Decentralized Supervisory Control of Discrete Event Systems: involving the fusion system in the decision-making

Ahmed Khoumsi and Hicham Chakib

Abstract—In the decentralized control of Discrete Event Systems, a set of supervisors take enabling/disabling decisions which can be fused for deducing actual decisions. The first architecture that has been developed for decentralized control is referred to as the conjunctive architecture. Several more general architectures have subsequently been proposed.

We propose a new architecture, whose basic principle is as follows: for every event which is controllable by several supervisors, each of the latter takes an enabling/disabling decision only when it is sure that this is the right decision which can be applied to the plant. Otherwise, the supervisor transmits its local information to the fusion system which will thus be involved in the enabling/disabling decision-making.

We compare our approach with the previous decentralized approaches.

I. INTRODUCTION

Discrete event systems (DES) have their behaviors defined by the sequences of events they can execute [1]. This paper deals with decentralized supervisory control (for brevity, the term supervisory can be omitted) of DES, where a set of supervisors cooperate in order to restrict the behavior of a plant so that it respects a specification. Each supervisor Supi observes only a part Σc,i of the events of the plant and can control only a part Σe,i of the events of the plant.

Many articles have studied decentralized control of DES [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14]. The first decentralized control architecture that has been proposed is referred to as the conjunctive architecture [5], and most of the results on decentralized control are based on this architecture. The authors of [7] present the first study of decentralized control with different fusion rules.

The authors of [11] propose a disjunctive architecture which is complementary with the conjunctive one. They also propose a general architecture that combines and generalizes the conjunctive and disjunctive ones. The authors of [12] use a conditional architecture, which generalizes the general architecture of [11] by authorizing the supervisors to take conditional decisions. In [8], [13], an approach is proposed where each supervisor can take decisions based on the ambiguities of the supervisors. The authors of [14] propose a general inference-based framework for managing ambiguity. They show that the architectures of [11], [12] are special cases in their framework.

The basic principle used in all these architectures, is that each each supervisor Supi takes a control decision for each event σ ∈ Σc,i, according to its local observations. When several supervisors take decisions concurrently for the same event, a global decision is synthesized by “fusing” these local decisions. The difference between these architectures is on the type of decisions (e.g., unconditional, conditional) taken by the supervisors, and consequently on the type of fusion.

A. Motivation and Principle of Our New architecture

In all previous decentralized architectures, each supervisor that controls an event σ always takes an enabling decision on σ, even when it is not sure of its decision. And when σ is controllable by several supervisors, a fusion module Fuseσ is involved as follows: Fuseσ receives decisions from each supervisor that controls σ, and then Fuseσ combines these decisions and generates an actual enabling decision on σ. Our point of view about this approach is the following: the decision taken by each supervisor is in fact a coarse element of information and is constructed from the more precise information which is the observation of the supervisor. In other words, each supervisor converts a precise information into a coarse information before sending it to a fusion module which takes the actual decision.

The above point of view has inspired us for proposing an architecture based on the following approach:

The information sent from a supervisor to a fusion module will be richer (i.e., more precise) than a simple decision.

The richer information can for example be the event sequence observed by the supervisor or some related information. Consequently, the decision taken by a fusion module will be based on a more complete information than in previous architectures.

We will explain later why our architecture can be conceptually seen as a special conditional architecture, where the conditions are more complex than in [12] for example.

The contribution and organization of this paper are:

1) Section II introduces the studies which, to our knowledge, are among the most advanced related work, namely: the general, the conditional and the inference-based architectures of [11], [12], [14], respectively.

2) In Section III, we present our new control architecture.

3) Section IV presents the notions of n-observability and feasibility which, together with controllability, characterize the class of languages achievable with our control architecture. We show that this class includes the classes achievable with the general, the conditional and the inference-base architectures of [11], [12], [14].
4) In Section V, we present some properties of \( n \)-observability and show that the infimal \( n \)-observable language of a class of languages exists, but the suprernal one does not necessarily exist. We also discuss about the synthesis of \( n \)-observable languages when the specification is non-achievable.

5) And in Section VI, we conclude by recapitulating our contributions and proposing relevant future work.

II. EXISTING ARCHITECTURES

In this section, we present succinctly the general architecture of [11], the conditional architecture of [12], and the inference-based framework of [14]. The general architecture of [11] is based on two more basic control architectures: the conjunctive and disjunctive architectures.

A. Conjunctive Control Architecture

Most of the studies on decentralized supervisory control are based on the conjunctive architecture [5], where the plant is controlled by \( n \) (>1) supervisors \( Sup_i \), \( i = 1, \ldots n \). Each \( Sup_i \) observes the plant through a projection mapping \( P_i \) which results in the observation of only a part \( \Sigma_{c,i} \) of the events of the plant. And each \( Sup_i \) can control only a part \( \Sigma_{c,i} \) of the events of the plant. Collectively, the supervisors observe \( \Sigma_o = \Sigma_{o,1} \cup \cdots \Sigma_{o,n} \) and can control \( \Sigma_c = \Sigma_{c,1} \cup \cdots \Sigma_{c,n} \). If \( \Sigma \) is the alphabet of the plant, we define \( \Sigma_{uo} = \Sigma \setminus \Sigma_o \) and \( \Sigma_{uc} = \Sigma \setminus \Sigma_c \), the sets of unobservable and uncontrollable events, respectively.

Each \( Sup_i \) continuously observes the plant through \( P_i \) and takes decisions consisting in disabling or enabling events. For a given event \( \sigma \in \Sigma_c \), every \( Sup_i \) that controls \( \sigma \) (i.e., \( \sigma \in \Sigma_{c,i} \)) takes an enabling/disabling decision on \( \sigma \). The local decisions taken by all the supervisors that control \( \sigma \) are fused in order to generate the actual decision that will be applied to the plant. With the conjunctive architecture:

1) A supervisor locally disables an event \( \sigma \) when and only when it is sure that \( \sigma \) is not accepted by the desired behaviour.

2) The local decisions of the supervisors are fused by intersection (or conjunctively), in order to generate the actual decision that will be applied to the plant. That is, an event is actually enabled if and only if it is locally enabled by all the supervisors

Note that an additional characteristics which is required in the conjunctive architecture, is that a controllable event \( \sigma \in \Sigma_c \) must be enabled by every \( Sup_i \) that has no control on it (i.e., \( \sigma \notin \Sigma_{c,i} \)). This requirement permits to simplify the representation of the conjunctive architecture as follows: for every event \( \sigma \in \Sigma_c \), the fusion is applied to all the supervisors, instead of being applied to only the supervisors that control \( \sigma \). Consequently, we have not to define, for each event, a set of supervisors involved in the fusion. In our opinion, this simplification is the unique advantage for having supervisors involved in the enabling of the events on which they have no control.

B. Disjunctive Control Architecture

The authors of [11] have proposed the disjunctive architecture. The only differences with the conjunctive architecture are related to the two points 1 and 2 which become:

1) A supervisor locally enables an event \( \sigma \) when and only when it is sure that \( \sigma \) is accepted by the desired behavior.

2) The local decisions of the supervisors are fused by union (or disjunctively), in order to generate the actual decision that will be applied to the plant. That is, an event is actually disabled if and only if it is locally disabled by all the supervisors.

Like with the conjunctive architecture, a requirement has been used in order to simplify the representation of the disjunctive architecture: every event \( \sigma \in \Sigma_c \) must be disabled by all the supervisors that have no control on it.


A general control architecture combining the conjunctive and disjunctive architectures is proposed in [11]. The idea is to partition the set \( \Sigma_c \) of controllable events into two disjoint sets \( \Sigma_{c,\wedge} \) and \( \Sigma_{c,\vee} \), that is, \( \Sigma_c = \Sigma_{c,\wedge} \cup \Sigma_{c,\vee} \) and \( \Sigma_{c,\wedge} \cap \Sigma_{c,\vee} = \emptyset \). The conjunctive and disjunctive architectures are applied to the events of \( \Sigma_{c,\wedge} \) and \( \Sigma_{c,\vee} \), respectively.

The authors of [11] show that a partition \((\Sigma_{c,\wedge},\Sigma_{c,\vee})\) can be found such that the class of languages achievable by the general architecture strictly includes those of the conjunctive and disjunctive architectures.

D. Conditional Control Architecture of [12]

The conditional architecture of [12] uses the same type of partitioning as in the general architecture of [11], but it allows each supervisor to take conditional decisions such as:

- Enable \( \sigma \) if no other supervisor disable \( \sigma \),
- Disable \( \sigma \) if no other supervisor enable \( \sigma \).

The authors of [12] show that the class of languages achievable by the conditional architecture strictly includes those of the general architecture.

E. Inference-Based Framework of [14]

The authors of [14] propose a general framework where each supervisor can take decisions based on its ambiguities together with the ambiguities of the other supervisors. The idea is that each supervisor associates to its decision a grade or level of ambiguity. In particular, a control decision with level-zero ambiguity means an unambiguous enabling/disabling decision. The principle is that when the fusion system receives several concurrent decisions from the supervisors, it will select the “winning” decision, that is, the decision with the lowest ambiguity level. With this approach, it can be be shown that the general architecture of [11] is a special case where the ambiguity level of the winning decisions is always 0. As another example, the conditional architecture of [12] is a special case where the ambiguity level of the winning decisions is at most 1.

Let \( \mathcal{L}(G) \) be the prefix-closed language of the plant, and \( K \) be the language of the specification. The authors of [14]
define a monotonically decreasing sequence of language pairs \((D_k(\mathcal{K}, \sigma), E_k(\mathcal{K}, \sigma))\) as follows, where \(\text{Ind}_\sigma\) is the set of indices \(i\) such that \(\sigma \in \Sigma_{c,i}^1:\)

- \(D_0(\mathcal{K}, \sigma) = \{ s \in \mathcal{K} | s \sigma \in \mathcal{L}(G) \setminus \mathcal{K} \}\),
- \(E_0(\mathcal{K}, \sigma) = \{ s \in \mathcal{K} | s \sigma \in \mathcal{K} \}\),
- \(D_{k+1}(\mathcal{K}, \sigma) = D_k(\mathcal{K}, \sigma) \cap (\bigcap_{i \in \text{Ind}_\sigma} P_i^{-1}(E_k(\mathcal{K}, \sigma)))\),
- \(E_{k+1}(\mathcal{K}, \sigma) = E_k(\mathcal{K}, \sigma) \cap (\bigcap_{i \in \text{Ind}_\sigma} P_i^{-1}(D_k(\mathcal{K}, \sigma)))\).

A specification \(\mathcal{K}\) is said to be \(N\)-inference-coobservable iff \(\forall \sigma \in \Sigma_c: D_N(\mathcal{K}, \sigma) = E_N(\mathcal{K}, \sigma) = \emptyset\). And it is said to be \(N\)-inference-coobservable iff there exists \(N\) such that it is \(N\)-inference-coobservable.

III. NEW CONTROL APPROACH

A. Motivation and Principle of Our New Architecture

In previous decentralized architectures, when an event \(\sigma\) is controllable by several supervisors, a fusion module \(\text{Fuse}_\sigma\) receives decisions from each of these supervisors, and then \(\text{Fuse}_\sigma\) combines these decisions and generates an actual enabling decision on \(\sigma\). Our point of view about this approach is that the decision taken by each supervisor is a coarse element of information and is constructed from the more precise information which is the observation of the supervisor. That is, each supervisor converts a precise information into a coarse information before sending it to a fusion module who takes the actual decision.

In this paper, we propose a new architecture which is motivated by the following reasoning: why does a supervisor transmit a coarse information instead of transmitting a richer information. The latter can for example be the event sequence the observer has observed or some related information.

The supervisors and the fusion modules are presented in Sections III-A.1 and III-A.2, respectively. We will see how these components behave in our architecture for controlling a plant so that it conforms to a specification.

Remark 3.1: Note that we do not propose to add new communication channels between supervisors and fusion modules. We just propose to transmit richer information on the existing channels.

Remark 3.2: For clarity and simplicity, we will consider uniquely controllable events. It is implicitly assumed that uncontrollable events are always enabled.

1) The Supervisors: let \(G\) be a plant generating the prefix-closed language \(\mathcal{L}(G)\) and the marked language \(\mathcal{L}_m(G)\). We consider a specification generating the language \(\mathcal{K}\). We use the notation \(\overline{\mathcal{L}}\) meaning the prefix-closure of a language \(\mathcal{L}\). For example, \(\overline{\mathcal{L}(G)} = \mathcal{L}(G)\). We assume \(\mathcal{K} \subseteq \mathcal{L}_m(G)\) and each \(\text{Sup}_i\) knows \(\mathcal{L}(G)\), \(\mathcal{L}_m(G)\) and \(\mathcal{K}\).

The supervisors observe continuously (and partially) the plant and generate outputs that are sent to the fusion system. Each \(\text{Sup}_i\) (\(i = 1, \cdots, n\)) generates one of the following three types of outputs, for every \(\sigma \in \Sigma_{c,i}^1:\)

- When \(\text{Sup}_i\) is sure that \(\sigma\) is accepted by \(\mathcal{K}\), its output is “Enable \(\sigma\)”.
- When \(\text{Sup}_i\) is sure that \(\sigma\) is not accepted by \(\mathcal{K}\), its output is “Disable \(\sigma\)”.
- When \(\text{Sup}_i\) cannot determine with certainty whether \(\sigma\) is accepted or not by \(\mathcal{K}\), this means that \(\text{Sup}_i\) has not enough information. In this case, its output consists of its elements of information that are relevant (but insufficient) for taking an enabling/disabling decision.

Therefore, the idea is that a supervisor behaves as an actual controller (by taking enabling/disabling decisions), only when it has enough information to be sure of its decisions. Otherwise, the supervisor transmits to the fusion system its relevant elements of information.

2) The Fusion System: it can be seen as a set of fusion modules such that to every controllable event \(\sigma \in \Sigma_c\) corresponds a fusion module \(\text{Fuse}_\sigma\). The outputs of the supervisors \(\text{Sup}_i\) that control \(\sigma\) (i.e., such that \(\sigma \in \Sigma_{c,i}\)) are sent to \(\text{Fuse}_\sigma\). The task of the latter is then to generate a decision “Enable \(\sigma\)” or “Disable \(\sigma\)” that is applied to the plant. More precisely, the decisions of \(\text{Fuse}_\sigma\) are as follows:

- If at least one of the supervisors has an output “Enable \(\sigma\)”, then \(\text{Fuse}_\sigma\) decides to Enable \(\sigma\).
- If at least one of the supervisors has an output “Disable \(\sigma\)”, then \(\text{Fuse}_\sigma\) decides to Disable \(\sigma\).
- If no supervisor has taken a decision on \(\sigma\) (i.e., \(\text{Fuse}_\sigma\) has received local elements of information from each \(\text{Sup}_i\) that controls \(\sigma\)), then \(\text{Fuse}_\sigma\) combines all these elements of information in order to construct a more complete information. From the latter, \(\text{Fuse}_\sigma\) tries to determine with certainty whether \(\sigma\) must be enabled or disabled and then applies its decision to the plant.

If the module \(\text{Fuse}_\sigma\) cannot decide of the enabling/disabling of a controllable event \(\sigma\) when the latter is accepted by \(\mathcal{L}(G)\), then \(\mathcal{K}\) is not achievable, that is, the control system (consisting of the supervisors and the fusion system) cannot force the plant to conform to \(\mathcal{K}\). In Section IV-C, we will present necessary and sufficient conditions for the achievability of \(\mathcal{K}\).

Our new architecture is illustrated in Figure 1 for a given \(\sigma \in \Sigma_c\).

![Figure 1. Our new architecture](image-url)

Remark 3.3: In the particular case of an event \(\sigma\) is controllable by a single supervisor \(\text{Sup}_1\), \(\text{Fuse}_\sigma\) is integrated into \(\text{Sup}_1\).

Remark 3.4: In the particular case where every event is controllable by at most a single supervisor, the fusion
system does not exist and all the decentralized architectures (including ours) are totally equivalent.

B. Formal Description of Our Architecture

Recall that $L(G)$ and $L_m(G)$ are the prefix-closed and marked languages of the plant. $K$ is the language of the specification, $K \subseteq L_m(G)$, and $P_i$ denotes the projection mapping onto the alphabet $\Sigma_{\alpha,i}$. Let us now define more formally the supervisors and the fusion modules which have been defined intuitively in sections III-A.1 and III-A.2.

1) Formal Description of the Supervisors: $Sup_i$ observes continuously (and partially) the plant. For every $\sigma \in \Sigma_{c,i}$ and every observed event sequence $\mu_i \in P_i(K)$, $Sup_i$ computes the following two languages:

$$
\begin{align*}
REJ_{i,\sigma}^\mu &= (P_i^{-1}(\mu_i) \cap K) \cap (L(G) \setminus \overline{K}) \\
ACC_{i,\sigma}^\mu &= P_i^{-1}(\mu_i) \cap \overline{K}
\end{align*}
$$

(1)

Note that $ACC_{i,\sigma}^\mu \subseteq K$ and $REJ_{i,\sigma}^\mu \subseteq L(G) \setminus \overline{K}$. The interpretation of these two languages is that:

- If $\sigma$ is accepted by $K$, then $\sigma$ leads to a sequence of $ACC_{i,\sigma}^\mu$.
- If $\sigma$ is accepted by $L(G) \setminus \overline{K}$ (we will also say: $\sigma$ is rejected by $K$), then $\sigma$ leads to a sequence of $REJ_{i,\sigma}^\mu$.

Therefore, $Sup_i$ generates one of the following three types of outputs, for every $\sigma \in \Sigma_{c,i}$:

1) When $REJ_{i,\sigma}^\mu = \emptyset$, $Sup_i$ deduces with certainty that $\sigma$ is accepted by $K$ if it is accepted by $L(G)$, and thus, its output is “Enable $\sigma$”.

2) When $ACC_{i,\sigma}^\mu = \emptyset$, $Sup_i$ deduces with certainty that $\sigma$ is rejected by $K$, and thus, its output is “Disable $\sigma$”.

3) When $ACC_{i,\sigma}^\mu \neq \emptyset$ and $REJ_{i,\sigma}^\mu \neq \emptyset$, $Sup_i$ cannot determine with certainty if $\sigma$ is accepted or not by $K$.

In this case, its output is the pair $(ACC_{i,\sigma}^\mu; REJ_{i,\sigma}^\mu) \in (K \times (L(G) \setminus \overline{K}))$.

Since the two situations $REJ_{i,\sigma}^\mu = \emptyset$ and $ACC_{i,\sigma}^\mu = \emptyset$ generate contradictory decisions ($En$ and $Dis$), it may seem that the following situation is problematic: $ACC_{i,\sigma}^\mu = REJ_{i,\sigma}^\mu = \emptyset$. Actually, there is no problem because this situation occurs only if $(P_i^{-1}(\mu_i) \cap \overline{K}) \cap L(G) = \emptyset$, that is, when $\sigma$ is not accepted by $L(G)$. In this case, $Sup_i$ knows that its decision does not matter, and thus, can take any of the two decisions (enable or disable $\sigma$).

For every $Sup_i$ and $\sigma \in \Sigma_{c,i}$, let $Sup_{i,\sigma}$ be a function that, to every observed event sequence $\mu_i \in P_i(K)$, associates the corresponding output of $Sup_i$. Let $En$ and $Dis$ denote “Enable” and “Disable”, that is, $Sup_{i,\sigma}(\mu_i) = En$ (resp. $Dis$) means: $\sigma$ is enabled (resp. disabled) by $Sup_i$ after the observation of $\mu_i$. Formally: $\forall i \in \{1, \ldots, n\}, \forall \sigma \in \Sigma_{c,i}$, $Sup_{i,\sigma} : P_i(K) \rightarrow \{En, Dis\} \cup (K \times (L(G) \setminus \overline{K}))$.

More precisely, we have for $\mu_i \in P_i(K)$:

$$
Sup_{i,\sigma}(\mu_i) =
\begin{cases}
En, & \text{if } REJ_{i,\sigma}^\mu = \emptyset \\
Dis, & \text{if } ACC_{i,\sigma}^\mu = \emptyset \\
(ACC_{i,\sigma}^\mu, REJ_{i,\sigma}^\mu), & \text{otherwise}
\end{cases}
$$

(2)

2) Formal Description of the Fusion System: For every event $\sigma \in \Sigma_c$, recall that $Ind_{\sigma}$ is the set of indices $i$ such that $\sigma \in \Sigma_{c,i}$. The module $Fuse_{\sigma}$ can be conceptually seen as a function that, to a combination of outputs of the $Sup_i$ such that $i \in Ind_{\sigma}$, associates an enabling/disabling decision on $\sigma$. Formally: $\forall \sigma \in \Sigma_c, Fuse_{\sigma} : \prod_{i \in Ind_{\sigma}} Sup_{i,\sigma} \rightarrow \{En, Dis\}$.

More precisely, we have:

$$
Fuse_{\sigma} \left( \prod_{i \in Ind_{\sigma}} Sup_{i,\sigma}(\mu_i) \right) =
\begin{cases}
En, & \text{if } (\exists i \in Ind_{\sigma} \text{ s.t. } Sup_{i,\sigma}(\mu_i) = En) \lor \\
 & (\bigcap_{i \in Ind_{\sigma}} Acc_{i,\sigma}^\mu = \emptyset) \\
Dis, & \text{if } (\exists i \in Ind_{\sigma} \text{ s.t. } Sup_{i,\sigma}(\mu_i) = Dis) \lor \\
 & (\bigcap_{i \in Ind_{\sigma}} REJ_{i,\sigma}^\mu = \emptyset) \\
Any, & \text{otherwise}
\end{cases}
$$

(3)

Note that in Eq. (3), each of $En$ and $Dis$ has two conditions. Here are some explanations related to this equation:

- The first condition of $En$ (resp. $Dis$) means that the module $Fuse_{\sigma}$ has received an $En$ (resp. $Dis$) from at least one $Sup_i$. In this case, $Fuse_{\sigma}$ applies this decision to the plant. Notice that $Fuse_{\sigma}$ may receive two conflicting decisions $En$ and $Dis$ only if $\sigma$ is not accepted by $L(G)$. In this case, $Fuse_{\sigma}$ knows that its decision does not matter, and thus, can apply any of the two decisions $En$ or $Dis$ (see our discussion on $ACC_{i,\sigma}^\mu = REJ_{i,\sigma}^\mu = \emptyset$ in Subsection III-B.1).

- The second conditions of $En$ and $Dis$ occur when $Fuse_{\sigma}$ receives no decision from the supervisors. Instead, it receives the sets $ACC_{i,\sigma}^\mu$ and $REJ_{i,\sigma}^\mu$, for $i \in Ind_{\sigma}$. $Fuse_{\sigma}$ computes the intersections $\bigcap_{i \in Ind_{\sigma}} REJ_{i,\sigma}^\mu$, and $\bigcap_{i \in Ind_{\sigma}} ACC_{i,\sigma}^\mu$, which are in some sense, refinements of the information elements received from the supervisors. The interpretation of $\bigcap_{i \in Ind_{\sigma}} REJ_{i,\sigma}^\mu$ and $\bigcap_{i \in Ind_{\sigma}} ACC_{i,\sigma}^\mu$ is that:

  - If $\sigma$ is accepted by $K$, then it leads to a sequence $\bigcap_{i \in Ind_{\sigma}} ACC_{i,\sigma}^\mu$. Therefore, $\bigcap_{i \in Ind_{\sigma}} ACC_{i,\sigma}^\mu = \emptyset$ implies that $\sigma$ is certainly rejected by $K$, and thus, is disabled by $Fuse_{\sigma}$.

  - If $\sigma$ is accepted by $L(G) \setminus K$, then it leads to a sequence $\bigcap_{i \in Ind_{\sigma}} REJ_{i,\sigma}^\mu$. Therefore, $\bigcap_{i \in Ind_{\sigma}} REJ_{i,\sigma}^\mu = \emptyset$ implies that $\sigma$ is certainly accepted by $K$ if it is accepted by $L(G)$, and thus, is enabled by $Fuse_{\sigma}$.

- The situation $Any$ occurs when both $\bigcap_{i \in Ind_{\sigma}} ACC_{i,\sigma}^\mu$ and $\bigcap_{i \in Ind_{\sigma}} REJ_{i,\sigma}^\mu$ are not empty, that is, when $Fuse_{\sigma}$ has not enough information to decide. The occurrence of $Any$ is not a problem when $\sigma$ is not accepted by $L(G)$, because the decision of $Fuse_{\sigma}$ does not matter in such a case.

On the other hand, the occurrence of $Any$ when $\sigma$ is accepted by $L(G)$, means that $K$ is not achievable with our architecture, that is, the control system cannot force the plant to conform to $K$. In Section IV, we will present necessary and sufficient conditions for the achievability of $K$. 
For the situation $\bigcap_{i \in \text{Ind}_\sigma} \text{ACC}^\mu_{i,\sigma} = \bigcap_{i \in \text{Ind}_\sigma} \text{REJ}^\mu_{i,\sigma} = \emptyset$, we can make similar comments as those already made for $\text{REJ}^\mu_{i,\sigma} = \text{ACC}^\mu_{i,\sigma} = \emptyset$. That is, the situation $\bigcap_{i \in \text{Ind}_\sigma} \text{ACC}^\mu_{i,\sigma} = \bigcap_{i \in \text{Ind}_\sigma} \text{REJ}^\mu_{i,\sigma} = \emptyset$ occurs only if $\sigma$ is not accepted by $\mathcal{L}(G)$. In this case, $\text{Fuse}_\sigma$ knows that its decision does not matter, and thus, can take any of the two decisions $\text{En}$ or $\text{Dis}$.

C. Simple Comparison with the Conjunctive and Disjunctive Architectures

We have seen that with our architecture, when a supervisor $\text{Sup}_i$ cannot take with certainty a enabling decision on an event $\sigma$, then it sends the languages $\text{ACC}^\mu_{i,\sigma}$ and $\text{REJ}^\mu_{i,\sigma}$ to $\text{Fuse}_\sigma$. After the reception of these languages from the supervisors $\text{Sup}_i$ (s.t. $i \in \text{Ind}_\sigma$), $\text{Fuse}_\sigma$ takes its decision using the following rules:

- $\text{En}$, if $(\bigcap_{i \in \text{Ind}_\sigma} \text{REJ}^\mu_{i,\sigma} = \emptyset)$
- $\text{Dis}$, if $(\bigcap_{i \in \text{Ind}_\sigma} \text{ACC}^\mu_{i,\sigma} = \emptyset)$

In order to clarify formally the difference between our architecture and the conjunctive and disjunctive architectures [5], [11], let us present these two architectures by using rules based on $\text{ACC}^\mu_{i,\sigma}$ and $\text{REJ}^\mu_{i,\sigma}$.

The conjunctive architecture (see Section II-A) is equivalent to the case where $\text{Fuse}_\sigma$ takes its decision using the following rule:

- $\text{Dis}$, iff $\exists i \in \text{Ind}_\sigma$ such that $\text{ACC}^\mu_{i,\sigma} = \emptyset$.

The disjunctive architecture (see Section II-B) is equivalent to the case where $\text{Fuse}_\sigma$ takes its decision using the following rule:

- $\text{En}$, iff $\exists i \in \text{Ind}_\sigma$ such that $\text{REJ}^\mu_{i,\sigma} = \emptyset$.

HELLO

D. Special Conditional Architecture

Let us explain how our new architecture can be seen as a special conditional architecture. We have seen that for each $\sigma \in \Sigma_c$, each $\text{Sup}_i$ (i.e. $i \in \text{Ind}_\sigma$) computes two languages $\text{ACC}^\mu_{i,\sigma}$ and $\text{REJ}^\mu_{i,\sigma}$. The fusion module $\text{Fuse}_\sigma$ computes the intersections $A = \bigcap_{i \in \text{Ind}_\sigma} \text{ACC}^\mu_{i,\sigma}$ and $B = \bigcap_{i \in \text{Ind}_\sigma} \text{REJ}^\mu_{i,\sigma}$. $\text{Fuse}_\sigma$ will enable (resp. disable) $\sigma$ if $B$ (resp. $A$) is empty. This is clearly equivalent to say that all the supervisors take the same following conditional decision:

- Enable $\sigma$ if $\bigcap_{i \in \text{Ind}_\sigma} \text{REJ}^\mu_{i,\sigma} = \emptyset$
- Disable $\sigma$ if $\bigcap_{i \in \text{Ind}_\sigma} \text{ACC}^\mu_{i,\sigma} = \emptyset$

$\text{Fuse}_\sigma$ needs to receive the languages $\text{ACC}^\mu_{i,\sigma}$ and $\text{REJ}^\mu_{i,\sigma}$ from the supervisors, in order to compute the two conditions.

E. Independency the Fusion System

Although our architecture generalizes the previous architectures, it keeps an important advantage which is the following:

Recall that the essential operations executed by a fusion module $\text{Fuse}_\sigma$ are the computations of: $(\bigcap_{i \in \text{Ind}_\sigma} \text{ACC}^\mu_{i,\sigma})$ and $(\bigcap_{i \in \text{Ind}_\sigma} \text{REJ}^\mu_{i,\sigma})$. That is, $\text{Fuse}_\sigma$ computes intersections of languages received from the supervisors. Therefore, these operations are independent of $\mathcal{L}(G), \mathcal{L}_m(G)$ and $\mathcal{K}$, and thus, the fusion modules have not to be adapted when $\mathcal{L}(G), \mathcal{L}_m(G)$ and/or $\mathcal{K}$ are modified.

F. Three Variants of our Architecture

We have seen in Subsection III-B.1, that $\text{Sup}_i$ generates one of the following types of outputs, for every $\sigma \in \Sigma_c$:

- “Enable $\sigma$”, if $\text{REJ}^\mu_{i,\sigma} = \emptyset$.
- “Disable $\sigma$”, if $\text{ACC}^\mu_{i,\sigma} = \emptyset$.
- $(\text{ACC}^\mu_{i,\sigma}, \text{REJ}^\mu_{i,\sigma})$, if $\text{ACC}^\mu_{i,\sigma} \neq \emptyset$ and $\text{REJ}^\mu_{i,\sigma} \neq \emptyset$.

Let us present three variants of our architecture which have the same class of achievable languages.

1) First Variant: For $\sigma \in \Sigma_c$, each $\text{Sup}_i$ never takes a decision for $\sigma$ and its output is $(\text{ACC}^\mu_{i,\sigma}, \text{REJ}^\mu_{i,\sigma})$. Therefore, decisions on $\sigma$ are always taken by the fusion system.

2) Second Variant: When $\text{ACC}^\mu_{i,\sigma} \neq \emptyset$ and $\text{REJ}^\mu_{i,\sigma} \neq \emptyset$, $\text{Sup}_i$ behaves as an observer, that is, its output is $\mu_i$. In other words, when $\text{Sup}_i$ cannot decide with certainty, $\text{Fuse}_\sigma$ receives $\mu_i$ instead of $(\text{ACC}^\mu_{i,\sigma}, \text{REJ}^\mu_{i,\sigma})$. Therefore, $\text{Fuse}_\sigma$ must compute $\text{ACC}^\mu_{i,\sigma}$ and $\text{REJ}^\mu_{i,\sigma}$, in order to determine its decision.

3) Third Variant: Each $\text{Sup}_i$ behaves always as a pure observer, that is, its output is always $\mu_i$. Therefore, in all situations, $\text{Fuse}_\sigma$ must compute $(\text{ACC}^\mu_{i,\sigma}, \text{REJ}^\mu_{i,\sigma})$ and take a decision on $\sigma$.

In comparison to our architecture, the drawbacks of each variant are as follows:

- **Variant 1**: The decisions are not well distributed between the supervisors and the fusion system. Indeed, $\text{Fuse}_\sigma$ takes decisions that can be made by the supervisors.
- **Variant 2**: The fusion system depends on $\mathcal{L}(G)$ and $\mathcal{K}$, and its computations are not trivial.
- **Variant 3** combines the drawbacks of variants 1 and 2.

IV. EXISTENCE RESULTS AND COMPARISON WITH OTHER ARCHITECTURES

Before we characterize the class of languages achievable with our architecture, let us recall the notion of controllability and define the new notions of $n$-observability and feasibility.

**Definition 4.1**: $\mathcal{K} \subseteq \mathcal{L}_m(G)$ is said to be controllable w.r.t $\mathcal{L}(G), \Sigma_{uc}$ iff $\mathcal{L}(G), \Sigma_{uc} \cap \mathcal{L}(G) \subseteq \mathcal{K}$.

**A. n-observability of $\mathcal{K}$**

**Definition 4.2**: $\mathcal{K} \subseteq \mathcal{L}_m(G)$ is said $n$-observable w.r.t $\mathcal{L}(G), \Sigma_{0,1}, \Sigma_{c,1}, \ldots, \Sigma_{0,n}, \Sigma_{c,n}$ iff $\forall \lambda, \mu \in \mathcal{K}, \forall \sigma \in \Sigma_c$ s.t. $(\lambda, \mu, \sigma \in \mathcal{L}(G)) \land (P_{\lambda}(\lambda) = P_{\mu}(\mu) \forall i \in \text{Ind}_\sigma)$, we have: $\lambda \sigma \in \mathcal{K} \iff \mu \sigma \in \mathcal{K}$.

Let us explain the intuition of $n$-observability of $\mathcal{K}$. We consider a controllable event $\sigma$ and two event sequences $\lambda$ and $\mu$ of $\mathcal{K}$ that are indistinguishable by the supervisors that control $\sigma$. Let us assume that the plant permits $\sigma$ to occur after both $\lambda$ and $\mu$. $n$-observability of $\mathcal{K}$ guarantees that $\mathcal{K}$: either permits $\sigma$ to occur after both $\lambda$ and $\mu$, or forbids $\sigma$ to occur after them.

**B. Feasibility of the Control System**

In Section III, we have explained how the global supervisor (consisting of the supervisors and the fusion modules) interacts with the plant with the objective to force it to conform to $\mathcal{K}$. This global supervisor will be denoted $\text{SUP}$.
Definition 4.3: \( \text{SUP} \) is said feasible w.r.t. \( \overline{K}, \mathcal{L}(G) \), if
\[
\Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \text{ s.t. for all } \sigma \in \Sigma_c, \forall \lambda \in \overline{K} \text{ such that } \lambda \sigma \in \mathcal{L}(G):
(\bigcap_{\sigma \in \text{Ind}_c} P_{\mathcal{K}}(\lambda) = \emptyset) \lor (\bigcap_{\sigma \in \text{Ind}_c} \mathcal{R}_{\lambda}(\lambda) = \emptyset).
\]

Intuitively, feasibility of \( \text{SUP} \) means that for every controllable event \( \sigma \), the situation \( \text{Any} \) (see Subsection III-B.2) never occurs when \( \sigma \) is accepted by \( \mathcal{L}(G) \). Recall that the situation \( \text{Any} \) occurs when \( \text{Fuse}_o \) has not enough information to take an enabling/disabling decision on \( \sigma \). The situation \( \text{Any} \) is a problem only if it occurs when \( \sigma \) is accepted by \( \mathcal{L}(G) \), because the decision of \( \text{Fuse}_o \) does not matter when \( \sigma \) is not accepted by \( \mathcal{L}(G) \).

When \( \text{SUP} \) is feasible, \( \text{SUP} \) can be seen as a function \( \text{SUP}: \mathcal{L}(G) \to 2^\mathcal{K} \) which associates to each sequence \( \lambda \in \mathcal{L}(G) \) the set of enabled events. Formally: \( \text{SUP}(\lambda) = \{ \sigma \in \Sigma_c | \text{Fuse}_\sigma(\prod_{\epsilon \in \text{Ind}_c} \text{Sup}_{\epsilon,\sigma}(\lambda)) = E_n \} \cup \Sigma_{uc} \).

C. Achievable Class of Languages

Let us present some propositions that characterize the class of languages achievable with our architecture. When \( \text{SUP} \) is feasible, let \( \mathcal{L}(\text{SUP}/G) \) denote the prefix-closed language generated by the plant under the control of \( \text{SUP} \), and let \( \mathcal{L}_m(\text{SUP}/G) = \mathcal{L}(\text{SUP}/G) \cap \mathcal{L}_m(G) \) be the corresponding marked language. \( \mathcal{L}(\text{SUP}/G) \) is formally defined as follows, where \( \varepsilon \) denotes the empty event sequence:

- \( \varepsilon \in \mathcal{L}(\text{SUP}/G) \)
- \( (\lambda \in \mathcal{L}(\text{SUP}/G)) \land (\lambda \sigma \in \mathcal{L}(G)) \land (\sigma \in \mathcal{SUP}(\lambda)) \land (\lambda \sigma \in \mathcal{SUP}(\lambda)) \leftrightarrow \lambda \sigma \in \mathcal{L}(\text{SUP}/G) \).

\( \text{SUP} \) is said non-blocking, \( \overline{K}_m(\text{SUP}/G) = \mathcal{L}(\text{SUP}/G) \).

Proposition 4.1: Consider a nonempty \( \mathcal{K} \subseteq \mathcal{L}_m(G) \). \( \text{SUP} \) satisfies C1 iff \( \mathcal{K} \) satisfies K1:

C1: \( \text{SUP} \) is feasible w.r.t. \( \overline{K}, \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \)
K1: \( \mathcal{K} \) is \( n \)-observable w.r.t. \( \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \).

Theorem 4.1: Consider a nonempty \( \mathcal{K} \subseteq \mathcal{L}_m(G) \). \( \text{SUP} \) satisfies C1 and C2 iff \( \mathcal{K} \) satisfies K0 and K1:

C1: \( \text{SUP} \) is feasible w.r.t. \( \overline{K}, \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \)
C2: \( \mathcal{L}(\text{SUP}/G) = \mathcal{K} \).
K0: \( \mathcal{K} \) is controllable w.r.t. \( \mathcal{L}(G) \) and \( \Sigma_{uc} \).
K1: \( \mathcal{K} \) is \( n \)-observable w.r.t. \( \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \).

Theorem 4.2: Consider a nonempty \( \mathcal{K} \subseteq \mathcal{L}_m(G) \). \( \text{SUP} \) satisfies C0 to C3 iff \( \mathcal{K} \) satisfies K0 to K2:

C0: \( \text{SUP} \) is non-blocking, i.e., \( \mathcal{L}_m(\text{SUP}/G) = \mathcal{L}(\text{SUP}/G) \).
C1: \( \text{SUP} \) is feasible w.r.t. \( \overline{K}, \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \)
C2: \( \mathcal{L}(\text{SUP}/G) = \mathcal{K} \).

D. Comparison with other architectures

- the inference-coobservability (or infer-coobservability) of the inference-based architecture of [14],
- the conditional-coobservability (or cond-coobservability) of the conditional architecture of [12],
- the coobservability (or gen-coobservability) of the general architecture of [11],
- the observability of the centralized architecture [15].

The four architectures of [14], [12], [11], [15] are compared to our new architecture by the following propositions:

Proposition 4.2: Infer-coobservability guarantees \( n \)-observability.

Proposition 4.3: Cond-coobservability guarantees \( n \)-observability.

Proposition 4.4: Gen-coobservability guarantees \( n \)-observability.

Proposition 4.5: \( n \)-observability guarantees observability.

Let us consider the following classes of languages:

- \( \mathcal{L}_{inf}(\mathcal{K}) = \{ G \subseteq \mathcal{K} : G \text{ is infer-coobservable w.r.t. } \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \} \)
- \( \mathcal{L}_{cond}(\mathcal{K}) = \{ G \subseteq \mathcal{K} : G \text{ is cond-coobservable w.r.t. } \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \} \)
- \( \mathcal{L}_{gen}(\mathcal{K}) = \{ G \subseteq \mathcal{K} : G \text{ is gen-coobservable w.r.t. } \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \} \)
- \( \mathcal{L}_{cond}(\mathcal{K}) = \{ G \subseteq \mathcal{K} : G \text{ is observable w.r.t. } \mathcal{L}(G), \Sigma_{o}, \Sigma_{c} \} \)
- \( \mathcal{L}_{new}(\mathcal{K}) = \{ G \subseteq \mathcal{K} : G \text{ is } n \text{-observable w.r.t. } \mathcal{L}(G), \Sigma_{o,1}, \Sigma_{c,1}, \ldots, \Sigma_{o,n}, \Sigma_{c,n} \} \).

In [12] it is proved that \( \mathcal{L}_{gen}(\mathcal{K}) \subseteq \mathcal{L}_{cond}(\mathcal{K}) \), and in [14] it is proved that \( \mathcal{L}_{cond}(\mathcal{K}) \subseteq \mathcal{L}_{inf}(\mathcal{K}) \). Therefore, using prop. 4.2 and 4.5, we deduce the following proposition:

Proposition 4.6: \( \mathcal{L}_{gen}(\mathcal{K}) \subseteq \mathcal{L}_{cond}(\mathcal{K}) \subseteq \mathcal{L}_{inf}(\mathcal{K}) \subseteq \mathcal{L}_{new}(\mathcal{K}) \subseteq \mathcal{L}_{cent}(\mathcal{K}) \).

Using Proposition 4.6, we can easily prove the following two propositions:

Proposition 4.7: The class of languages achievable with our new architecture strictly includes the classes achievable with the other decentralized (general, conditional, inference-based) architectures.

Proposition 4.8: The class of languages achievable with the centralized architecture strictly includes the class achievable with our new architecture.

E. Examples of Comparison

We will illustrate \( \mathcal{L}_{inf}(\mathcal{K}) \subseteq \mathcal{L}_{new}(\mathcal{K}) \) by two examples of \( \mathcal{K} \in \mathcal{L}_{new}(\mathcal{K}) \) \( \mathcal{L}_{inf}(\mathcal{K}) \). We will also illustrate \( \mathcal{L}_{new}(\mathcal{K}) \subseteq \mathcal{L}_{cent}(\mathcal{K}) \) by one example of \( \mathcal{K} \in \mathcal{L}_{cent}(\mathcal{K}) \) \( \mathcal{L}_{new}(\mathcal{K}) \).

The other inequalities \( \mathcal{L}_{gen}(\mathcal{K}) \subseteq \mathcal{L}_{cond}(\mathcal{K}) \) and \( \mathcal{L}_{cond}(\mathcal{K}) \subseteq \mathcal{L}_{inf}(\mathcal{K}) \) are illustrated in [12] and [14], respectively. For simplicity, we will consider prefix-closed languages, that is, all states are implicitly marked.
1) First Example Illustrating $\mathcal{K} \in \mathcal{L}_{\text{new}}(\mathcal{K}) \setminus \mathcal{L}_{\text{inf}}(\mathcal{K})$:

Let us consider the plant of Figure 2 defined over the alphabet $\{a_1, a_2, c\}$, where $\Sigma_{c,1} = \Sigma_{c,2} = \{c\}, \Sigma_{a,1} = \{a_1\}, \Sigma_{a,2} = \{a_2\}$. The specification $\mathcal{K}$ is obtained from the plant by forbidding $c$ in the three states represented by a square, that is, by removing states 5, 9 and 10. Let us show that this specification is $n$-observable with our architecture. The events $a_1$ and $a_2$ are enabled in all situations, and for the event $c$ the control system proceeds as follows:

- When both $\text{Sup}_1$ and $\text{Sup}_2$ have observed $c$: 
  - $\text{Sup}_1$ computes $\text{REJ}_{1,c} = \{c\}$ and $\text{ACC}_{1,c} = \{a_2c\}$,
  - $\text{Sup}_2$ computes $\text{REJ}_{2,c} = \{c\}$ and $\text{ACC}_{2,c} = \{a_1c\}$,
  - $\text{Fuse}_c$ deduces $\text{REJ}_{1,c} \cap \text{REJ}_{2,c} = \{c\}$ and $\text{ACC}_{1,c} \cap \text{ACC}_{2,c} = \emptyset$. Therefore, $c$ is disabled.

- When $\text{Sup}_1$ and $\text{Sup}_2$ have observed $a_1$ and $a_2$ resp.:
  - $\text{Sup}_1$ computes $\text{REJ}_{1,c} = \{a_1c, a_2c\}$ and $\text{ACC}_{1,c} = \{a_1c\}$,
  - $\text{Sup}_2$ computes $\text{REJ}_{2,c} = \{a_1c, a_2c\}$ and $\text{ACC}_{2,c} = \{a_2c\}$,
  - $\text{Fuse}_c$ deduces $\text{REJ}_{1,c} \cap \text{REJ}_{2,c} = \{a_1c, a_2c\}$ and $\text{ACC}_{1,c} \cap \text{ACC}_{2,c} = \emptyset$. Therefore, $c$ is disabled.

2) Second Example Illustrating $\mathcal{K} \in \mathcal{L}_{\text{new}}(\mathcal{K}) \setminus \mathcal{L}_{\text{inf}}(\mathcal{K})$:

For our example, $E_0(\mathcal{K}, c) = \{a_1, a_2\} \neq \emptyset$ and $D_0(\mathcal{K}, c) = \{\varepsilon, a_1c, a_2, a_2c\}$. It can be shown that:

- $D_0(\mathcal{K}, c) \subseteq \bigcap_{i \in \text{Ind}_a} P_i^{-1} P_i(E_0(\mathcal{K}, c)),
- E_0(\mathcal{K}, c) \cap \bigcup_{i \in \text{Ind}_a} P_i^{-1} P_i(D_0(\mathcal{K}, c)).$

Therefore:

- $D_1(\mathcal{K}, c) = D_0(\mathcal{K}, c) \cap \bigcup_{i \in \text{Ind}_a} P_i^{-1} P_i(E_0(\mathcal{K}, c)) = D_0(\mathcal{K}, c),
- E_1(\mathcal{K}, c) = E_0(\mathcal{K}, c) \cap \bigcup_{i \in \text{Ind}_a} P_i^{-1} P_i(D_0(\mathcal{K}, c)) = E_0(\mathcal{K}, c).$

By induction, we deduce $\forall k > 0$:

$(D_k(\mathcal{K}, c) = D_0(\mathcal{K}, c) \neq \emptyset) \land (E_k(\mathcal{K}, c) = D_0(\mathcal{K}, c) \neq \emptyset),$

which means that $\mathcal{K}$ is not infer-coobservable.

2) Second Example Illustrating $\mathcal{K} \in \mathcal{L}_{\text{new}}(\mathcal{K}) \setminus \mathcal{L}_{\text{inf}}(\mathcal{K})$:

We consider the plant of Figure 3 defined over the alphabet $\{a_1, a_1, a_2, b_1, b_2\}$. We take $\Sigma_{a,1} = \{a_1, a_2\}, \Sigma_{a,2} = \{a_1, a_2\}, \Sigma_{b,1} = \Sigma_{a,2} = \{c\}$. The specification $\mathcal{K}$ is obtained from the plant by forbidding $c$ in the two states represented by a square, that is by removing the dotted transitions and states. Let us show that this specification is $n$-observable with our architecture. The events $a_1, b_1, a_2, b_2$ are enabled in all situations, and for the event $c$ the control system proceeds as follows:

- $\text{Fuse}_c$ enables $c$ in the following two situations:
  - $\text{Sup}_1$ and $\text{Sup}_2$ have observed the sequences $a_1 b_1$ and $a_2b_2$, respectively;
  - $\text{Sup}_1$ and $\text{Sup}_2$ have observed the sequences $b_1 a_1$ and $b_2a_2$, respectively.

Indeed, the reader can check that $\text{ACC}_{1,c} \cap \text{ACC}_{2,c} = \emptyset$ in the above two situations.

3) Example illustrating $\mathcal{K} \in \mathcal{L}_{\text{cent}}(\mathcal{K}) \setminus \mathcal{L}_{\text{new}}(\mathcal{K})$:

Note that a similar example is used in [14] for illustrating $\mathcal{K} \in \mathcal{L}_{\text{cent}}(\mathcal{K}) \setminus \mathcal{L}_{\text{inf}}(\mathcal{K})$. We consider the plant $\mathcal{L}_m(G)$ of Figure 4 defined over the alphabet $\{a_1, a_2, c\}$. We take
\[ \Sigma_{o,1} = \{ a_1 \}, \Sigma_{o,2} = \{ a_2 \}, \Sigma_c = \Sigma_{c,1} = \Sigma_{c,2} = \{ c \}. \] The specification \( K \) is obtained from the plant by forbidding \( c \) in the state represented by a square. That is, \( c \) is enabled (resp. disabled) after the execution of the sequence \( a_1 a_2 \) (resp. \( a_2 a_1 \)). In a centralized architecture, the only unobservable event is \( c \), and thus, the sequences \( a_1 a_2 \) and \( a_2 a_1 \) are distinguishable. Therefore, \( K \) is observable.

Let us show that this specification is not \( n \)-observable with our architecture. When \( \text{Sup}_1 \) and \( \text{Sup}_2 \) have observed \( a_1 \) and \( a_2 \) respectively:

- The outputs of \( \text{Sup}_1 \) are \( \text{REJ}^{a_1}_{1,c} = \{ a_2 a_1 c \} \) and \( \text{ACC}^{a_2}_{1,c} = \{ a_1 a_2 c \} \).
- The outputs of \( \text{Sup}_2 \) are \( \text{REJ}^{a_2}_{2,c} = \{ a_2 a_1 c \} \) and \( \text{ACC}^{a_1}_{2,c} = \{ a_1 a_2 c \} \).
- \( \text{Fuse}_c \) computes \( \text{REJ}^{a_1}_{1,c} \cap \text{REJ}^{a_2}_{2,c} = \{ a_2 a_1 c \} \) and \( \text{ACC}^{a_1}_{1,c} \cap \text{ACC}^{a_2}_{2,c} = \{ a_1 a_2 c \} \).

The control system is not feasible because both intersections are not empty (see Def. 4.3). Therefore, the specification is not \( n \)-observable (from Prop. 4.1). Intuitively, distinct decisions are to be taken in indistinguishable situations:

- \( c \) must be enabled after the execution of \( a_1 a_2 \), i.e., when \( \text{Sup}_1 \) and \( \text{Sup}_2 \) have observed \( a_1 \) and \( a_2 \), respectively.
- \( c \) must be disabled after the execution of \( a_2 a_1 \), i.e., when \( \text{Sup}_1 \) and \( \text{Sup}_2 \) have observed \( a_1 \) and \( a_2 \), respectively.

![Fig. 4. Example Illustrating \( K \in \mathcal{L}_{\text{cent}}(K) \setminus \mathcal{L}_{\text{new}}(K) \)]

V. PROPERTIES OF \( n \)-OBSERVABILITY

Similarly to the *-coobservabilities of the previous architectures (i.e., general, conditional, inference based), \( n \)-observability satisfies the following two propositions:

**Proposition 5.1:** \( n \)-observability is not necessarily preserved under union of languages.

**Proposition 5.2:** \( n \)-observability is preserved under intersection of languages when the latter are prefix-closed.

Proposition 5.2 implies that when \( K \) is not \( n \)-observable, the infimal \( n \)-observable language containing \( K \) exists when the languages considered are prefix-closed.

Proposition 5.1 implies that when \( K \) is not \( n \)-observable, the supremal \( n \)-observable language included in \( K \) does not necessarily exist. A solution would be to find a property, let us call it \( n \)-normality, which is:

1) preserved under union of languages,
2) stronger than \( n \)-observability,
3) weaker than the inter-coobservability, and thus characterizes a class of languages which includes those of the previous decentralized architectures.

In this case, if \( K \) is not achievable (i.e., is not at the same time controllable, \( n \)-observable and \( \mathcal{L}_m(G) \)-closed), then the aim will be to synthesize the supremal \( M \subseteq K \) which is controllable, \( n \)-normal and \( \mathcal{L}_m(G) \)-closed.

In [16], a joint-observability has been defined and shown to be undecidable. Although joint-observability seems quite similar to our \( n \)-observability, it can be easily proved that the two notions are distinct. Indeed, unlike \( n \)-observability, joint-observability is not based on the controllability of events. Moreover, we have the following proposition:

**Proposition 5.3:** Joint-observability of [16] is stronger than \( n \)-observability.

VI. CONCLUSION

In this paper, we proposed a new architecture for decentralized control of DES whose principle is as follows: when an event \( \sigma \) is controllable by several supervisors, each of the latter takes an enabling/disabling unconditional decision if and only if it is sure that this is the right decision which can be applied to the plant. Otherwise, the supervisor transmits its local information to the fusion system. Consequently, when no supervisor can take a decision, the fusion system combines the elements of information transmitted by the supervisors in order to synthesize a decision which is applied to the plant.

We have defined the property of \( n \)-observability, which characterizes the class of languages which are achievable with our architecture. Like the *-coobservability properties of the previous architectures, our \( n \)-observability is not preserved under union of languages, and thus the supremal \( n \)-observable language which is included in \( K \) does not exist in general.

The main advantage of our new architecture is that \( n \)-observability characterizes a class of languages which strictly includes those of the previous decentralized architectures. And although the fusion system does more work than in prior decentralized architectures, we show that it is independent of the plant and the specification.

But note that the proposed approach manipulates languages \( \text{ACC}^{a_1}_{1,\sigma} \) and \( \text{REJ}^{a_2}_{2,\sigma} \) which are infinite. In order to make the approach applicable, these languages need to be converted efficiently into a finite representation. The efficiency constraint makes the problem not trivial and we are presently working on this aspect.

REFERENCES


VII. PROOFS

A. Proof of Proposition 4.1

Proof of feasibility of $\mathcal{SUP} \Rightarrow n$-observability of $\mathcal{K}$:

We consider a control system $\mathcal{SUP}$ which is feasible. Let $\sigma \in \Sigma_c$, $\lambda, \mu \in \overline{\mathcal{K}}$ such that $\lambda \sigma \in \mathcal{K}$ and $\mu \sigma \in \mathcal{L}(G) \setminus \mathcal{K}$.

1) From feasibility of $\mathcal{SUP}$:

$1a$: $(\bigcap_{i \in \text{Ind}_\sigma} P^{-1}_i(\lambda \cap \mathcal{K}) \cap (\mathcal{L}(G) \setminus \mathcal{K}) = \emptyset$,

$1b$: $(\bigcap_{i \in \text{Ind}_\sigma} P^{-1}_i(\mu) \sigma \cap \mathcal{K}) = \emptyset$.

2) From Item 1:

From $1a$: $\forall s \in \mathcal{K}$ such that $s \sigma \in \mathcal{L}(G) \setminus \mathcal{K}$:

$\exists \sigma \in \text{Ind}_\sigma$ s.t. $P_i(\lambda) \neq P_i(s)$.

From $1b$: $\forall t \sigma \in \mathcal{K}$ such that $t \sigma \in \mathcal{K}$:

$\exists \tau \in \text{Ind}_\sigma$ s.t. $P_i(\mu) \neq P_i(\lambda)$.

3) From Def. 4.2 and Item 3, we deduce $n$-observability of $\mathcal{K}$.

Proof of $n$-observability of $\mathcal{K} \Rightarrow$ feasibility of $\mathcal{SUP}$:

Equivalently, let us prove:

$\mathcal{SUP}$ unfeasible $\Rightarrow \mathcal{K}$ not $n$-observable.

Let $\sigma \in \Sigma_c$.

1) From unfeasibility of $\mathcal{SUP}$ (using Def. 4.3): there exists $\lambda \in \overline{\mathcal{K}}$ such that:

$(\bigcap_{i \in \text{Ind}_\sigma} \overline{\text{ACC}}^P_i(\lambda) \neq \emptyset) \land (\bigcap_{i \in \text{Ind}_\sigma} \overline{\text{REJ}}^P_i(\lambda) \neq \emptyset)$.

2) Item 1 implies that there exists $\alpha, \beta \in \overline{\mathcal{K}}$ such that:

$\alpha \sigma \in \overline{\mathcal{K}}$, $\beta \sigma \in \mathcal{L}(G) \setminus \mathcal{K}$ and $P_i(\alpha) = P_i(\beta) = P_i(\lambda)$ for all $i \in \text{Ind}_\sigma$.

3) From Def. 4.2 and Item 2: $\mathcal{K}$ is not $n$-observable.

B. Proof of Theorem 4.1: $(C1 \land C2) \Rightarrow (K0 \land K1)$

Proof of $(K0 \land K1) \Rightarrow (C1 \land C2)$:

We assume $K0$ and $K1$. From Prop. 4.1 and $K1$, we deduce $C1$: $\mathcal{SUP}$ is feasible, and thus, the function $\mathcal{SUP}$ and the language $\mathcal{L}(\mathcal{SUP}/G)$ are defined. It remains to prove $C2$: $\mathcal{L}(\mathcal{SUP}/G) = \overline{\mathcal{K}}$.

Proof of $\mathcal{L}(\mathcal{SUP}/G) \subseteq \overline{\mathcal{K}}$:

1) Let us consider $\lambda \sigma \in \mathcal{L}(\mathcal{SUP}/G)$, i.e., $\lambda \in \mathcal{L}(\mathcal{SUP}/G)$ and $\sigma \in \mathcal{L}(G)$.

We assume inductively $\lambda \in \overline{\mathcal{K}}$.

In the following items 2 and 3, we consider the cases $\sigma \in \Sigma_{uc}$ and $\sigma \in \Sigma_c$, respectively.

2) $\sigma \in \Sigma_{uc}$, $\lambda \in \overline{\mathcal{K}}$, $\lambda \sigma \in \mathcal{L}(G)$ imply $\lambda \sigma \in \overline{\mathcal{K}} \cap \mathcal{L}(G)$. From the controllability of $\mathcal{K}$, we deduce $\lambda \sigma \in \overline{\mathcal{K}}$.

3) $\sigma \in \Sigma_c$, $\sigma \in \mathcal{SUP}(\lambda)$, $\lambda \in \overline{\mathcal{K}}$ imply $\lambda \sigma \in \overline{\mathcal{K}}$.

4) From Items 1, 2 and 3: $(\lambda \sigma \in \mathcal{L}(\mathcal{SUP}/G)) \Rightarrow (\lambda \sigma \in \overline{\mathcal{K}})$, i.e., $\mathcal{L}(\mathcal{SUP}/G) \subseteq \overline{\mathcal{K}}$.

Proof of $\overline{\mathcal{K}} \subseteq \mathcal{L}(\mathcal{SUP}/G)$:

1) Let us consider $\lambda \sigma \in \overline{\mathcal{K}}$ and thus $\lambda \sigma \in \mathcal{L}(G)$.

We assume inductively $\lambda \in \mathcal{L}(\mathcal{SUP}/G)$. In the following items 2 and 3, we consider the cases $\sigma \in \Sigma_{uc}$ and $\sigma \in \Sigma_c$, respectively.

2) $\sigma \in \Sigma_{uc}$ (and thus, $\sigma \in \mathcal{SUP}(\lambda)$) and $\lambda \in \mathcal{L}(\mathcal{SUP}/G)$ imply $\lambda \sigma \in \mathcal{L}(\mathcal{SUP}/G)$.

3) $\sigma \in \Sigma_c$ and $\lambda \sigma \in \overline{\mathcal{K}}$ imply $\sigma \in \mathcal{SUP}(\lambda)$, and thus, $\lambda \sigma \in \mathcal{L}(\mathcal{SUP}/G)$.
Proof of (C1 \land C2) \Rightarrow (K0 \land K1):

We assume C1 and C2. From Prop. 4.1 and C1, we deduce that K1: \( K \) is \( n \)-observable. It remains to prove K0: \( K \) is controllable.

1) Let us consider \( \lambda \sigma \in \Sigma_{uc} \cap \L(G) \), i.e., \( \lambda \in \K, \sigma \in \Sigma_{uc} \) and \( \lambda \sigma \in \L(G) \).
2) \( \lambda \in \K \) (Item 1) and \( \L(SUP/G) = \K \) (C2) imply:
   \( \lambda \in \L(SUP/G) \).
3) \( \sigma \in \Sigma_{uc} \) (Item 1) implies: \( \sigma \in SUP(\lambda) \).
4) Items 2 and 3 imply: \( \lambda \sigma \in \L(SUP/G) \). From C2, we deduce \( \lambda \sigma \in \K \).
5) From Items 1 and 4: \( (\lambda \sigma \in \Sigma_{uc} \cap \L(G)) \Rightarrow (\lambda \sigma \in \K) \), i.e., \( \K_{uc} \cap \L(G) \subseteq \K \), which means that \( K \) is controllable.

C. Proof of Theorem 4.2: (C0 \land C1 \land C2 \land C3) \Rightarrow (K0 \land K1 \land K2)

Proof of (K0 \land K1 \land K2) \Rightarrow (C0 \land C1 \land C2 \land C3):

We assume K0, K1 and K2. From Theorem 4.1 and (K0 \land K1), we deduce that \( SUP \) is feasible (C1) and \( \L(SUP/G) = \K \) (C2). It remains to prove \( \L_m(SUP/G) = \L(SUP/G) \) (C0) and \( \L_m(SUP/G) = \K \) (C3).

Proof of C3: \( \L_m(SUP/G) = \K \):
1) By definition \( \L_m(SUP/G) = \L(SUP/G) \cap \L_m(G) \).
2) \( \L(SUP/G) = \K \) (C2) and Item 1 imply:
   \( \L_m(SUP/G) = \K \cap \L_m(G) \).
3) \( \K = \K \cap \L_m(G) \) (K2) and Item 2 imply:
   \( \L_m(SUP/G) = \K \).

Proof of C0: \( \L_m(SUP/G) = \L(SUP/G) \):
1) From C3, \( \L_m(SUP/G) = \K \).
2) C2 and Item 1 imply: \( \L_m(SUP/G) = \L(SUP/G) \).

Proof of: (C0 \land C1 \land C2 \land C3) \Rightarrow (K0 \land K1 \land K2):

We assume C0, C1, C2 and C3. From Theorem 4.1 and (C1 \land C2), we deduce K0 and K1. It remains to prove K2: \( \K = \K \cap \L_m(G) \).
1) By definition: \( \L_m(SUP/G) = \L(SUP/G) \cap \L_m(G) \).
2) C2, C3 and Item 1 imply: \( \K = \K \cap \L_m(G) \).

D. Proof of Proposition 4.2

We will prove the following equivalent proposition:

if \( K \) is not \( n \)-observable, then it is not infer-coobservable.

In this proof, we will use the functions \( E_k(K, \sigma) \) and \( D_k(K, \sigma) \) of [14] also defined in subsection IV-E.1.

Let us consider \( K \) which is not \( n \)-observable.

1) From Def. 4.2: \( \exists s, t \in \K, \exists \sigma \in \Sigma_n \) such that: \( (s \sigma \in \K) \land (t \sigma \in \L(G) \setminus \K) \land (v \in Ind_\sigma \land P_i(s) = P_i(t)) \).

2) From (\forall i \in Ind_\sigma: P_i(s) = P_i(t)) of Item 1, we obtain:
   - \( E_k(K, \sigma) \cap (\bigcap_{i \in Ind_\sigma} P_i^{-1} P_i(D_k(K, \sigma))) = E_{k+1}(K, \sigma) \)
   - \( D_k(K, \sigma) \cap (\bigcap_{i \in Ind_\sigma} P_i^{-1} P_i(E_k(K, \sigma))) = D_{k+1}(K, \sigma) \)

3) For the \( s \) and \( t \) of Item 1, we have \( s \in E_0(K, \sigma) \) and \( t \in D_0(K, \sigma) \).

Suppose \( s \in E_k(K, \sigma) \) and \( t \in D_k(K, \sigma) \) for some \( k \geq 0 \). From Item 2, it follows that: \( s \in E_{k+1}(K, \sigma) \) and \( t \in D_{k+1}(K, \sigma) \).

4) Item 3 implies that:
   \( \forall k \geq 0, (s \in E_k(K, \sigma)) \land (t \in D_k(K, \sigma)) \), and thus:
   \( \forall k \geq 0, (E_k(K, \sigma) \neq \emptyset) \land (D_k(K, \sigma) \neq \emptyset) \).

Item 4 means that \( K \) is not infer-coobservable.

E. Proof of Proposition 4.3

1) Theorem 4 of [14] proves that cond-coobservability guarantees infer-coobservability.

2) Prop. 4.2 proves that infer-coobservability guarantees \( n \)-observability.

From Items 1 and 2, we deduce that cond-coobservability guarantees \( n \)-observability.

F. Proof of Proposition 4.4

1) Theorem 3 of [14] proves that gen-coobservability guarantees infer-coobservability.

2) Prop. 4.2 proves that infer-coobservability guarantees \( n \)-observability.

From Items 1 and 2, we deduce that gen-coobservability guarantees \( n \)-observability.

G. Proof of Proposition 4.5

1) From Def. 4.2, \( n \)-observability implies:
   \( \forall s, t \in \K, \forall \sigma \in \Sigma_n \) such that: \( (s \sigma \in \K) \land (t \sigma \in \L(G) \setminus \K) \land (v \in Ind_\sigma \land P_i(s) \neq P_i(t)) \).

2) \( P_i(s) \neq P_i(t) \) of Item 1 implies \( P(s) \neq P(t) \), where \( P \) denotes the projection on \( \Sigma_n = \bigcup_{i=1, \ldots, n} \Sigma_{i,o} \).

3) Items 1 and 2 imply:
   \( \forall s, t \in \K, \forall \sigma \in \Sigma_n \) such that: \( (s \sigma \in \K) \land (t \sigma \in \L(G) \setminus \K) \land (P(s) \neq P(t)) \), which is the definition of observability in the centralized architecture.

H. Proof of Proposition 4.6

1) \( \L_{gen}(K) \subseteq \L_{cond}(K) \) is proved in [12].

2) \( \L_{cond}(K) \subseteq \L_{inf}(K) \) is proved in [14].

3) \( \L_{inf}(K) \subseteq \L_{new}(K) \) can be deduced from Prop. 4.2.

4) \( \L_{new}(K) \subseteq \L_{cent}(K) \) can be deduced from Prop. 4.5.

I. Proofs of Propositions 4.7 and 4.8

[11], [12], [14] and [15] contain theorems similar to our Theorem 4.2, but are related to the general, the conditional, the inference-based and the centralized architectures, respectively. Using Proposition 4.6, we deduce:

1) The class of languages achievable with our new architecture strictly includes the class achievable with the inference-based architecture.

2) The class of languages achievable with the inference-based architecture strictly includes the class achievable with the conditional architecture.

3) The class of languages achievable with the conditional architecture strictly includes the class achievable with the general architecture.

4) The class of languages achievable with the centralized architecture strictly includes the class achievable with our new architecture.
J. Proof of Proposition 5.1

Let \( K_1 \) and \( K_2 \) be two \( n \)-observable languages w.r.t \( L(G), \Sigma_{o,1}, \Sigma_{c,1}, \cdots, \Sigma_{o,n}, \Sigma_{c,n} \). For simplicity, let us assume that \( K_1, K_2, L_m(G) \) are prefix-closed.

1) From Def. 4.2, for \( p = 1, 2 \): \( \forall s, t \in K_p, \)
\[ \forall \sigma \in \Sigma_c = \bigcup_{i=1}^n \Sigma_{c,i} \text{ s.t. } (s \sigma \in K_p) \land (t \sigma \in L(G) \setminus K_p); \exists \lambda \in \text{Ind}_\sigma \text{ s.t. } P_i(\lambda) \neq P_i(t). \]

2) Let \( \sigma \in \Sigma_c \) and \( s, t \in K_1 \cap K_2 \) such that
\[ (s \sigma \in K_1 \cup K_2) \land (t \sigma \in L(G) \setminus K_1 \cup K_2). \]
From the prefix closure of \( K_1, K_2, L_m(G) \), we deduce:
- \( s \sigma \in K_1 \) or \( s \sigma \in K_2 \), and
- \( t \sigma \in L(G) \setminus K_1 \) or \( t \sigma \in L(G) \setminus K_2 \).
If \( s, t \in K_1 \) or \( s, t \in K_2 \), we can apply Item 1 to deduce \( \exists i \in \text{Ind}_\sigma \text{ s.t. } P_i(s) \neq P_i(t). \)
But Item 1 is of no help when \( s \in K_1 \setminus K_2 \) and \( t \in K_2 \setminus K_1 \).

Therefore, \( K_1 \cup K_2 \) is not \( n \)-observable in general.

K. Proof of Proposition 5.2

Let \( K_1 \) and \( K_2 \) be two \( n \)-observable languages w.r.t \( L(G), \Sigma_{o,1}, \Sigma_{c,1}, \cdots, \Sigma_{o,n}, \Sigma_{c,n} \). We assume \( K_1, K_2, L_m(G) \) prefix-closed.

1) The \( n \)-observability of \( K_p \) (\( p = 1, 2 \)) implies: \( \forall s, t \in K_p, \forall \sigma \in \Sigma_c = \bigcup_{i=1}^n \Sigma_{c,i} \text{ s.t. } (s \sigma \in K_p) \land (t \sigma \in L(G) \setminus K_p); \exists i \in \text{Ind}_\sigma \text{ s.t. } P_i(s) \neq P_i(t). \)

2) Let \( \sigma \in \Sigma_c \) and \( s, t \in K_1 \cap K_2 \) such that
\[ (s \sigma \in K_1 \cup K_2) \land (t \sigma \in L(G) \setminus K_1 \cap K_2). \]
From the prefix closure of \( K_1, K_2, L_m(G) \), we deduce:
- \( s \sigma \in K_1 \) and \( s \sigma \in K_2 \), and
- \( t \sigma \in L(G) \setminus K_1 \) or \( t \sigma \in L(G) \setminus K_2 \).

We can apply Item 1 to any of the \( K_p \) (\( p = 1, 2 \)) for which \( t \sigma \in L(G) \setminus K_p \) to deduce that:
\[ \exists i \in \text{Ind}_\sigma \text{ s.t. } P_i(s) \neq P_i(t). \]

Therefore, \( K_1 \cap K_2 \) is \( n \)-observable when \( K_1, K_2, L_m(G) \) are prefix-closed.

L. Proof of Proposition 5.3

1) \( K \subseteq L_m(G) \) is \( n \)-observable w.r.t \( L(G), \Sigma_{o,1}, \Sigma_{c,1}, \cdots, \Sigma_{o,n}, \Sigma_{c,n} \), iff \( \forall \sigma \in \Sigma_c, \forall \lambda, \mu \in \overline{\Sigma}: (\lambda \sigma \in \overline{\Sigma}) \land (\mu \sigma \in L(G) \setminus \overline{\Sigma}) \Rightarrow \exists i \in \text{Ind}_\sigma \text{ s.t. } P_i(\lambda) \neq P_i(\mu). \)

2) \( K \subseteq L_m(G) \) is joint-observable w.r.t \( L(G), \Sigma_{o,1} \cdots , \Sigma_{o,n} \), iff \( \forall \phi, \psi: (\phi \in \overline{\Sigma}) \land (\psi \in L(G) \setminus \overline{\Sigma}) \Rightarrow \exists i \in \text{Ind}_\sigma \text{ s.t. } P_i(\phi) \neq P_i(\psi). \)

We see that Item 1 is obtained from Item 2 by taking \( \phi = \lambda \sigma \) and \( \psi = \mu \sigma \), and thus, we have the following constraints:
- \( \phi \) and \( \psi \) terminate by a same controllable event.
- \( \mu \in \overline{\Sigma}. \)

Therefore, joint-observability is defined in a bigger language (and thus, is stronger) than \( n \)-observability.