Conjunctive and Disjunctive Architectures for Decentralized Prognosis of Failures in Discrete Event Systems

Ahmed Khoumsi, and Hicham Chakib

Abstract—[1] has proposed an architecture for the decentralized prognosis of DES, where several local prognosers cooperate to predict failures in a discrete event system (DES). In this paper, we first present the proposition of [1] as a disjunctive architecture, and then we develop a conjunctive architecture which is dual and complementary to the disjunctive one. We also propose a mixed architecture that combines and generalizes the disjunctive and conjunctive architectures. We finally show that our work can be easily extended to predict a failure at least $k$ steps before its occurrence, for a given $k \geq 1$.

Note to Practitioners—This study is in the scope of discrete event systems (DES), whose behaviors are defined by sequences of events. DES may be representations of multi-modes continuous systems, where we are interested by the sequences of modes, and not by the continuous behavior in each mode. For example, during a flight, an aircraft goes through a sequence of flight modes, such as heading, altitude changes, and speed changes. DES may also be artificial systems whose dynamics is intrinsically defined by sequences of events. For example, a telecommunication system where “receive a message” and “send a message” are examples of events. DES are encountered in several areas, such as: avionics, air traffic, road traffic, wireless sensor networks, web services, communications protocols. In the design of DES, we are usually confronted with the risk that a DES violates required properties; such violation will be referred to as fault. In this paper, we study prognosis, which consists in predicting faults. The interest of prognosis is that we are informed of a fault or at least reduce its effects. In this paper, we study decentralized prognosis, where several so-called local prognosers cooperate to make an effective prediction. A justification of using decentralized prognosis is that concrete DES are usually decentralized by the fact that they consist of several modules. Another reason is that the decentralized approach can help increase the clarity and reduce complexity of the prognosis system. The presented study is promising, but prognosis of DES is quite new and necessitates more developments before being commonly used in industry.

Index Terms—Discrete event systems (DES), Decentralized prognosis, Disjunctive architecture, Conjunctive architecture.

I. INTRODUCTION

Prognosis consists in predicting failures before their occurrence. In the context of discrete event systems (DES), a plant modeled as a DES is observed with the objective to predict failure events before their occurrence. Here, the term failure event means an event whose execution is undesired. Note the contrast with diagnosis, which aims at detecting failures after their occurrence [2], [3]. Failure prognosis is an active area of research (e.g., [4] and its bibliography). Due to our long background in supervisory control of DES, we approach failure prognosis as it has been studied since recently by the supervisory control community. Due to the recent involvement of this community in prognosis, there has been little prior work on prognosis by the control community. We consider that a failure prognosis is issued when the prognosis system can predict with certainty that a failure will occur in a bounded future, contrary to the statistical approach where a failure prognosis is issued when the prognosis system estimates that a failure will very probably occur in a bounded future [5].

The authors of [6] proposed a prognosis framework in the case of a partially-observed DES. A so-called prognoser issues a prognosis on whether a failure will occur, based on the partial observation it has of the plant. The notion of predictability (we will say prognosability) was defined formally for characterizing the class of languages for which: 1) every failure is prognosed before its occurrence, and 2) after a failure has been prognosed, it will certainly occur. In [7], an off-line polynomial-time and an on-line algorithms were proposed for checking prognosability.

The authors of [1] proposed a framework for the decentralized prognosis, where several local prognosers cooperate in their tasks of failure prediction. It is assumed that the local prognosers do not communicate directly among each other. Based on the partial observation it has of the plant, a local prognoser issues a prognosis “1” when it is certain that a failure will occur. Otherwise, a prognosis “0” is issued. Because of their limited observation capability, the local prognosers cooperate by transmitting their local prognoses to a fusion system that synthesizes an effective prognosis. The authors of [1] define the notion of coprognosisability for generalizing prognosability to their decentralized framework.

[1] has been generalized in [8] by developing an inference-based framework, where each local prognoser associates to its prognosis a level of ambiguity. In particular, a prognosis with level-zero ambiguity means an unambiguous prognosis. The principle is that when the fusion system receives several concurrent prognoses from the local prognosers, it will select the “winning” prognosis, that is, the prognosis with the lowest ambiguity level.

Our essential contributions consist of the following points:

1) We present the prognosis method of [1] as a disjunctive architecture.
2) We develop a conjunctive prognosis architecture, which

A. Khoumsi and H. Chakib are with the Department of Electrical and Computer Engineering, University of Sherbrooke, Sherbrooke, QC, J1L1T8 CANADA e-mail: Firstname.Lastname@USherbrooke.ca.
is dual and complementary to the disjunctive one.

3) We develop a mixed architecture that combines and generalizes the disjunctive and conjunctive architectures. More precisely, two categories of failures are supported, which are predicted using the disjunctive and conjunctive architectures, respectively.

4) In all the above points, a failure could be prognosed just before its occurrence. We will show that our work can be easily extended for predicting a failure at least $k$ steps before its occurrence, for a given $k \geq 1$.

[9] presents a short version of the disjunctive and conjunctive architectures. Compared to [9], here:

- We add as requirement that once a failure is predicted, it remains continuously predicted until its occurrence. This requirement will be called: no changed positive prediction.
- We present a mixed architecture that combines the disjunctive and conjunctive architectures (above item 3), while in [9] it is very briefly evoked.
- The explanations are more detailed in general.
- The proofs of our results are presented in the appendix.

We have noted that the inference-based framework of [8] generalizes the disjunctive architecture. It is worth noting that such an inference-based prognosis is incomparable with, and hence does not generalize, our conjunctive architecture.

The remainder of the paper is structured as follows. In Section II, we present some preliminaries and notations, and we define formally the prognosis objective by the satisfaction of three conditions: no missed prediction, no false prediction, and no changed positive prediction. Section III formulates the prognosis method of [1] as a disjunctive architecture. In Section IV, we develop a conjunctive prognosis architecture that is dual and complementary to the disjunctive one. Section V proposes a mixed architecture that combines and generalizes the disjunctive and conjunctive architectures. Section VI explains how our contributions can be easily extended for predicting a failure at least $k$ steps before its occurrence, for a given $k \geq 1$.

The same section recapitulates briefly our main contributions and proposes some relevant future work. And last but not least, the proofs are presented in the Appendix.

II. PRELIMINARIES AND NOTATIONS

We denote by $\Sigma$ the finite set of events (also called alphabet) that can be executed by the DES to be prognosed. $\Sigma^*$ is the set of all finite sequences of events of $\Sigma$, including the empty sequence $\varepsilon$. The term sequence will mean event sequence. For any sequence $\lambda \in \Sigma^*$, $|\lambda|$ denotes the length of $\lambda$, $\Sigma^k$ (resp. $\Sigma^{\leq k}$) is the subset of $\Sigma^*$ consisting of the sequences whose length is greater (resp. smaller) than or equal to $k$. Formally, $\Sigma^k = \{ \lambda \in \Sigma^* \mid |\lambda| \geq k \}$, $\Sigma^{\leq k} = \{ \lambda \in \Sigma^* \mid |\lambda| \leq k \}$. For any sequences $\lambda, \mu \in \Sigma^*$, $\mu \preceq \lambda$ means that $\mu$ is a prefix of $\lambda$, i.e., $\exists \nu \in \Sigma^*$ s.t. $\lambda = \mu \nu$. The set of all prefixes of a language $K$ is denoted $K^\preceq$, and $K$ is said prefix-closed if $K = K^\preceq$. Let $\mathbb{N}^+ = \{1, 2, 3, \cdots \}$ denote the set of strictly positive integers.

We consider the case where the plant (i.e., the DES to be prognosed) is modeled by a prefix-closed language $L$. The plant consists of a failure part and a non-failure (or healthy) part, modeled by $F$ and $H$, respectively. We have $L = H \cup F$ and $H \cap F = \emptyset$. Typically, $F$ and $H$ are related to a set of failure events $\Sigma_f \subset \Sigma$ as follows: $F$ is the part of $L$ which contains the sequences with at least one failure event, and $H$ contains the sequences without failure event. Clearly, $H$ is prefix-closed: if an execution $\lambda$ is healthy then its “past” (i.e., its prefixes) is healthy. We will use the following notions of boundary, indicator, and non-indicator sequence [1], [8]:

Definition 2.1: (Boundary, Indicator, Non-Indicator) Consider two prefix-closed languages $L$ and $H$ such that $H \subseteq L$ (and then, we deduce $F = L \setminus H$):

- A boundary sequence of $H$ w.r.t $L$ is a sequence of $H$ for which a failure in a next step is possible. The set of boundary sequences of $H$ w.r.t $L$ is formally defined by: $\partial = \{ \lambda \in H \mid \{ \lambda \} \Sigma \cap F \neq \emptyset \}$.
- An indicator sequence of $H$ w.r.t $L$ is a sequence of $H$ for which a failure in future is guaranteed. The set of indicator sequences of $H$ w.r.t $L$ is formally defined by: $\mathbb{I} = \{ \lambda \in H \mid \exists k \in \mathbb{N} : (\lambda)^{\Sigma^{2k}} \cap L = \emptyset \}$.
- A non-indicator sequence of $H$ w.r.t $L$ is a sequence of $H$ for which a failure in future is not guaranteed. The set of non-indicator sequences of $H$ w.r.t $L$ is formally defined by: $\mathbb{N} = H \setminus \mathbb{I} = \{ \lambda \in H \mid \forall k \in \mathbb{N}^+ : (\lambda)^{\Sigma^{2k}} \cap H \neq \emptyset \}$.

For simplicity, $L$ and $H$ are not indicated in $\partial$, $\mathbb{I}$ and $\mathbb{N}$. We consider the case where the task of a prognoser is to observe the plant and issue a prognosis 0 or 1 respecting the following three points:

1) Predict a failure before its occurrence. Formally: $\text{Prog}(\lambda) = 1$, if $\lambda \in \partial$.

2) Predict a failure only if the failure will certainly occur, that is, predict no failure if no failure is certain. Formally: $\text{Prog}(\lambda) = 0$, if $\lambda \in \mathbb{I}$.

3) Any prognosis is tolerated in the other situations. Formally: $\text{Prog}(\lambda) \in \{1, 0\}$, if $\lambda \in \mathbb{N} \setminus \partial$.

In [1] the prognosis $\phi$ is issued instead of 0, hence the prognosis 0 is never issued. And in [8], the prognosis $\phi$ is issued instead of 0 in Case 3: $\phi$ is issued when the prognoser cannot determine whether $\lambda \in \mathbb{N} \setminus \partial$. The frameworks of [1], [8] become consistent with ours by simply replacing prognosis $\phi$ by 0. With this adaptation, the “No missed prediction” and “No false prediction” conditions of [1], [8] become:

No missed prediction: Each failure is prognosed before its occurrence. In other words, a failure is prognosed if its occurrence is possible in the next step. Formally:

$$\forall \lambda \in \partial : \text{Prog}(\lambda) = 1$$

No false prediction: A failure prognosis guarantees that a failure will occur in future. In other words, a failure is not prognosed if its occurrence in future is not guaranteed. Formally:

$$\forall \lambda \in \mathbb{N} : \text{Prog}(\lambda) = 0$$

Eqs. (1) and (2) are defined w.r.t $\partial$ and $\mathbb{N}$, respectively. We will say that Eqs. (1,2) are defined w.r.t ($\partial$, $\mathbb{N}$).

For the sake of consistency, we will also use the “No changed positive prediction” condition which states that when-
ever the prognosis 1 is issued, the prognoser will not change its mind in future. Formally:

\[ \forall \lambda, \mu \in \Sigma^* \text{ s.t. } \lambda \mu \in \mathcal{H} : \text{Prog}(\lambda) = 1 \Rightarrow \text{Prog}(\lambda \mu) = 1 \quad (3) \]

**Remark 2.1:** Our Eqs. (1,2) correspond to Eqs. (2,3) of [1], respectively, and our Eq. (3) is not used in [1]. More precisely:

- Our Eq. (1) has not the same form as Eq. (2) of [1], but it can be easily proved that the two equations are equivalent.
- Our Eq. (2) becomes equivalent to Eq. (3) of [1], if in [1] we replace decision \( \phi \) (meaning “don’t know”) by 0.
- The architecture used in [1] (presented in Section III) guarantees our Eq. (3), but it does not state it as a requirement. The authors of [1] explain that their Eq. (2) implies that the prognoser does not change its mind. This statement must be understood as: the prognosis 1 is issued before a failure occurrence, the prognoser will not change its mind. This is different from our Eq. (3) which is obtained by replacing “the last time” by “whenever”.

Figure 1 represents decentralized prognosis [1], [8], where \( n \) local prognosers \((\text{Prog}_i)_{1 \leq i \leq n}\) observe the plant and cooperate with each other in order to predict a failure before its occurrence. After the execution of a sequence \( \lambda \in \mathcal{H} \), each \( \text{Prog}_i \) makes a local prognosis \( \text{Prog}_i(P_i(\lambda)) \in \{0,1\} \) depending on what it has observed, i.e., on \( P_i(\lambda) \). The local prognoses \((\text{Prog}_i(P_i(\lambda)))_{i=1 \cdots n}\) are combined in order to synthesize an effective prognosis \( \text{Prog}(\lambda) \in \{0,1\} \) that must satisfy Equations (1,2). In [1], the effective prognosis \( \text{Prog}(\lambda) \) is obtained by fusing disjunctively the local prognoses \( \text{Prog}_i(P_i(\lambda)) \) \((i = 1, \cdots, n)\), that is:

\[ \text{Prog}(\lambda) = \bigvee_{i=1, \cdots, n} \text{Prog}_i(P_i(\lambda)) \quad (4) \]

**Definition 3.1:** (Disjunctive prognoser, \( \lor \)-prognoser) A disjunctive prognoser, or more shortly \( \lor \)-prognoser, is a set of local prognosers whose local prognoses are combined using Equation (4) to generate an effective prognosis.

A particular \( \lor \)-prognoser is obtained by computing the local prognoses \( \text{Prog}_i(P_i(\lambda)) \) using the following rule: a local prognoser issues the local prognosis “1” when it is certain that a failure in future is guaranteed. Otherwise, it issues the local prognosis “0”. Formally:

\[ \forall i \in \{1 \cdots n\} : \text{Prog}_i(P_i(\lambda)) = \begin{cases} 0, & \text{if } P_i(\lambda) \in P_i(\mathcal{Y}) \\ 1, & \text{otherwise} \end{cases} \quad (5) \]

By analogy with a terminology in decentralized supervisory control [10], the prognoser defined by Equations (4,5) can be qualified as locally anti-permissive, in the sense that each of its local prognosers issues a local decision 0 when it is uncertain whether the global decision is 1. Hence, we use the following definition:

**Definition 3.2:** (D&A-prognoser) D&A-prognoser denotes the particular Prognoser defined by Equations (4,5). D&A is for Disjunctive and Anti-permissive.

Although the disjunctive architecture is studied in [1], we present here an example which will be useful to compare the disjunctive and conjunctive architectures in Section V. We consider the example of Fig. 2, where \( \Sigma_{a,1} = \{a_1, \sigma\} \), \( \Sigma_{a,2} = \{a_2, \sigma\} \), \( \Sigma_{u,0} = \{f\} \). We obtain \( \mathcal{H} = \sigma^*(a_1 + a_2) = \sigma^*(\varepsilon + a_1 + a_2) \), \( \mathcal{F} = \sigma^*(a_1 + a_2) f^* \), \( \partial = \sigma^*(a_1 + a_2) \), and \( \mathcal{Y} = \sigma^* \). Table I outlines the local and effective prognoses of the D&A-prognoser defined by Equations (4,5). It can be easily checked that this D&A-prognoser satisfies Equations (1,2,3).

![Fig. 1. Decentralized prognosis architecture of [1], [8]](image)

In the sequel, respecting Eqs. (1,2,3) is indeed the objective of all the proposed decentralized prognosers.

## III. DISJUNCTIVE PROGNOSIS ARCHITECTURE OF [1]

It is worth noting that the relevance of Section III is not to present new results (theorems, propositions, ...). Its relevance is rather to present the architecture of [1] as disjunctive, which inspired us to develop a dual conjunctive architecture which is presented in Section IV. Let us consider two prefix-closed languages \( \mathcal{L} \) and \( \mathcal{H} \) modeling the plant and its healthy part, respectively. The failure part is deduced by \( \mathcal{F} = \mathcal{L} \setminus \mathcal{H} \).

### B. Existence Results as Reported in [1]

We consider a pair \((\mathcal{L}, \mathcal{H})\) of prefix-closed languages such that \( \mathcal{H} \subseteq \mathcal{L} \), and their corresponding \((\partial, \mathcal{Y})\). In order to
TABLE I

<table>
<thead>
<tr>
<th>λ</th>
<th>P₁(λ)</th>
<th>Prog₁(P₁(λ))</th>
<th>P₂(λ)</th>
<th>Prog₂(P₂(λ))</th>
<th>Prog(λ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>σ₁</td>
<td>σ₁</td>
<td>0</td>
<td>σ⁺</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>σ⁺a₁</td>
<td>σ⁺a₁</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>σ⁺a₂</td>
<td>σ⁺a₂</td>
<td>0</td>
<td>σ⁺a₂</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Disjunctive prognosis results for the example of Fig. 2

The local prognoses. In the same way, in the conjunctive architecture, the effective prognosis is obtained by fusing the local prognoses conjunctively. But an important difference is that now, to guarantee Eq. (3), we need to require that once a prognosis Prog(λ) = 1 is issued, the prognoser is blocked on this positive prognosis. This can be trivially implemented by using a variable Prog, which is initialized to 0 and switches to 1 the first time Prog(λ) = 1 is issued. When Prog = 0, the prognoser computes Prog(λ) by a conjunctive equation. When Prog = 1, the prognoser issues Prog(λ) = 1 without any computation. Formally:

\[ Prog(λ) = Prog_c ∨ (∩_{i=1,⋯,n} Prog_c(P_i(λ))) \]  \hspace{1cm} (7)

**Definition 4.1:** (Conjunctive prognoser, ∨-prognoser) A conjunctive prognoser whose local prognoses are combined using Equation (7) to an generate effective prognosis.

A particular ∨-prognoser is obtained by computing the local prognoses Prog_c(P_1(λ)) using the following rule: a local prognoser issues a local prognosis "0" when it is certain that the next event is not a failure. Otherwise, the local prognosis is "1". Formally:

\[ ∀i \in \{1,⋯,n\} : Prog_c(P_i(λ)) = \begin{cases} 1, & \text{if } P_i(λ) ∈ P_i(δ) \\ 0, & \text{otherwise} \end{cases} \]  \hspace{1cm} (8)

By analogy with a terminology in decentralized supervisory control [10], the prognoser defined by Equations (7,8) can be qualified as locally permissive, in the sense that each of its local prognosers issues a local decision 1 when it is uncertain whether the global decision is 0. Hence, we use the following definition:

**Definition 4.2:** (C&P-prognoser) C&P-prognoser denotes the particular Prognoser defined by Equations (7, 8). C&P is for Conjunctive and Permissive.

Consider the example of Fig. 3, where \( \Sigma_{o,1} = \{a_1, σ\} \), \( \Sigma_{o,2} = \{a_2, σ\} \), \( \Sigma_{w,o} = \{f\} \), \( H = σ^*(a_1σa_2 + a_0σa_1) \) contains the sequences without \( f \), and \( F = σ^*(a_1σa_2 + a_2σa_1)ff^* \) contains the sequences with \( f \). \( δ = σ^*(a_1σa_2 + a_2σa_1) \) contains the sequences of \( H \) for which \( f \) is possible in the next step. \( Y = σ^*(a_1a_2)σ^* = σ^* + σ^*a_1σ^* + σ^*a_2σ^* \) contains the sequences of \( H \) for which \( f \) is not guaranteed in future. Table II outlines the local and effective prognoses of the C&P-prognoser defined by Equations (7,8). We see that the effective prognosis “Prog(λ) = 1” is issued for all sequencess of \( δ \), i.e., \( σ^*a_1σa_2 \) and \( σ^*a_2σa_1 \). And the effective prognosis “Prog(λ) = 0” is issued for all sequences of \( Y \), i.e., \( σ^*, σ^*a_1σ^* \) and \( σ^*a_2σ^* \). We also see that the prognosis “1” is never followed by a prognosis “0”. Therefore, the C&P-prognoser defined by Equations (7,8) satisfies Equations (1,2,3).

**Remark 4.1:** In this particular example, we obtain the same results even if we do not use Prog_c, i.e. if we take Eq. 7 as:

\[ Prog(λ) = ∏_{i=1,⋯,n} Prog_i(P_i(λ)) \].

**B. Existence Results**

We consider a pair \( (L,H) \) of prefix-closed languages such that \( H \subseteq L \) and their corresponding \( (δ, Y) \). In order to
Fig. 3. Example for illustrating the prognosis issued by the C&P-prognoser

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$P^*_1(\lambda)$</th>
<th>$\text{Prog}_1(P^*_1(\lambda))$</th>
<th>$P^*_2(\lambda)$</th>
<th>$\text{Prog}_2(P^*_2(\lambda))$</th>
<th>$\text{Prog}(\lambda)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^<em>_1a^</em>_2$</td>
<td>$\sigma^*$</td>
<td>0</td>
<td>$\sigma^*$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a^<em>_1a^</em>_2a^*_1$</td>
<td>$\sigma^*$</td>
<td>1</td>
<td>$\sigma^*$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$a^<em>_1a^</em>_1$</td>
<td>$\sigma^*$</td>
<td>1</td>
<td>$\sigma^*$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$a^*_1$</td>
<td>$\sigma^*$</td>
<td>0</td>
<td>$\sigma^<em>_2a^</em>$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$a^<em>_1a^</em>_1$</td>
<td>$\sigma^*$</td>
<td>1</td>
<td>$\sigma^<em>_2a^</em>$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**TABLE II**

**CONJUNCTIVE PROGNOSIS RESULTS FOR THE EXAMPLE OF FIG. 3**

determine whether there exists a $\wedge$-prognoser defined by Eq. (7) and satisfying Eqs. (1,2), we define the notion of conjunctive coproposability, or more shortly $\wedge$-COPROG. Intuitively, $(\partial, \mathcal{Y})$ is $\wedge$-COPROG if:

While a failure is not guaranteed in future, its occurrence is impossible in the next step and at least one local $\text{Prog}_i$ is aware of this impossibility.

Formally:

**Definition 4.3:** ($\wedge$-COPROG) Given a pair $(\mathcal{L}, \mathcal{H})$ of prefix-closed languages with $\mathcal{H} \subseteq \mathcal{L}$, the corresponding $(\partial, \mathcal{Y})$ is said $\wedge$-COPROG if:

$$\bigcap_{i=1,\ldots,n} [P_i^{-1} \mathcal{P}_i(\partial)] \cap \mathcal{Y} = \emptyset \quad (9)$$

The following theorem relates $\wedge$-COPROG to the C&P-prognoser.

**Theorem 4.1:** Given a pair $(\mathcal{L}, \mathcal{H})$ of prefix-closed languages with $\mathcal{H} \subseteq \mathcal{L}$ and their corresponding $(\partial, \mathcal{Y})$, we have that the C&P-prognoser defined by Eqs. (7,8) satisfies Eqs. (1,2,3), if and only if $(\partial, \mathcal{Y})$ is $\wedge$-COPROG.

Since the C&P-prognoser defined by Eqs. (7,8) is a particular $\wedge$-prognoser defined by Eqs. (7), we deduce trivially the following corollary (hence its proof is omitted):

**Corollary 4.1:** Given a pair $(\mathcal{L}, \mathcal{H})$ of prefix-closed languages with $\mathcal{H} \subseteq \mathcal{L}$ and their corresponding $(\partial, \mathcal{Y})$, we have that there exists a $\wedge$-prognoser defined by Eq. (7) and satisfying Eqs. (1,2,3), if $(\partial, \mathcal{Y})$ is $\wedge$-COPROG.

Note that $\wedge$-COPROG of $(\partial, \mathcal{Y})$ is a necessary and sufficient condition in Theorem 4.1, while it is only sufficient in Corollary 4.1. This implies that even if $(\partial, \mathcal{Y})$ is not $\wedge$-COPROG, we may have a $\wedge$-prognoser defined by Eq. (7) and satisfying Eqs. (1,2,3), but such a $\wedge$-prognoser will not satisfy Eq. (8). This is due to the use of $\text{Prog}_c$.

Let us prove the nonnecessity of $\wedge$-COPROG in Corollary 4.1 by using the following example: $\mathcal{H} = a_1(b_1a_2a^*_2 + a_2b_1)$, $\mathcal{F} = a_1a_2b_1ff^*$, $\Sigma_{0,1} = \{a_1, b_1\}$, $\Sigma_{0,2} = \{a_2\}$, $\Sigma_{uo} = \{f, \sigma\}$. We compute $\partial = a_1a_2b_1$, $\mathcal{Y} = a_1b_1a_2a^*_2$, and $P_i^{-1} \mathcal{P}_i(\partial) \cap P_j^{-1} \mathcal{P}_j(\partial) \cap \mathcal{Y} = \emptyset$. If we use the $\wedge$-prognoser defined by Eqs. (7,10), we compute that $\text{Prog}(\lambda) = 1$ uniquely for sequences of $a_1a_2b_1$, and hence Eqs. (1,2,3) are satisfied. If we remove $\text{Prog}_c$ in Eq. (7), we obtain that $\text{Prog}(a_1a_2) = 1$ but $\text{Prog}(a_1a_2b_1) = 0$, hence Eqs. (1,3) are not satisfied.

$$\forall i \in \{1, 2\} : \text{Prog}_i(P_i(\lambda)) = \begin{cases} 1, & \text{if } P_i(\lambda) = a_i \\ 0, & \text{otherwise} \end{cases} \quad (10)$$

Let us return to the example of Fig. 3, where $\mathcal{Y} = \sigma^* + a^*_1\sigma^* + a^*_2\sigma^*$, $\mathcal{P} = \sigma^*(a_1\sigma^*a_2 + a_2\sigma^*a_1)$. Let us show that $(\mathcal{L}, \mathcal{H})$ is $\wedge$-COPROG. We compute: $P_i(\partial) = \sigma^* a_i \sigma^*$, $\mathcal{P}_i^{-1} P_i(\partial) \cap \mathcal{Y} = \sigma^* a_i \sigma^*$, for $i = 1, 2$, and thus $\bigcap_{i=1,2} \mathcal{P}_i^{-1} P_i(\partial) = \emptyset$. Therefore, from Def. 4.3, $(\partial, \mathcal{Y})$ is $\wedge$-COPROG. We deduce from Theorem 4.1 that a solution is the C&P-prognoser defined by Eqs. (7,10) which is outlined in Table II.

**C. Comparison with the Inference-Based Architecture of [8]**

Let us compare the inference-based architecture of [8] with the disjunctive and conjunctive architectures. We consider the series of pairs of languages $(\mathcal{Z}[k], \mathcal{Y}[k])_{k \in \mathbb{N}}$ defined inductively as follows:

- **Basis:** $\mathcal{Z}[0] = \mathcal{Y}, \mathcal{Y}[0] = \mathcal{Y}$.
- **Inductive step:** for $k \geq 0$,

  $$\mathcal{Z}[k+1] = \mathcal{Z}[k] \cap \bigcap_{i=1,\ldots,n} \mathcal{P}_i^{-1} P_i(\mathcal{Y}[k])$$

  $$\mathcal{Y}[k+1] = \mathcal{Y}[k] \cap \bigcap_{i=1,\ldots,n} \mathcal{P}_i^{-1} P_i(\mathcal{Z}[k])$$

In [8], the condition of existence of solutions for inference-based architectures is called $N$-inference-prognosability and formally defined by $\partial \subseteq \mathcal{Z} \setminus \mathcal{Z}[N+1]$, where $N$ is any nonnegative integer.

The inference-based architecture of [8] generalizes the disjunctive architecture of Section III by the fact that the disjunctive coproposability of Def. 3.3 is equivalent to 0-inference prognosability of [8], i.e., it is a particular case of $N$-inference prognosability for $N = 0$.

To stress the contribution of Section IV, let us show that the inference-based architecture of [8] does not generalize the conjunctive architecture. For the example treated in Section IV (Figure 3), we have shown in Section IV-B that we have conjunctive coproposability (Def. 4.3). For the same example, it can be easily checked that there exist no nonnegative integer $N$ for which we have $N$-inference-prognosability. Indeed, for this example we compute $\mathcal{Z}[1] = \mathcal{Z}$ and $\mathcal{Y}[1] = \mathcal{Y}$, and hence $\mathcal{Z}[N+1] = \mathcal{Z}$ and $\mathcal{Y}[N+1] = \mathcal{Y}$. Therefore $\mathcal{Z} \setminus \mathcal{Z}[N+1] = 0$, from which we deduce $\partial \not\subseteq \mathcal{Z} \setminus \mathcal{Z}[N+1]$, that is, we have not $N$-inference-prognosability. We have shown that conjunctive coproposability does not necessarily imply $N$-inference prognosability for some nonnegative $N$. Hence, the inference-based architecture does not generalize the conjunctive architecture.

**V. MIXED ARCHITECTURE**

By using an approach used in decentralized supervisory control [10], we propose a mixed architecture that combines and generalizes the disjunctive and conjunctive architectures. We have the following proposition:
Proposition 5.1: \(\lor\)-COPROG and \(\land\)-COPROG are incomparable, i.e., none of them implies the other.

Let us prove Prop. 5.1 using the two examples of Figures 2 and 3. We have shown in Sect. III that the example of Fig. 2 is \(\lor\)-COPROG; it can be checked that it is not \(\land\)-COPROG by computing \(\bigcap_{i=1,2}[P_{i}^{-1}P_{i}(\partial)] \cap \Upsilon = \Upsilon \neq \emptyset\). And we have shown that the example of Fig. 3 is \(\land\)-COPROG; it can be checked that it is not \(\lor\)-COPROG by computing \(\bigcap_{i=1,2}[P_{i}^{-1}P_{i}(\partial)] \cap \Upsilon = \emptyset\). To recapitulate, we have found an example which is \(\lor\)-COPROG and not \(\land\)-COPROG (Fig. 2), and an example which is \(\land\)-COPROG and not \(\lor\)-COPROG (Fig. 3).

More generally, we can conceive a system with two categories, 1 and 2, of failures. A failure of category \(k\) will be called \(k\)-failure, for \(k = 1, 2\). More precisely, the failure part \(F\) is categorized in two parts \(F[1]\) and \(F[2]\), where \(F = F[1] \cup F[2]\). \(F[1]\) and \(F[2]\) are not necessarily disjoint with each other. The healthy part corresponding to each \(F[k]\) is then defined by \(H[k] = L \setminus F[k]\). For each \(k = 1, 2\), \(F[k]\) (resp. \(H[k]\)) is the part of \(L\) with (resp. without) \(k\)-failures. For each \(k = 1, 2\), we can then compute the pair \((\partial[k], \Upsilon[k])\) corresponding to \((H[k], F[k])\). The objective of the mixed architecture is to find a decomposition \((F[1], F[2])\) of \(F\), for which failures in \(F[1]\) (resp. \(F[2]\)) are predictable using a disjunctive (resp. conjunctive) architecture. From theorems 3.1 and 4.1, we deduce that we have to find a decomposition \((F[1], F[2])\) of \(F\), if any, such that:

- \((\partial[1], \Upsilon[1])\) is \(\lor\)-COPROG,
- \((\partial[2], \Upsilon[2])\) is \(\land\)-COPROG.

If the above decomposition exists, then from Theorems (3.1,4.1) and Propositions (3.1,4.1),

- The failures of \(F[1]\) can be predicted using the D&A-prognoser defined by Eqs. (4,5), but w.r.t \(\Upsilon[1]\) instead of \(\Upsilon\).
- The failures of \(F[2]\) can be predicted using the C&P-prognoser defined by Eqs. (7,8), but w.r.t \(\partial[2]\) instead of \(\Upsilon\).

VI. CONCLUSION

A. Failure Prediction in Advance

The prognosis architectures presented in Sections (III and IV) are based on the use of \(\Upsilon\) and \(\partial\). Remind that \(\partial = \{\lambda \in H | \{\lambda\} \Sigma \cap F = \emptyset\}\). This definition of \(\partial\) guarantees that the failure is predicted at the latest just before its occurrence.

Let us now consider that the objective is to predict a failure at least \(k\) steps before its occurrence, for a given \(k \geq 1\). The interest of our framework is that it remains applicable with this new objective by just replacing \(\partial\) by \(\partial(k) = \{\lambda \in H | \{\lambda\} \Sigma^{\leq k} \cap F = \emptyset\}\). Note that \(\partial(1) = \partial\).

B. Contributions and future work

We have studied the decentralized prognosis, where local prognosers cooperate in order to predict failures. Our essential contributions can be summarized by the following points:

1) In Section III, we formulate the architecture of [1] as a disjunctive (or D&A) architecture. A first relevance of that “disjunctive viewpoint” is that it clarifies the analogy with the disjunctive architectures in supervisory control and diagnosis. Moreover, in Section III we have clarified the intuitive meaning of coprognosability of [1] (which we call disjunctive coprognosability) by the following intuitive explanation: Before a failure occurs, its occurrence becomes inevitable and at least one local prognoser is aware of this inevitability. Briefly, Section III allows a better understanding of the architecture of [1].

2) The fact to present the architecture of [1] explicitly as disjunctive, inspired us for developing in Section IV a dual architecture qualified as conjunctive. Note that the inference-based architecture of [8] generalizes the disjunctive one, but it does not generalize the conjunctive one.

3) The disjunctive and conjunctive architectures are incomparable (i.e., none of them is more general than the other one). We have then been able to define in Section V a more general architecture, called mixed architecture, by combining the disjunctive and conjunctive architectures.

4) In all the above points, a failure can be predicted at the latest just before its occurrence. We show that our work can be straightforwardly adapted for predicting a failure at least \(k\) steps before its occurrence, for a given \(k \geq 1\).

In a near future, we plan to develop an efficient method for checking coprognosability and computing prognosers in the disjunctive and conjunctive architectures.

For the following proofs in Appendices A, B and C, recall that: \(\lor\)-prognoser is any prognoser defined by Eq. (4), D&A-prognoser is the prognoser defined by Eqs. (4,5), \(\land\)-prognoser is any prognoser defined by Eq. (7), and C&P-prognoser is the prognoser defined by Eqs. (7,8).

APPENDIX A

PROOF OF THEOREM 3.1

We will use the following two results that can be deduced from Theorem 1 and 5 of [1]: 1) If there exists a \(\lor\)-prognoser satisfying Eqs. (1,2), then \((\partial, \Upsilon)\) is \(\lor\)-COPROG; 2) If \((\partial, \Upsilon)\) is \(\lor\)-COPROG, then the D&A-prognoser satisfies Eqs. (1,2).

For the purpose of our proof, let us first prove that the D&A-prognoser satisfies Eq. (3). Eqs. (4,5) imply that \(\text{Prog}(\lambda) = 1\) is equivalent to \(\exists i \in \{1 \cdots n\}\) such that \(P_{i}(\lambda) \notin P_{i}(\Upsilon)\). Since \(\Upsilon\) is prefix-closed, we have that \(P_{1}(\lambda) \notin P_{1}(\Upsilon)\) implies \(P_{1}(\lambda \mu) \notin P_{1}(\Upsilon)\) for any \(\mu \in \Sigma^{*}\). By combining the last two results, we obtain \(\text{Prog}(\lambda) = 1\) implies \(\text{Prog}(\lambda \mu) = 1\) for any \(\mu \in \Sigma^{*}\). Hence, Eq. (3).

We have to prove that there exists a \(\lor\)-prognoser satisfying Eqs. (1,2,3) if and only if \((\partial, \Upsilon)\) is \(\lor\)-COPROG. The “only if” is trivially deduced from the above result 1 of [1], because the satisfaction of Eqs. (1,2,3) implies the satisfaction of Eqs. (1,2). Let us prove the “if”. We therefore assume that \((\partial, \Upsilon)\) is \(\lor\)-COPROG. This assumption and the above result 2 of [1] imply that the D&A-prognoser satisfies Eqs. (1,2). And we have proved above that the D&A-prognoser also satisfies...
Eq. (3). Therefore, if \((\partial, \Upsilon)\) is \(\triangledown\)-COPROG, then the D&A-prognoser satisfies Eqs. (1,2,3).

APPENDIX B
PROOF OF PROPOSITION 3.1
Consider a pair \((L, H)\) of prefix-closed languages with \(H \subseteq L\). We have to prove that the D&A-prognoser satisfies Eqs. (1,2,3) if and only if \((\partial, \Upsilon)\) is \(\triangledown\)-COPROG. The "only if" is trivially deduced from the "only if" of Theorem 3.1 because the D&A-prognoser is a particular \(\check{\triangledown}\)-prognoser. Let us prove the "if". We therefore assume that \((\partial, \Upsilon)\) is \(\triangledown\)-COPROG. This assumption and the proof of the "if" of Theorem 3.1 imply that the D&A-prognoser satisfies Eqs. (1,2,3).

APPENDIX C
PROOF OF THEOREM 4.1
Consider a pair \((L, H)\) of prefix-closed languages with \(H \subseteq L\). We have to prove that the C&P-prognoser satisfies Eqs. (1,2,3), if and only if \((\partial, \Upsilon)\) is \(\land\)-COPROG.

A. Proof of "if"

We assume that \((\Upsilon, \partial)\) is \(\land\)-COPROG, that is, Eq. (9) is satisfied. Let us prove that Eqs. (1,2,3) are satisfied by the C&P-prognoser. Eq. (3) is guaranteed by Eq. (7). Therefore, it remains to prove Eqs. (1,2).

**Proof of Equation (2):** Let us prove that if Eq. (9) is satisfied, then for every \(\lambda \in \Upsilon\), we obtain \(\text{Prog}(\lambda) = 0\) using the C&P-prognoser. Assume therefore that Eq. (9) is satisfied and consider \(\lambda \in \Upsilon\).

1) Eq. (9) and \(\lambda \in \Upsilon\) imply that \(\exists i \in \{1, \cdots, n\}\) such that \(P_i(\lambda) \not\in P_i(\partial)\).
2) Item 1 and Eq. (8) imply that \(\exists i \in \{1, \cdots, n\}\) such that \(\text{Prog}_i(P_i(\lambda)) = 0\), and hence \(\bigwedge_{i=1}^{n} \text{Prog}_i(P_i(\lambda)) = 0\).
3) The fact that \(\Upsilon\) is prefix-closed implies that \(\lambda\) is pre-ceeded uniquely by sequences of \(\Upsilon\).
4) Items 2 and 3 imply that until \(\lambda\) is reached, Eq. (7) is used with \(\text{Prog}_i = 0\).
5) Items 2 and 4 imply that \(\text{Prog}(\lambda) = 0\).

**Proof of Equation (1):** Let us prove that for every \(\lambda \in \partial\), we obtain \(\text{Prog}(\lambda) = 1\) using the C&P-prognoser. Consider therefore \(\lambda \in \partial\).

1) \(\lambda \in \partial\) implies that \(\forall i \in \{1, \cdots, n\}, P_i(\lambda) \in P_i(\partial)\).
2) Item 1 and Eq. (8) imply that \(\forall i \in \{1, \cdots, n\}\), \(\text{Prog}_i(P_i(\lambda)) = 1\).
3) Item 2 and Eq. (7) imply that \(\text{Prog}(\lambda) = 1\).

B. Proof of "only if"

We assume that the C&P-prognoser satisfies Eqs. (1,2,3). Let us prove that \((\Upsilon, \partial)\) is \(\land\)-COPROG, that is, \(\bigcap_{i=1, \cdots, n} [P_i^{-1}(\partial)] \cap \Upsilon = \emptyset\) is satisfied. This is equivalent to prove the following expression: \(\lambda \in \Upsilon \Rightarrow \exists i \in \{1, \cdots, n\}\) such that \(P_i(\lambda) \not\in P_i(\partial)\). Let us therefore consider \(\lambda \in \Upsilon\). From Eq. (2), we deduce \(\text{Prog}(\lambda) = 0\). Hence, from Eq. (7), we have: \(\exists i \in \{1, \cdots, n\}\) such that \(\text{Prog}_i(P_i(\lambda)) = 0\). Then, from Eq. (8), we obtain: \(\exists i \in \{1, \cdots, n\}\) such that \(P_i(\lambda) \not\in P_i(\partial)\).

REFERENCES


