a system model G, its prefix-closed language $L(G)$, which describes both normal and faulty behaviors of the system, is the set of words produced by G, $L(G) = \{s \in \Sigma^* | \exists q \in Q, (q^0, s, q) \in \delta\}$. In the following, we call a word from $L(G)$ a trajectory in the system G and a sequence $q_0\sigma_0q_1\sigma_1...$ a path in G, where $\sigma_0\sigma_1...$ is a trajectory in G and we have $\forall i, (q_i, \sigma_i, q_{i+1}) \in \delta$. Given $s \in L(G)$, we denote its set of strict prefixes as $\overline{s}$, i.e., $s \notin \overline{s}$, and the post-language of $L(G)$ after s by $L(G)/s$, formally defined as: $L(G)/s = \{t \in \Sigma^* | s.t \in L(G)\}$. The projection of the trajectory s to observable events of G (resp. G_i in distributed system) is denoted by $P(s)$ (resp. $P_i(s)$). Traditionally, we assume that the system language is always live (any trajectory has a continuation, i.e. is a strict prefix of another trajectory) without unobservable cycle. Precisely, we have at least one transition from any state and every cycle in the system contains at least one observable event. This makes it feasible to check the infiniteness of a trajectory. Given two FSMs G_1 and G_2, their synchronization with respect to the set of synchronized events $\Sigma_s \subseteq \Sigma_1 \cap \Sigma_2$ [4] consists in synchronizing only the events in Σ_s, denoted by $G_1 \parallel_{\Sigma_s} G_2$. All events not in Σ_s can occur independently whenever possible. It is easy to generalize the synchronization for a set of FSMs using its associativity properties [2]. We will need also some infinite objects. So, we denote by Σ^ω the set of infinite words on Σ and by $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$ the set of words on Σ, finite or infinite. We define in an obvious way $L^\omega(G)$ and $L^\infty(G)$ and thus infinite trajectories and infinite paths.

2.2 Predictability of DESs

Predictability is considered as a crucial property of a DES in the sense that a predictable fault can possibly be avoided. Similar to diagnosability, the predictability algorithm that we will propose has exponential complexity with the number of fault types. For the sake of reducing complexity and simplicity, only one fault type at a time is considered but multiple occurrences of faults are allowed, and the other types of faults are processed as unobservable normal events. However, this framework can be extended in a straightforward way such that a number of different faults can be considered simultaneously. Now we rephrase the predictability definition [6], where a trajectory ending with a first occurrence

[4] To avoid heavy notations, we will use sometimes $\Sigma_s \nsubseteq \Sigma_1 \cap \Sigma_2$. Synchronization set has then to be understood as $\Sigma_s \cap \Sigma_1 \cap \Sigma_2$.

of the fault F (enough to cover the case with several occurrences of F) is denoted by s^F and the set of natural numbers by $\mathbb{N}$.

Definition 1 *(Predictability). A fault F is predictable in a DES G, iff*

$$\exists k \in \mathbb{N}, \forall s^F \in L(G), \exists \eta \in \overline{s^F}, \forall p \in L(G), \forall p' \in L(G)/p,$$
$$[(P(p) = P(\eta)) \wedge (F \notin p) \wedge (|p'| \geq k) \Rightarrow (F \in p')].$$

A fault F is predictable iff for any trajectory s^F ending with a first occurrence of F, there exists at least one strict prefix of s^F, denoted by η (thus η does not contain F) such that for each normal trajectory p with the same observations as η, all the long enough (depending only on F) continuations of p should contain F. Only in this way, F can be certainly predicted before its occurrence.

2.3 Sufficient and Necessary Condition

Suppose now that we have two trajectories in a given system such that exactly one, denoted by s^F, ends with the fault F, the other without F has at least one prefix with the same observations as the maximum strict prefix of s^F and is infinite. With such two trajectories, whatever we observe before the occurrence of F, we cannot tell whether F will occur or not since both are possible while only one will contain F in the future. Now we formally define such a pair of trajectories.

Definition 2 *(Pre-Violating Pair (PVP)). Given a system G and a fault F to be predicted, a pair of trajectories $s^F, p.p' \in L^\infty(G)$ is called a Pre-Violating Pair (PVP) with respect to F if the three conditions are satisfied: 1) $P(s^F) = P(p)$; 2) $F \notin p.p'$; 3) p' is infinite.*

Here is the sufficient and necessary condition of predictability.

Theorem 1. *A fault F is predictable in a system G iff there is no PVP in G with respect to F.*

Proof. $\Rightarrow$ Suppose that we have a PVP in G, i.e., s^F and $p.p'$ as in Definition 2. Hence, we have $P(s^F) = P(p)$, i.e., the maximum normal prefix of s^F has the same observations as p since F is not observable. It follows that $\forall \eta \in \overline{s^F}$, $\exists \eta' \in \overline{p} \cup \{p\}$ such that $P(\eta) = P(\eta')$. Furthermore, η' has at least one normal infinite continuation since $F \notin p.p'$ and p' is infinite. This violates Definition 1, i.e., F is not predictable.
$\Leftarrow$ Now suppose that F is not predictable. It follows that Definition 1 is violated, which can be expressed by the following: $\forall k \in \mathbb{N}$, $\exists s^F \in L(G)$, $\forall \eta \in \overline{s^F}$, $\exists p \in L(G)$, $F \notin p$, $P(p) = P(\eta)$, $\exists p' \in L(G)/p, |p'| \geq k$, $F \notin p'$. Let η as the maximum normal prefix of s^F. The above formula implies (by taking k greater than $|Q|$) that there must exist a normal infinite trajectory $p.p'$, i.e., $F \notin p.p'$, such that $P(p) = P(\eta)$. This means $P(p) = P(s^F)$ since η is the maximum normal prefix of s^F. Hence, $p.p'$ and s^F constitute exactly a PVP. ■

3 Centralized Framework

Since the predictability verification of a given fault F is to check the existence of PVP, from Definition 2, we take three steps: 1) obtain the set of maximum strict prefixes for all s^F, and actually we can restrict to those s^F which are minimal for the order induced by the prefix relation (with F excluded), which is enough from Theorem 1 as a PVP for $w.w'.F$ is a PVP for w.F; 2) obtain the set of infinite normal trajectories; 3) compare the above two sets in terms of observations to check the existence of PVP. We will construct one FSM for each step. Before this, given a system model, we first construct its non-deterministic diagnoser to explicitly show fault information, which is different from the deterministic diagnoser proposed in [17].

Definition 3 *(Diagnoser). Given a system G, its diagnoser with respect to a considered fault F is a nondeterministic FSM $D = (Q_D, \Sigma_D, \delta_D, q_D^0)$, where 1) $Q_D \subseteq Q \times \{N, F\}$ is the set of states; 2) $\Sigma_D = \Sigma_o$ is the set of events; 3) $\delta_D \subseteq Q_D \times \Sigma_D \times Q_D$ is the set of transitions; 4) $q_D^0 = (q^0, N)$ is the initial state. The transitions of δ_D are those $((q, \ell), e, (q', \ell'))$, with (q, ℓ) reachable from the initial state q_D^0, such that there is a transition path $p = (q \xrightarrow{u_1} q_1 ... \xrightarrow{u_m} q_m \xrightarrow{e} q')$ in G, with $u_k \notin \Sigma_o, \forall k \in \{1, ..., m\}$, $e \in \Sigma_o$ and $\ell' = F$, if $\ell = F \vee F \in \{u_1, ..., u_k\}$, and otherwise, $\ell' = N$.*

The diagnoser preserves all observable information. Then we append the fault label F to those states, up to which the fault has already occurred, and normal label N to those without the fault occurrence. The right part of Figure 1 depicts the diagnoser of the system in Example 1, where gray nodes represent the states where F has effectively occurred. Based on such a diagnoser, we then construct two different FSMs to capture the set of maximum normal prefixes of all minimal faulty trajectories and the set of normal ones, respectively.

Definition 4 *(Fault-Prefix Diagnoser). Given a diagnoser D, the fault-prefix diagnoser D_{FP} is constructed as follows:*

- *Keep only the minimal paths containing the fault label;*
- *$\forall((q, l), e, (q', l')) \in \delta_D$, $l = N, l' = F$, it is transformed into $((q, l), \Sigma_o, (q, l)) \in \delta_D$, i.e., (q, l) goes back to itself with any observable event. Such a state (q, l) is called an absorbing state in the following.*

Recall that predictability analysis consists in first checking whether the maximum normal prefix of a faulty trajectory has the same observations with a normal one, and then examining whether the normal one is infinite, which is represented by a cycle in a FSM. This is why in a fault-prefix diagnoser, we keep the exact observable events before the fault and then add to an absorbing state a self-cycle with all observable events. The idea is to make it able to synchronize with a normal trajectory to check whether the latter has a cycle in the future.

Definition 5 *(Normal Diagnoser). Given a diagnoser D, the normal diagnoser D_N is obtained by retaining only normal states with their associated transitions.*

To check the existence of PVP, we synchronize the fault-prefix diagnoser and the normal diagnoser based on the set of observable events. This synchronization is actually the intersection of the maximum normal prefixes of minimal faulty trajectories and the normal trajectories in terms of observations.

Definition 6 *(Pre-Verifier). Given a system, its pre-verifier PV is constructed by synchronizing its fault-prefix diagnoser D_{FP} and its normal diagnoser D_N based on observable events, i.e., $PV = D_{FP} \parallel_{\Sigma_o} D_N$.*

A state of a pre-verifier s^v is composed of a state of the fault-prefix diagnoser and a state of the normal diagnoser, denoted by $s^v = (q^f, q^n)$. All states in D_{FP}, D_N and thus PV having by construction a normal label N, it will be skipped in the following. If q^f is an absorbing state, then s^v is also called an absorbing state. In a pre-verifier, a path containing a cycle made up of absorbing states is called a violating path. Note that pre-verifier proposed here greatly reduces the state space compared to twin plant used both in [6] and [13]. The latter is constructed directly by synchronizing the whole diagnoser defined in Definition 3 with itself. Clearly, both fault-prefix diagnoser D_{FP} and normal one D_N are smaller than the diagnoser D. Thus, the pre-verifier $D_{FP} \parallel_{\Sigma_o} D_N$ is much smaller than $D \parallel_{\Sigma_o} D$, even in the worst case.

Lemma 1. *A path in PV is a violating one iff it corresponds to a PVP with minimal s^F in the corresponding system.*

Proof. $\Rightarrow$ Let ρ a violating path in PV. Thus $\rho = s_0^v\sigma_0 \dots s_i^v\sigma_i \dots s_j^v\sigma_j \dots s_k^v\sigma_k \dots$ where $0 \leq i \leq j$, $j < k$, $s_j^v = s_k^v$ and $\forall l < i$ s_l^v is not an absorbing state, $\forall l \geq i$ s_l^v is an absorbing state. By construction of D_{FP}, $\sigma_0 \dots \sigma_{i-1}$ comes from minimal s^F where $P(s^F) = \sigma_0 \dots \sigma_{i-1}$. By construction of D_N, $\sigma_0 \dots \sigma_k$ comes from $p.p' \in L^\omega(G)$ where $P(p) = \sigma_0 \dots \sigma_{i-1}$, $F \notin p.p'$ and p' is infinite (with $P(p') = \sigma_i \dots (\sigma_j \dots \sigma_{k-1})^\omega$). Thus $s^F, p.p'$ is a PVP in G.
$\Leftarrow$ Let $s^F, p.p'$ a PVP in G with s^F minimal. s^F gives birth in D_{FP} to $q_0^f\sigma_0 \dots \sigma_{i-1}q_i^f$, $0 \leq i$, where q_i^f is an absorbing state. $p.p'$ gives birth in D_N to $q_0^n\sigma_0 \dots \sigma_{i-1}q_i^n \dots q_k^n\sigma_k \dots$ where $i < k$ and $\exists j, j < k, q_j^n = q_k^n$. Then $\rho = (q_0^f, q_0^n)\sigma_0 \dots \sigma_{i-1}(q_i^f, q_i^n) \dots (q_i^f, q_k^n)\sigma_k \dots$ is a violating path in PV. ■

Figure 2 shows the two diagnosers and a part of PV for G in Example 1. In PV, a state is composed of a fault-prefix diagnoser state (top) and a normal diagnoser state (bottom). The absorbing states in the fault-prefix diagnoser (X2 and X8) and in PV (all states whose top part is X2 or X8) are bold circles. A violating path in PV, i.e., with an absorbing state cycle, corresponds to a PVP. For example, the path whose trajectory is $O2.O2.O2^\omega$ is a violating path. Its corresponding trajectories in G are $C1.O2.F$ and $C2.O2.U1.O2^\omega$, which are exactly a PVP with $s^F = C1.O2.F$, $p = C2.O2$ and $p' = U1.O2^\omega$. So F is not predictable in G.

The following theorem is from Theorem 1 and Lemma 1.

Theorem 2. *A fault F is predictable in a system G iff there is no violating path in the corresponding PV.*

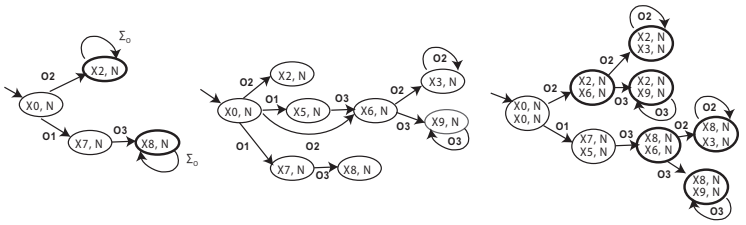

Fig. 2. Fault-prefix diagnoser (left), normal diagnoser (middle) and part of PV (right) of G in Example 1.

4 Distributed Framework

In the previous section, we have presented a centralized approach of predictability analysis. However, it is not realistic to construct a global model for a complex system due to combinatorial explosion. In this section, we will show how to extend our centralized method to a distributed framework.

4.1 Distributed Model

We consider a distributed DES G composed of a set of components $G_1, ..., G_n$ that communicate with each other by communication events. Similar to the system model in the centralized approach, each component is modeled by a FSM, denoted by $G_i = (Q_i, \Sigma_i, \delta_i, q_i^0)$. Differently, the set of events Σ_i is divided into four disjoint parts instead of three: $\Sigma_i = \Sigma_{i_o} \uplus \Sigma_{i_u} \uplus \Sigma_{i_f} \uplus \Sigma_{i_c}$, where Σ_{i_c} is the set of unobservable correct communication events. For any pair of distinct local components G_i and G_j, we have $\Sigma_{i_o} \cap \Sigma_{j_o} = \emptyset, \Sigma_{i_u} \cap \Sigma_{j_u} = \emptyset$, and $\Sigma_{i_f} \cap \Sigma_{j_f} = \emptyset$. In other words, the only shared events between different components are communication ones. Thus, given a considered fault F, it can only occur in one component, denoted by G_F (called the faulty component, the others being the normal ones). Similarly, we assume that the language for each component is always live without unobservable cycle.

Example 2. A distributed system G' is composed of two components G_1 and G_2, where the system in Example 1 is now considered as G_1 with the difference $\Sigma_{1_u} = \{U1\}$ and $\Sigma_{1_c} = \{C1, C2\}$, and G_2 is shown in the left part of Figure 3 with $\Sigma_{2_o} = \{O4, O5, O6\}$ and $\Sigma_{2_c} = \{C1, C2\}$.

Given a distributed DES, to apply the centralized predictability algorithm, we have first to synchronize all components based on communication events to obtain the global model. The global pre-verifier is calculated by synchronizing the fault-prefix diagnoser with the normal diagnoser, both built from the global model, based on observable events before the fault. The PVP is then checked directly on this global pre-verifier. To save search space but with the same result, the idea of our distributed algorithm is to first construct local structures (e.g.,

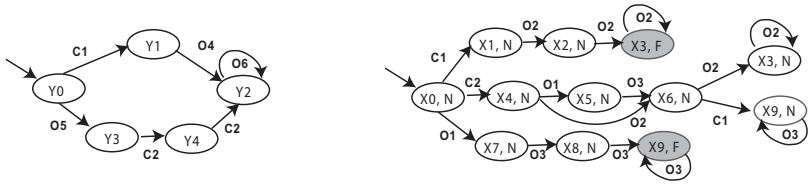

Fig. 3. Component G_2 (left) and the diagnoser of G_1 (right) for the system G' in Example 2.

local pre-verifier) by synchronization on local observable events before the fault to search for local version of PVP. The correspondence between this local version and global PVP is then checked by the synchronization based on communication events, which is done partially and incrementally. We will provide a formal proof for the equivalence between centralized and distributed approach.

4.2 Local Analysis

In a distributed DES, since the fault to be predicted can only occur in G_F, we should obtain the original local predictability information from G_F before determining the global decision. For this, we first define the local version of PVP, the projection of s on the local observations in G_F denoted by $P_F(s)$. Different from PVP, local PVP does not require an infinite trajectory, which will be checked from global point of view.

Definition 7 *(Local PVP). In the component G_F, a pair of local trajectories $s^F, p.p' \in L^\infty(G_F)$ is called a local PVP if $F \notin p.p'$ and $P_F(s^F) = P_F(p)$.*

Lemma 2. *Given a distributed DES G, the projection of a PVP on G_F is a local PVP. But conversely, it is not true that all local PVPs can be extended into (global) PVPs.*

Proof. $\Rightarrow$ Suppose that two global trajectories, denoted by s^F and $p.p'$ are a PVP. We show that the projections of s^F and $p.p'$ on G_F, denoted by $\mathbb{P}_F(s^F)$ and $\mathbb{P}_F(p.p')$, are a local PVP. Since F can only occur in G_F, We must have $\mathbb{P}_F(s^F) = s'^F$, i.e., the projection of s^F on G_F should also be a local trajectory ending with F. From $F \notin p.p'$, we have $F \notin \mathbb{P}_F(p.p')$. Furthermore, $\Sigma_{i_o} \cap \Sigma_{j_o} = \emptyset$ for $i \neq j$ implies that $P(s^F) = P(p) \Rightarrow P_F(\mathbb{P}_F(s^F)) = P_F(\mathbb{P}_F(p))$. From Definition 7, $\mathbb{P}_F(s^F)$ and $\mathbb{P}_F(p.p')$ constitute a local PVP.
$\nLeftarrow$ Now consider the two local trajectories $p1 = C1.O2.F$ and $p2 = C2.O2.U1$ in G_1 that constitute a local PVP and show that they are not extendible into a PVP. The reason is that when synchronizing G_1 and G_2, for $p2$, $O5$ occurs necessarily before $O2$ to synchronize on $C2$ while, for $p1$, we have $O2$ without $O5$ before F. Thus, $O5$ distinguishes the normal trajectory from its corresponding faulty one before F. Hence, $p1$ and $p2$ cannot be extended into a global PVP. ■

From Lemma 2, we know that a local PVP may or may not correspond to a PVP. To verify predictability, it is necessary to check whether a local PVP can be effectively extended into a global PVP. To obtain the set of local PVPs, we construct the local diagnoser exactly as in Definition 3 except that here the set of events retained are not only observable events but also communication events, the latter used to check the extendibility of a local PVP into a PVP in the following step. The right part of Figure 3 shows the diagnoser of G_1 in Example 2. Then the local fault-prefix diagnoser D_{FP_F} and local normal diagnoser D_{N_F} are constructed in the same way as Definition 4 and Definition 5 with the only difference that their construction is based on the local diagnoser. To obtain the local pre-verifier for G_F, denoted by PV_F , we distinguish the unobservable communication events in D_{FP_F}, prefixed by F, and those in D_{N_F}, prefixed by N, such that PV_F is built by synchronizing D_{FP_F} and D_{N_F} based on the set of observable events. From now on, we call a path of a local PV containing at least one absorbing state a local violating path. The left part of Figure 4 depicts a part of the local PV for G_1, where absorbing states are represented by bold circles. All paths of length at least 3 shown here are local violating paths.

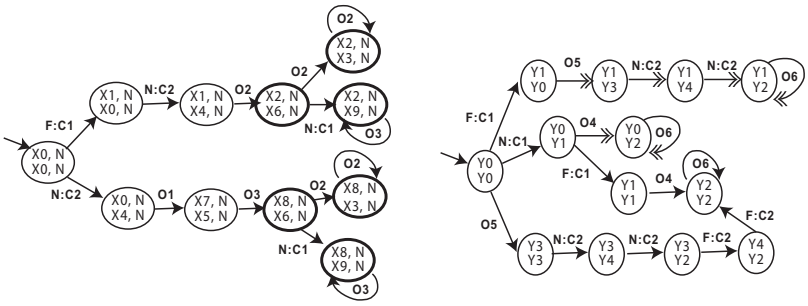

Fig. 4. Part of local PV of G_1 (left) and part of normal verifier of G_2 (right) for the system G' in Example 2.

The proof of Lemma 3 is similar to that of Lemma 1.

Lemma 3. *A path in PV_F is a local violating one iff it corresponds to a local PVP with minimal s^F in G_F.*

4.3 Global Checking

Since a local violating path corresponds to a local PVP, the predictability checking for a distributed system is to check whether a local violating path in PV_F corresponds to a PVP.

Definition 8 *(Global Extendibility of Local Violating Paths). Given a system G, let p_1 and p_2 be the corresponding local trajectories in G_F for a local violating path ρ in the PV_F. ρ is globally extendible if the following two conditions are satisfied: 1) $\exists p'_1, p'_2 \in L^{\infty}(G)$, such that $\mathbb{P}_F(p'_1) = p_1$ and $\mathbb{P}_F(p'_2) = p_2$; 2) p'_1 and p'_2 constitute a PVP.*

Now the following theorem can be proved from Theorem 1, Lemma 2, Lemma 3 and Definition 8.

Theorem 3. *A fault F is predictable in a system G iff there is no local violating path in PV_F that is globally extendible.*

Checking the extendibility of a local violating path means to check firstly whether the observations in other components can distinguish the corresponding two trajectories before fault occurrence and secondly whether the normal trajectory can be extended into an infinite one. For this, given a normal component, we construct the following structure.

Definition 9 *(Normal Verifier). Given a normal component G_i, its normal verifier NV_i is constructed as follows:*

1. *Construct a coarser model G'_i based on G_i by keeping only the set of communication and observable events;*
2. *Construct two instances $G_i'^F$ and $G_i'^N$ of G'_i by prefixing the communication events by F and N, respectively, and then synchronize them based on the observable events: $NV_i = G_i'^F \parallel_{\Sigma_{i_o}} G_i'^N$;*
3. *Retrieve the lost parts of $G_i'^N$ in NV_i that are blocked by different observable events during step 3, called Normal Unique parts, shortly NU parts. i.e., add to NV_i all transitions $(q^f, q_0^n) \xrightarrow{e_1} (q^f, q_1^n) \dots \xrightarrow{e_k} (q^f, q_k^n)$ such that: $\forall j, 1 \leq j \leq k, q_{j-1}^n \xrightarrow{e_j} q_j^n$ is a transition in $G_i'^N$, $e_1 \in \Sigma_{i_o}$, $\exists\ q^f \xrightarrow{o} q'^f$ transition in $G_i'^F$ with $o \in \Sigma_{i_o}$ and $\forall\ q^f \xrightarrow{o} q'^f$ transition in $G_i'^F$ with $o \in \Sigma_{i_o}$, $e_1 \neq o$.*

A normal verifier has two characteristics: 1) obtain all pairs of observationally equivalent trajectories (non NU parts); 2) recover all parts of trajectories in $G_i'^N$ that are blocked by different observable events (NU parts). The first one is to check the same observations before the fault while the second is to examine whether the normal trajectory is infinite. The right part of Figure 4 shows a part of the normal verifier for G_2, where the transitions with double arrows are the NU parts.

Given the local pre-verifier PV_F and a normal verifier NV_i, to check the extendibility of local violating paths in the subsystem composed of G_F and G_i, we take two steps: 1) synchronizing PV_F and the non NU parts of NV_i based on communication events (with the same prefix F or N) until arriving in an absorbing state; 2) from an absorbing state, synchronizing based on communication events only with the prefix N. Intuitively, the first step checks whether the corresponding two trajectories in this subsystem have the same observations before the fault. In NV_i, only the non NU parts have the same observations for

both trajectories. Hence, before achieving absorbing states, only the non NU parts should synchronize with PV_F to guarantee the same observations before the fault. The second step extends only the normal trajectory to check its infinity since synchronized events now become the common communication events with the prefix N. If we synchronize PV_F with all NV_i with the two steps without any reduction, then what we get is isomorphic (same set of paths from origin, and thus same language) to the global pre-verifier obtained by the centralized approach.

Theorem 4. *Given a distributed system $G = (G_1, ..., G_n)$, the final synchronized product of PV_F with NV_i for all normal components is isomorphic to the global pre-verifier.*

Proof. **(sketch)** *Base Case*: We first show this equivalence for n=2, i.e., two components with $G_F = G_1$. Given $G = (G_1, G_2)$, the global pre-verifier is obtained by $PV_2^c = [G_1 \parallel_{\Sigma_c} G_2]^{FP+P_F} \parallel_{\Sigma_o} [G_1 \parallel_{\Sigma_c} G_2]^{N+P_N}$, where $FP + P_F$ (resp. $N + P_N$) means constructing fault-prefix diagnoser (resp. normal diagnoser) and adding prefix of F (resp. N) for communication events. This can be transformed into $PV_2^c = [D^F_{1_{FP}} \parallel_{\Sigma_{F:c}} G'^F_2]^\star \parallel_{\Sigma_o} [D^N_{1_N} \parallel_{\Sigma_{N:c}} G'^N_2]$, where $\star$ represents absorbing state calculation. With our distributed algorithm, the final FSM obtained is $PV_2^d = [D^F_{1_{FP}} \parallel_{\Sigma_{1_o}} D^N_{1_N}] \parallel_{\Sigma_{F:c\star,N:c}} [G'^F_2 \parallel_{\Sigma_{2_o}} G'^N_2]^{\star\star}$, where $\Sigma_{F:c\star,N:c}$ signifies the synchronized events are F communications events before absorbing states and all N communication events (in particular those in NU parts after absorbing states) and $\star\star$ means the recuperation of NU parts, i.e., 3rd step in Definition 9. From $[D^F_{1_{FP}} \parallel_{\Sigma_{F:c}} G'^F_2]^\star$, in PV_2^c, we only keep F communication events before absorbing states while retaining the rest of normal trajectory. This is transformed in PV_2^d through the set of synchronized events $\Sigma_{F:c\star,N:c}$ with $\star\star$ operation. Hence, we have $PV_2^c \simeq PV_2^d$.
Inductive Case: Suppose $PV_m^c \simeq PV_m^d$, we now show $PV_{m+1}^c \simeq PV_{m+1}^d$. From above, we have the following demonstration, where the synchronized events are omitted that are similar to the base case: $PV^c_{m+1} = [D^F_{1_{FP}} \parallel ... \parallel G'^F_{m+1}]^\star \parallel [D^N_{1_N} \parallel ... \parallel G'^N_{m+1}] = [[D^F_{1_{FP}} \parallel ... \parallel G'^F_m]^\star \parallel G'^F_{m+1}]^\star \parallel [[D^N_{1_N} \parallel ... \parallel G'^N_m] \parallel G'^N_{m+1}] \simeq PV^c_m \parallel [G'^F_{m+1} \parallel G'^N_{m+1}]^{\star\star} \simeq PV^d_m \parallel [G'^F_{m+1} \parallel G'^N_{m+1}]^{\star\star} = PV^d_{m+1}$. ■

We have proved the equivalence between the centralized result and the distributed one. However, to save search space, in distributed framework, we can partially and incrementally synchronize all local violating paths in PV_F with NV_i for connected components and we have the following theorem.

Theorem 5. *Given a system G, let Θ be the set of connected components containing G_F. After incrementally checking the extendibility of local violating paths in PV_F with NV_i of all normal components in Θ, a local violating path is globally extendible iff there exists a path p in the final FSM satisfying one of the following conditions: 1) p has an absorbing state cycle; 2) p has an absorbing state and $\Theta \neq G$.*

The global checking of predictability is much more complex than that of diagnosability [13]. For the latter, it is enough to construct local twin plants for

all components before synchronizing them in a unique way since the violating pair has the same observations in the whole way. While for the former, as shown in this paper, we have to construct different structures for normal and faulty components with different ways of synchronization before and after fault.

Consider the part of PV_F and of NV_2 shown in Figure 4. The local violating paths whose trajectories are $t1 = F{:}C1.N{:}C2.O2.O2.O2^*$ (resp. $t2 = F{:}C1.N{:}C2.O2.N{:}C1.O3^*$) are not globally extendible because $N{:}C2$ is blocked during extendibility checking with NV_2. The reason is explained in the proof of Lemma 2: observations before F are not the same after synchronization. For the local violating path whose trajectory is $t3 = N{:}C2.O1.O3.N{:}C1.O3^*$, after checking $t3$ with NV_2, the normal trajectory is blocked by $N{:}C1$ after arriving in absorbing states. This means that the trajectories of the corresponding pair have the same observations before F but the normal one cannot be infinite. Thus, $t3$ is not globally extendible. Consider finally the trajectory $t4 = N{:}C2.O1.O3.O2.O2^*$. Its extendibility checking with NV_2 achieves absorbing states and makes the normal trajectory infinite with a cycle, i.e., an absorbing state cycle. Precisely, $O5.C2.O1.O3.U1.O2^\omega$ is an infinite normal trajectory in G' of Example 2, which has the same observable prefix $O5.O1.O3$ with the faulty trajectory $O5.O1.O3.F$. It follows that $t4$ is globally extendible, i.e., F is not predictable in G'.

4.4 Algorithm

Now we formally describe our distributed predictability algorithm based on Theorem 5, which is shown by Algorithm 1. Given the input (line 1) as the set of component models and the fault F in G_F, which is used as initialization of the current subsystem G_S (line 2), we first construct PV_F (see Section 4.2 for more details) and reduce it to only retain local violating paths, i.e., with at least one absorbing state (lines 3-4). If the reduced PV_F is not empty and there exists at least one connected component to G_S, i.e., with at least one common communication event (line 5), meaning that the retained PV_F should be further checked in an extended subsystem, then we repeatedly perform the following steps: 1) Select one component G_i not in G_S but connected to it before constructing its normal verifier NV_i as described in Definition 9 (lines 6-7); 2) Check the extendibility of PV_F with NV_i as described in the previous section (line 8); 3) Reduce the newly obtained PV_F to keep only paths containing at least one absorbing state before updating the subsystem G_S by adding G_i (lines 9-10). In the final resulting FSM, if there exists at least one globally extendible violating path (line 11), F is not predictable and PV_F is provided (line 12). Otherwise, F is predictable and predictable information is returned (lines 13-14).

5 Experimental Results

We have proved the correctness of our algorithms from theoretical point of view. To show their efficiency from practical point of view, we have implemented and compared our centralized algorithm with twin plant method in [6] as well as

Algorithm 1 Predictability Algorithm for Distributed DES

INPUT: component models $G_1, ..., G_n$ in G; F in G_F
Initializations: $G_S \leftarrow G_F$(current subsystem)
$PV_F \leftarrow ConstructPV_F(G_F)$
$PV_F \leftarrow Reduce(PV_F)$
while $PV_F \neq \emptyset$ and $ConnectComp(G_S, G) \neq \emptyset$ **do**
  $G_i \leftarrow Select(ConnectComp(G_S, G))$
  $NV_i \leftarrow ConstructNV_i(G_i)$
  $PV_F \leftarrow CheckExtendibility(PV_F, NV_i)$
  $PV_F \leftarrow Reduce(PV_F)$
  $G_S \leftarrow ADD(G_S, G_i)$
if $\exists\rho(\rho \in PV_F \wedge GlobExtViolating(\rho))$ **then**
  return PV_F
else
  return "the fault is predictable in G"

our distributed one with that described in [21] (for this comparison we also implemented algorithms described in [6, 21] as codes were not available from the authors). All our experimental results are obtained by running our java program on a Mac OS laptop running on a 1.7 GHz Intel Core i7 processor with 8 Go 1600 MHz DDR3 of memory.

	Centralized		Distributed			
	S/T (TP) vs. S/T(PV)	T (ms)	T(LPV) [21] vs. O	T(NV) [21] vs. O	T(FP) [21] vs. O	T(ms)
Ex G	36/62 vs. 16/21	26 vs. 21	—			
G_1[6]	9/10 vs. 4/3	16 vs. 9	—			
G_2[6]	10/12 vs. 3/4	17 vs. 10	—			
Ex1 [21]	23/27 vs. 7/7	15 vs. 12	—			
h-c c1	300/566 vs. 21/16	51 vs. 23	—			
Ex G'	—		69 vs. 21	43 vs. 25	86 vs. 28	186 vs. 43
Ex2 [21]	—		68 vs. 21	29 vs. 16	106 vs. 33	81 vs. 33
h-c d1	—		51 vs. 22	204 vs. 20	836 vs. 51	8s vs. 30
h-c d2	—		226 vs. 16	204 vs. 20	536 vs. 15	6mn vs. 33
h-c d3	—		116 vs. 21	254 vs. 24	1344 vs. 42	1mn vs. 36

Table 1. Experimental comparison results for centralized and distributed algorithms

Table 1 shows part of our experimental results, where final verdict results, i.e., whether the system is predictable or not, of all examples are omitted, which are exactly the same for all algorithms. For centralized comparison, we give the number of states/transitions of twin plant (S/T(TP)) for algorithm in [6] and that of our pre-verifier (S/T(PV)), for the examples G in this paper, G_1, G_2 in [6], Ex1 in [21] and one hand-craft (h-c c1) example. And for distributed comparison, we have the number of transitions in local pre-verifier(T(LPV)), normal verifier (T(NV)) as well as in the final synchronized product (T(FP)),

both for algorithm proposed in [21] and our distributed one (O). The examples that we chose to show here include G' in this paper and the distributed one Ex2 in [21] with some h-c examples to show the scalability. For the sake of simplicity, we use the name of structures defined in this paper to compare the different local structures with the same goal in both algorithms to show the state space that can be saved by our algorithm. To compare running time for all these algorithms, we use millisecond (ms) as time unit by default and s for second and mn for minute.

Our experimental results show that our algorithms can save at least 50% space for most of our examples. Note that in this figure, we only give the results for systems with two components due to space limit. Actually our experimental results with more components show that more components we have, more space can be saved by our algorithm. Another important observation is that the state space saved by our algorithm also depends on two other factors. One is the percentage of observable events, less this percentage is, more space is saved. For example, h-c c1 has the same structure in terms of observable events as Ex G in this paper but with more unobservable events. We can save state space much more for h-c c1 compared to Ex G. Another one is the position of the fault, earlier the fault occurs, more space is saved. For example, in h-c d1, the fault occurs almost at the latest step while in h-c d2, the fault occurs at the very beginning. The faulty components have the same structure. The results show that our algorithm can save more space in the case of early occurrence of the fault, which is reasonable considering that we introduce absorbing states to not only guarantee the same observations before the fault but also avoid keeping all the events after the fault. The time saved is even more dramatical in the extreme case h-c d2 in terms of the fault position, 6 minutes vs. 33 milliseconds. Note that the state space saved for big examples (e.g., hand-craft ones) is more clear than for smaller ones (e.g., ones found in the literature).

6 Discussion

In [6], the authors analyzed predictability directly on a global twin plant. They captured the ambiguous behaviors violating predictability in different paths of the twin plant, which is not suitable for a distributed framework. While we propose a different structure where a PVP violating predictability is caught by only one path, which facilitates the distributed extension by enabling the extensibility checking of local violating paths. To verify diagnosability in a distributed way, it is sufficient to construct a local twin plant for each component and then synchronize them to make sure that the two corresponding trajectories of each path in the final product have the same global occurrence of the observations; this is the distributed algorithm proposed in [13]. While to verify predictability, we have to construct a structure where each path captures a pair of trajectories with the same observations only before the fault but where the normal trajectory cannot be blocked after the fault, which is quite different from diagnosability and much more difficult, especially in the distributed case. Another close work is the distributed algorithm in [21], where the condition violating predictability

is not formally proved. Moreover, the authors chose to exploit all states after the fault, which is not necessary since predictability concerns the same observations only before the fault. It follows that, in the worst case, the state space could be the same as the centralized approach. While we construct two different diagnosers with absorbing states, which greatly reduce the search space even in the worst case but always with the correct result. As in the previous approaches proposed in the above papers, our predictability checking is also polynomial in the number of system states. But it can practically be much more applicable for large complex systems considering the reduced space. This is confirmed by our experimental results shown in the previous section.

7 Conclusion

In this paper, we propose a new approach for predictability analysis both in a centralized and a distributed framework. First, we formally characterize pairs of trajectories violating predictability. Then, we show how to check the existence of such pairs in a centralized way before adapting it for a distributed framework. Finally, we provide some experimental results to support the efficiency of our algorithms. One perspective of this work is to adapt our approach to deal with fault patterns [11], which is more general in the diagnosis domain. In the literature, only a centralized framework is proposed in [10], which is not extendable to a distributed one since predictability violation is also checked directly on twin plant. Another one is to extend our approach to distributed systems with asynchronous communication events, which is not yet handled in the literature.

References

1. N. Bertrand, É. Fabre, S. Haar, S. Haddad, and L. Hélouët. Active diagnosis for probabilistic systems. In Anca Muscholl, editor, *Proceedings of the 17th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'14)*, volume 8412 of *Lecture Notes in Computer Science*, pages 29–42, Grenoble, France, april 2014. Springer.
2. C. G. Cassandras and S. Lafortune. *Introduction To Discrete Event Systems, Second Edition.* Springer, 2008.
3. A. Cimatti, C. Pecheur, and R. Cavada. Formal Verification of Diagnosability via Symbolic Model Checking. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 363–369. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence, Inc., 2003.
4. L. Console, C. Picardi, and D. Theseider Dupré. A Framework for Decentralized Qualitative Model-based Diagnosis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 286–291. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence, Inc., 2007.
5. R. Debouk, R. Malik, and B. Brandin. A Modular Architecture for Diagnosis of Discrete Event Systems. In *Proceedings of the 41st IEEE Conference on Decision and Control (CDC-02)*, volume 1, pages 417–422. IEEE., 2002.
6. S. Genc and S. Lafortune. Predictability of Event Occurrences in Partially-observed Discrete-event Systems. *Automatica*, 45(2):301–311, 2009.

7. V. Germanos, S. Haar, V. Khomenko, and S. Schwoon. Diagnosability under weak fairness. In *Proceedings of the 14th International Conference on Application of Concurrency to System Design (ACSD'14)*, Tunis, Tunisia, june 2014. IEEE Computer Society Press.
8. A. Grastien, J. R. Anbulagan, and E. Kelareva. Diagnosis of Discrete-event Systems Using Satisfiability Algorithms. In *Proceedings of the 22th American National Conference on Artificial Intelligence (AAAI-07)*, pages 305–310. Menlo Park, Calif.: AAAI Press., 2007.
9. S. Haar, S. Haddad, T. Melliti, and S. Schwoon. Optimal constructions for active diagnosis. In Anil Seth and Nisheeth Vishnoi, editors, *Proceedings of the 33rd Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13)*, volume 24 of *Leibniz International Proceedings in Informatics*, pages 527–539, Guwahati, India, december 2013. Leibniz-Zentrum für Informatik.
10. 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.
11. 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*, pages 262–268, 2006.
12. S. Jiang, Z. Huang, V. Chandra, and R. Kumar. A Polynomial Time Algorithm for Testing Diagnosability of Discrete Event Systems. *Transactions on Automatic Control*, 46(8):1318–1321, 2001.
13. Y. Pencolé. Diagnosability Analysis of Distributed Discrete Event Systems. In *Proceedings of the 16th European Conference on Articifial Intelligent (ECAI04)*, pages 43–47. Nieuwe Hemweg: IOS Press., 2004.
14. Y. Pencolé and M. O. Cordier. A Formal Framework for the Decentralised Diagnosis of Large Scale Discrete Event Systems and Its Application to Telecommunication Networks. *Artificial Intelligence*, 164:121–170, 2005.
15. R. Reiter. A Theory of Diagnosis from First Principles. *Artificial Intelligence*, 32(1):57–95, 1987.
16. 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. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence, Inc., 2007.
17. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of Discrete Event System. *Transactions on Automatic Control*, 40(9):1555–1575, 1995.
18. A. Schumann and J. Huang. A Scalable Jointree Algorithm for Diagnosability. In *Proceedings of the 23rd American National Conference on Artificial Intelligence (AAAI-08)*, pages 535–540. Menlo Park, Calif.: AAAI Press., 2008.
19. P. Struss. Fundamentals of Model-based Diagnosis of Dynamic Systems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 480–485. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence, Inc., 1997.
20. L. Ye and P. Dague. Diagnosability Analysis of Discrete Event Systems with Autonomous Components. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-10)*, pages 105–110. Nieuwe Hemweg: IOS Press., 2010.
21. 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.