An Efficient Modular Method for the Control of Concurrent Discrete Event Systems: A Language-Based Approach

Benoit Gaudin, Hervé Marchand

An Efficient Modular Method for the Control of Concurrent Discrete Event Systems: A Language-Based Approach*

Benoit Gaudin†, Hervé Marchand‡

March 6, 2007

Abstract

In this paper, we are interested in the control of a particular class of Concurrent Discrete Event Systems defined by a collection of components that interact with each other. We investigate the computation of the supremal controllable language contained in the language of the specification. We do not adopt the decentralized approach. Instead, we have chosen to use a modular centralized approach and to perform the control on some approximations of the plant derived from the behavior of each component. The behavior of these approximations is restricted so that they respect a new language property for discrete event systems called *partial controllability condition* that depends on the specification. It is shown that, under some assumptions, the intersection of these "controlled approximations" corresponds to the supremal controllable language contained in the specification with respect to the plant. This computation is performed without having to build the whole plant, hence avoiding the state space explosion induced by the concurrent nature of the plant. It is finally shown that the class of specifications on which our method can be applied strictly subsumes the class of separable specifications.

1 Introduction

Supervisory control ([27, 5]) consists in modifying a *system* (plant) such that the modified (or *controlled*) system satisfies a given *specification* (e.g. a *safety property*). In this paper, we focus on the control of Concurrent Discrete Event Systems defined by a collection of components that interact with each other. We adopt the formalism of the Supervisory Control Theory (SCT) [27] and model the system and the specification by regular languages (or by the associated Finite State Machines). Given a system and a specification modeled by languages, one important phase in the SCT is the computation of the *supremal controllable sub-language contained in the specification* from which one can easily derive a supervisor.

In many applications (as e.g. manufacturing systems, control-command systems, protocol networks, etc) and control problems, systems are also often modeled by several components acting concurrently. In this paper, we are concerned with the control of a system where its construction is assumed not to be feasible (due to the state space explosion resulting from the composition), making the use of classical supervisory control methodologies impractical (See e.g. [27] or [5]).

*Corresponding author: H. Marchand. E-Mail: herve.marchand@irisa.fr, Fax : +33 2 99847171

†UCD, campus de Belfield, Dublin 4, Ireland

‡IRISA-Inria Rennes, Campus Univ. de Beaulieu, 35042 Rennes, France

Several approaches have been recently investigated to deal with the complexity issue of the control of Concurrent Discrete Event Systems (CDES). Given a CDES G modeled by a set of sub-systems $\{G_i\}_{1\leq i\leq n}$ and a specification expressed by a language K, the problem under consideration is to compute the supremal controllable sublanguage of $K \cap L(G)$ w.r.t. $L(G)$ without having to explicitly build $\L(G)$. One classical approach consists in using Binary Decision Diagrams (BDD) [4] to encode the system ([15, 14, 21, 20, 25]) and to use transformer predicates to perform all the operations. Even though the whole system needs to be built, such encoding is efficient in the sense that it avoids the state space enumeration during the synthesis phase. Meanwhile, it is still interesting to combine these symbolic approaches with methods that are independent of the implementation. These methods should be based on the structure of a CDES and have to be efficient even though the tool, that is used to perform the supervisor synthesis, is based on enumerative methods. In [8, 7], the authors consider the control of a product plant (i.e. systems composed of asynchronous subsystems, not sharing common events)[1]. Given a set of specifications, for each of them, a local sub-system is built from the components that are coordinated by the corresponding specification (i.e. all the components that share some events used to express it). It is then sufficient to compute local supervisors ensuring each specification with respect to the corresponding local sub-system in order to obtain the result on the whole system. An incremental and modular approach have been presented in [3, 2]. When several specifications are under considerations, the global supervisor can be obtained by performing the parallel composition between the different corresponding "local" supervisors. Finally, closely related to the decentralized theory[2], under the hypothesis that the specification is *separable* (See Section 3.1.1) and that the shared events are controllable, the authors of [26] provide a solution allowing to compute local supervisors S_i acting upon each sub-system G_i and to operate the individually controlled system S_i/G_i concurrently in such a way that the behavior of the controlled system (i.e. $L(\|_i\ S_i/G_i)$) corresponds to the supremal controllable sublanguage of K w.r.t. the system $L(G)$. Note that each supervisor is efficiently computed since it is derived from each sub-system thus avoiding to build the whole system. The same methodology has been used in [23] for the control of concurrent systems for which the various components are supposed to have an identical structure. Knowing that the local supervisors S_i are only operating on a subset of the local events, the authors give necessary and sufficient conditions over the specification K, to obtain a non-blocking controlled system that exactly matches K. Finally note that the a decentralized architecture could be used to efficiently implement the supervisors computed according to the theory of [8, 7]. See also [17, 19] and [20, 1, 12, 18] for other works related to the control of concurrent systems and hierarchical systems.

In this paper, our motivation is similar as that of the previous works. We want to use the particular structure of a concurrent system in order to avoid the building of the whole system. In this study, we have chosen not to consider the non-blocking aspect, but rather to focus on the computation of supervisors for prefix-closed specifications with the aim of extending the class of specifications for which a maximal controllable solution exists. Indeed, besides the non-blocking aspect, that is tackled by some of the above works, one common condition that is required is the separability of the specification (as e.g. in [26]) or of the solution (as e.g. in [23, 17]). However, requesting the specification [26] or the solution [23, 17] to be separable happens to be a quite restrictive

[1]Given a CDES G modeled by $\{G_i\}_{1\leq i\leq n}$, the authors actually collapse the sub-systems in order to obtain a product plant.

[2]see e.g. [6, 24, 29] for details related to this theory.

condition. Indeed, this condition does not permit to model a particular behavioral interleaving between different components or a particular scheduling of actions that belong to different components (for example, it is not possible to specify that one subsystem G_1 can only trigger a local action a_1 when one other subsystem G_2 already triggered a local action a_2). So, our aim was to find another methodology for which the separability condition is not required. To do so, we have chosen not to adopt a "*decentralized*" approach as in [8, 26, 23, 17]. Instead of having one local supervisor per sub-system (or a set of sub-systems [8]) that enforces local control actions with respect to the events of this component (resp. set of components), we perform the control on some approximations of the system, each of them derived from the behavior of one component. The behavior of these approximations is restricted so that they respect a new language property for discrete event systems called *partial controllability condition* that depends on the specification. It is shown that, a supervisor can be efficiently derived from these "controlled approximations" such that the behavior of the controlled system is actually controllable with respect to the plant and the uncontrollable events, even though the specification is not separable. At this point we only require that the components that share an event agree on the control status of this event. However, it is generally the case, that the obtained solution is not the most permissive one. We thus give some new conditions under which the behavior of the obtained controlled system corresponds to the supremal controllable language contained in the specification with respect to the system. One condition concerns the system itself and requires the shared events to be controllable (as in [26]), the other one concerns the specification and is called local consistency (it is shown to be strictly less restrictive than separability condition).

The rest of the paper is organized as follows: in Section 2, the model and the main concepts of the Supervisory Control Theory are introduced. In Section 3.2, the central notion of our paper called partial controllability is defined, whereas in Section 3.3 we provide sufficient conditions under which our methodology gives access to a modular way to synthesize the supremal controllable language of a specification w.r.t. a concurrent DES. In section 4, some comments regarding our approach and how it encompasses the one proposed in [26] are provided. Finally, in section 5, we apply our method to the often studied *Automated Guided Vehicle* example ([16]), showing that the class of locally consistent languages is large enough to deal with relevant problems.

2 Preliminaries

2.1 Model

Each component of the system is modeled as Finite State Machines (FSM), defined by a 4-tuple $G = \langle \Sigma, Q, q_o, \delta \rangle$, where Σ is a finite alphabet. The notation $\delta(\sigma, q)!$ means that there is a transition labeled by an event σ out of state q in machine G. Likewise, for $q \in Q$ and $s \in \Sigma^*$, $\delta(s, q)$ denotes the state reached by taking the sequence of events defined by trace s from state q in machine G. The behavior of the system is described by the language $\mathcal{L}(G) \subseteq \Sigma^*$ generated by G and defined as follows: $L(G) = \{s \in \Sigma^* \mid \delta(s, q_o)!\}$. Given $s, s' \in \Sigma^*$, we say that $s' \leq s$ whenever s' is a prefix of s (i.e. it exists $t \in \Sigma^*$ s.t. $s = s't$). We denote $\overline{L}$ the prefix-closure of a language $L \subseteq \Sigma^*$ ($\overline{L} = \{s \in \Sigma^* \mid \exists s' \in L, s \leq s'\}$). Note that $L(G)$, as defined above, is prefix-closed (i.e. $L(G) = \overline{L(G)}$). For $L \subseteq \Sigma^*$ and $\Sigma' \subseteq \Sigma$, we use $L(s, \Sigma')$ to denote the set of suffixes of L after s that belongs to Σ'^*, i.e.

$$L(s, \Sigma') = \{t \in \Sigma'^* \mid st \in L\} \tag{1}$$

Let $\Sigma' \subseteq \Sigma$, then $P_{\Sigma'} : \Sigma^* \to \Sigma'^*$ is the natural projection from Σ^* to Σ'^* that erases in a sequence of Σ^* all the events that do not belong to Σ'. This definition can be easily extended to regular languages: $P_{\Sigma'}(L) = \{s' \in \Sigma'^* \mid \exists s \in L,\ s' = P_{\Sigma'}(s)\}$ Given $L \subseteq \Sigma'^* \subseteq \Sigma^*$, the inverse projection of L is defined by $P^{-1}_{\Sigma'}(L) = \{s \in \Sigma^* \mid P_{\Sigma'}(s) \in L\}$. Next, we introduce technical lemmas that will be useful in the remainder of the paper.

Lemma 1 ([11]) *Let $L \subseteq \Sigma'^*$ and $\Sigma' \subseteq \Sigma$,*

- *let $s \in L$ and $s' \in \Sigma^*$, then $ss' \in P^{-1}_{\Sigma'}(L) \Longrightarrow sP_{\Sigma'}(s') \in P^{-1}_{\Sigma'}(L)$.*
- *let $s \in P^{-1}_{\Sigma'}(L)$ and $s' \in \Sigma^*$, then $ss' \in P^{-1}_{\Sigma'}(L) \Longrightarrow sP_{\Sigma'}(s') \in P^{-1}_{\Sigma'}(L)$*

2.2 Review of the Supervisory Control Problem

Supervisory control theory deals with control of Discrete Event Systems [27]. In this theory, the behavior of the plant is assumed not to be fully satisfactory. Hence, it has to be reduced by means of a feedback control (named Supervisor) in order to achieve a given set of requirements [27].

Given a system G to be controlled, some events of G are said to be uncontrollable (Σ_{uc}), i.e. the occurrence of these events cannot be prevented by a supervisor, while the others are controllable (Σ_c). The behavior restriction of G is achieved by means of a feedback control (named *Supervisor*). Formally, a supervisor is given by a function $S : L(G) \to \{\gamma \in 2^{\Sigma} \mid \Sigma_{uc} \subseteq \gamma\}$, delivering the set of actions that are allowed in G under the control of S after a trajectory $s \in L(G)$. We write S/G for the closed-loop system, consisting in the initial system G controlled by the supervisor S. The closed-loop system S/G is a Discrete Event System that can be characterized by the language $L(S/G)$ which is recursively defined as follows:

1. $\epsilon \in L(S/G)$
2. $[(s \in L(S/G))$ *and* $s\sigma \in L(G)$ *and* $\sigma \in S(s)] \Leftrightarrow s\sigma \in L(S/G)$

Next we recall the definition of a *controllable language* [27].

Definition 1 *Let G be an FSM modeling the system and $K \subseteq L(G)$ the prefix-closed specification. Then K is controllable with respect to Σ_{uc} and G (or $L(G)$) if $K\Sigma_{uc} \cap L(G) \subseteq K$.* •

In general, given a system G and a (prefix-closed) specification $K \subseteq L(G)$, K is not controllable w.r.t. $L(G)$ and Σ_{uc}, which means that it is necessary to restrict the behavior of K in order to obtain a sublanguage of K that is controllable with respect to both $L(G)$ and Σ_{uc}.

Basic Supervisory Control Problem [27]. *Given a system modeled by an FSM G and an expected language $K \subseteq L(G)$, the problem is to build a supervisor S such that $L(S/G) \subseteq K$, and for any other supervisor S', $L(S'/G) \subseteq K \ \Rightarrow L(S'/G) \subseteq L(S/G)$.*

Remark 1 *This definition can be extended to the case where $K \not\subseteq L(G)$ by considering $K \cap L(G)$ as a new specification.*

In the sequel, we will be more interested in the computation of S/G rather than in the computation of the supervisor S itself, since one can easily extract S from S/G. One solution of this problem is given by the *supremal controllable sub-language of K w.r.t. Σ_{uc} and $L(G)$* (see [27]) and is denoted $K^{\uparrow c}$ or $SupC(K, \Sigma_{uc}, L(G))$ in the remainder of this paper. If both the system and the specification are given by one FSM, then the complexity of computing $SupC(K, \Sigma_{uc}, L(G))$ is linear in the number of states of both the system and the specification.

If many specifications are requested, the supervision can be achieved in a modular fashion [28, 27]. Each specification is implemented by a supervisor, say S_i, and the global control action is then given by $S(s) = S_1(s) \cap \cdots \cap S_n(s)$ (See Figure 3)). Based on this modular control policy, the behavior of the controlled system under the supervision of S is simply given by

$$L(S/G) = L(S_1/G) \cap \cdots \cap L(S_n/G)$$

In some situations, it is also of interest to compute the *infimal prefix-closed and controllable superlanguage of K* w.r.t. $L(G)$ and Σ_{uc}[3]. This language is denoted $K^{\downarrow L(G),c}$ and corresponds to the smallest prefix-closed language that contains K and that is controllable w.r.t. Σ_{uc} and $L(G)$. It can be shown (see e.g. [5]) that

$$K^{\downarrow L(G),c} = K\Sigma_{uc}^* \cap L(G) \tag{2}$$

2.3 Concurrent Discrete Event System.

In this paper, our purpose is to control a system composed of several components, sharing common events. To do so, let us consider a system G modeled as a collection of FSM $G_i = (\Sigma_i, Q_i, q_{oi}, \delta_i)$ $(1 \leq i \leq n)$, with $\Sigma_i \neq \Sigma_j$. The alphabet Σ of G is given by $\Sigma = \bigcup_i \Sigma_i$, and G represents the composition of FSM $(G_i)_{1 \leq i \leq n}$[4].

$$G = G_1 \parallel \cdots \parallel G_n$$

where the operation $\parallel$ is the classical *parallel composition* (i.e. $G_1 \parallel G_2$ represents the concurrent behavior of G_1 and G_2 with synchronization on the shared events).

Definition 2 *Let $G_i = (\Sigma_i, Q_i, q_{o_i}, \delta_i)$, $i = 1, 2$ be two FSM. The* parallel composition $G_1 \parallel G_2$ *of G_1 and G_2 is the FSM $G = Acc((\Sigma, Q, q_o, \delta))$ where $\Sigma = \Sigma_1 \cup \Sigma_2$, $Q = Q_1 \times Q_2$, $q_o = \langle q_{o_1}, q_{o_2} \rangle$, and δ is defined by: for all $q = \langle q_1, q_2 \rangle \in Q$ and $\sigma \in \Sigma$*

$$\delta(\sigma, \langle q_1, q_2 \rangle) = \begin{cases} \langle \delta_1(\sigma, q_1), q_2 \rangle & \text{if } \sigma \in \delta_1(q_1) \setminus \Sigma_2 \\ \langle q_1, \delta_2(\sigma, q_2) \rangle & \text{if } \sigma \in \delta_2(q_2) \setminus \Sigma_1 \\ \langle \delta_1(\sigma, q_1), \delta_2(\sigma, q_2) \rangle & \text{if } \sigma \in \delta_1(q_1) \cap \delta_2(q_2) \\ \textit{Undefined} & \textit{otherwise} \end{cases}$$

where the function Acc deletes the states that are not accessible from the initial state and update the transition function according to the new set of states.

[3] This language and its constructive definition will be useful for technical reasons in Theorem 4.

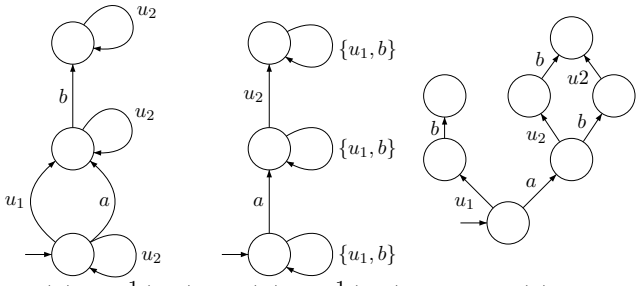
[4] Note that even though useful for theoretical reasons, the FSM G will never be computed in the remainder of this paper.

Given a concurrent discrete event system $\{G_i\}_{1\leq i\leq n}$, with $L(G_i) \subseteq \Sigma_i^*$, we simply denote by P_i the projection from Σ^* to Σ_i^* and by P_i^{-1} the inverse projection from Σ_i^* to Σ^*. Based on these operations, the language resulting from the parallel composition[5] of FSM is characterized by:

$$\begin{aligned} L(G) = L(G_1 \parallel \cdots \parallel G_n) &= L(G_1) \parallel \cdots \parallel L(G_n) \\ &= P_1^{-1}[L(G_1)] \cap \cdots \cap P_n^{-1}[L(G_n)] \end{aligned} \tag{3}$$

In the remainder of the paper, we will denote $P_i^{-1}(G_i)$ the FSM such that $L(P_i^{-1}(G_i)) = P_i^{-1}(L(G_i))$ obtained from G_i by simply adding self-loops labeled by events in $\Sigma \setminus \Sigma_i$ to each state of G_i.

Example 1 *Let $G = G_1 \parallel G_2$, with $L(G_1) = \{\epsilon, a, u_1, a.b, u_1.b\}$ and $L(G_2) = \{\epsilon, a, au_2\}$. We have $\Sigma_1 = \{a, u_1, b\}$, $\Sigma_2 = \{a, u_2\}$. Figure 1(a) and 1(b) represent the FSM generating $P_1^{-1}(L(G_1))$ and $P_2^{-1}(L(G_2))$, whereas Figure 1(c) is the FSM G (note that the FSM G_1 and G_2 can be easily obtained from Figure 1(a) and 1(b) by removing the self-loops).*



(a) $P_1^{-1}(G_1)$ (b) $P_2^{-1}(G_2)$ (c) G

Figure 1: G and its components

Event status: Given a CDES $(G_i)_{i\leq n}$ modeling G, we denote by Σ_s the set of shared events of G. It represents the set of event shared by at least two different subsystems:

$$\Sigma_s = \bigcup_{i\neq j} (\Sigma_i \cap \Sigma_j)$$

The alphabet of one subsystem G_i is split into the controllable event set $\Sigma_{i,c}$ and the uncontrollable event set $\Sigma_{i,uc}$, i.e. $\Sigma_i = \Sigma_{i,uc} \dot\cup \Sigma_{i,c}$. The alphabet of the global system G is given by:

$$\Sigma = \bigcup_i \Sigma_i, \Sigma_c = \bigcup_i \Sigma_{i,c}, \text{ and } \Sigma_{uc} = \Sigma \setminus \Sigma_c.$$

Moreover, we assume that the following relation holds between the control status of shared events:

$$\forall i, j,\ \Sigma_{i,uc} \cap \Sigma_{j,c} = \emptyset \tag{4}$$

[5]By slight abuse of notation, we use the same $\parallel$ notation for the parallel composition of languages.

which simply means that the components that share an event agree on the control status of this event. Under this hypothesis, we have that $\Sigma_{uc} = \cup_i \Sigma_{i,uc}$. Note that this assumption is not restrictive whenever a centralized approach is considered (See Section 3.1.2 which explains our approach).
The following technical lemma recalls the link that can be made between $L(G)$ and the languages $P_i^{-1}(L(G_i))$.

Lemma 2 ((3.1) of [26]) *Let G a concurrent system modeled by $\{G_i\}_{1\leq i\leq n}$, $s \in L(G)$, $i \in \{1,\ldots,n\}$ and $\sigma \in \Sigma_i \setminus \Sigma_s$. Then $s\sigma \in L(G) \Longleftrightarrow s\sigma \in P_i^{-1}(L(G_i))$*

3 Control of concurrent DES

Let G a concurrent system to be controlled, modeled by the set of FSM $\{G_i\}_{1\leq i\leq n}$, and $K \subseteq \Sigma^*$ be the specification not necessarily included in $L(G)$. Based on remark 1, the problem we are interested in is the computation of the supremal controllable sub-language $(K \cap L(G))^{\uparrow c}$ of $K \cap L(G)$ w.r.t. Σ_{uc} and $L(G)$. As we consider concurrent systems, the construction of the entire system may not be feasible (due to the state-space explosion resulting from the composition), as well as the construction of an FSM generating $K \cap L(G)$[6]. Moreover, the use of classical supervisory control methodologies (See e.g. [27] or [5]) may be impractical. It is then important to design algorithms that perform the controller synthesis phase by taking advantage of the structure of G without building it. Hence, the actual problem is to compute $(K \cap L(G))^{\uparrow c}$ without computing neither $L(G)$ nor $K \cap L(G)$.

3.1 Various approaches

Due to the concurrent nature of the system it seems quite natural to solve the control problem using a decentralized methodology based on the structure of the system. First introduced in [26], this corresponds to the classical approach that has been investigated in the literature so far [8, 23, 17, 19][7]. However, it is worthwhile noting that a centralized approach can also be used to solve this problem. We now briefly outline the works of [26] based on a decentralized approach before presenting our methodology based on a centralized but modular approach and pointing out the differences between these two approaches.

3.1.1 A Decentralized approach

The works of [26] is related to the decentralized control theory. The authors consider the control of Concurrent Discrete Event Systems G modeled by several subsystems $\{G_i\}_{1\leq i\leq n}$. Given a specification modeled as a prefix-closed language K, they provide a method that computes local modular supervisors S_i on G_i (based on a notion of separable specification (See Definition 3)) and to operate the individually controlled system S_i/G_i concurrently in such a way that the controlled behavior corresponds to the supremal controllable sublanguage of $K \cap L(G)$ w.r.t. $L(G)$.

Definition 3 *$L \subseteq \Sigma^*$ is said to be separable w.r.t. $\{\Sigma_i\}_{i\leq n}$ with $\cup_{i\leq n}\Sigma_i = \Sigma$, whenever there exists a set of languages $\{L_i\}_{i\leq n}$, s.t. $L_i \subseteq \Sigma_i^*$ and $L = L_1 \parallel \cdots \parallel L_n$.*

[6] When $L(G)$ is computable, one may assume that $K \subseteq L(G)$ (it is sufficient to consider the computable language $K \cap L(G)$ as new specification), however this would be a restricting hypothesis in our framework as $K \cap L(G)$ is not computable in general.

[7] Even though differently presented, one can use a decentralized architecture to implement the result of [8].

It may be shown that when L is separable w.r.t. $\{\Sigma_i\}_{i\leq n}$, then this language can be rewritten as : $L = P_1(L) \parallel \cdots \parallel P_n(L)$. Now, based on Definition 3, the authors of [26] show that, given a concurrent system G modeled by $\{G_i\}_{1\leq i\leq n}$ and $K \subseteq \Sigma^*$, if $\Sigma_s \subseteq \Sigma_c$, then there exists a set of local supervisors $(S_i)_{1\leq i\leq n}$ such that $\|_{1\leq i\leq n}\ L(S_i/G_i) = \mathrm{SupC}(K \cap L(G), \Sigma_{uc}, L(G))$ if and only if $\mathrm{SupC}(K \cap L(G), \Sigma_{uc}, L(G))$ is separable. Even though interesting, this result requires to build $\mathrm{SupC}(K \cap L(G), \Sigma_{uc}, L(G))$ and to check whether it is separable or not. Nevertheless, the authors show the following theorem that does not request this verification:

Theorem 1 *Let G a concurrent system modeled by $\{G_i\}_{1\leq i\leq n}$, with $L(G_i) \subseteq \Sigma_i^*$ and $K \subseteq \Sigma^*$ the specification. If $\Sigma_s \subseteq \Sigma_c$ and K is separable w.r.t. $\{\Sigma_i\}_{i\leq n}$, then*

$$\|_{i\leq n}\ SupC(P_i(K) \cap L(G_i), \Sigma_{i,uc}, L(G_i)) = SupC(K \cap L(G), \Sigma_{uc}, L(G)).$$

Hence, given a Concurrent DES G and a separable specification K, Theorem 1 shows that there exists a set of local supervisors S_i acting upon G_i such that $\|_{i\leq n}\ L(S_i/G_i) = (K \cap L(G))^{\uparrow c}$ (c.f. Figure 2).

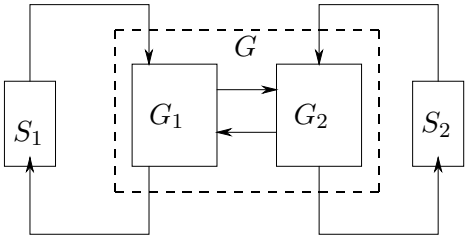


Figure 2: A Decentralized architecture for Concurrent DES

Complexity overview. If K is separable w.r.t. $\{\Sigma_i\}_{i\leq n}$ (which can be checked in $O(N_K^{n+1})$[8], where N_K is the size of the FSM that generates K), then synthesizing the local supervisors requires the computation of the projection of K over Σ_i. In the worst case, the size of the FSM that generates $P_i(K)$ is in $O(2^{N_K})$. Hence, in the worse case, solving the supervisory control problem requires $O(n.2^{N_K}.N + N_K^{n+1})$ in both space and time where N is the size of each component. Note that if the specification is directly given as a concurrent specification $K^1 \parallel \cdots \parallel K^n$ that is compatible with the alphabet of the system then the complexity raises down to $O(n.N.N_K)$, where N_K is the size of each component K^i. We will come back to this point in Section 4.2.

3.1.2 Our approach

In the previous works, the specification is requested to be separable, which is a quite restrictive condition. Indeed, this condition does not allow to model a particular behavioral interleaving between different components or a particular scheduling of actions that belong to different components. So, our aim is to find another methodology for which this condition is not needed. Instead of adopting a "*decentralized*" approach we prefer to use a *"centralized and modular"* approach [28] as shown in Figure 3. Rather than derive local specifications according to each subsystem, the idea consists in

[8] where O means *order* (See [22] for details).

approximating the global system according to each subsystem, and enforcing the initial specification with respect to these approximations leading to the computation of several supervisors. Each of them is assumed to observe the whole behavior and to give access to the set of events that are allowed after a trace of the system.

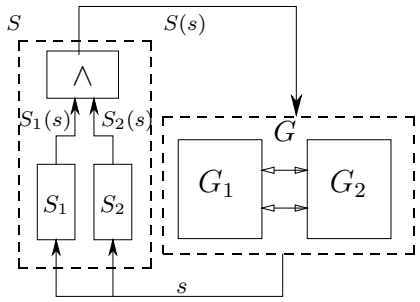


Figure 3: Supervision Scheme

From a computational point of view, the system G can be described by the following parallel composition of FSM

$$G = P_1^{-1}(G_1) \parallel \cdots \parallel P_n^{-1}(G_n), \tag{5}$$

where $P_i^{-1}(G_i)$ is the FSM such that $L(P_i^{-1}(G_i)) = P_i^{-1}(L(G_i)) \subseteq \Sigma^*$. In fact, each $P_i^{-1}(G_i)$ can be seen as an approximation of the system G, which corresponds to all the behavioral knowledge that we may deduce on G from G_i knowing that this component is coupled with other components.

Further, compared to [26], given a specification K, instead of performing computations w.r.t. each component G_i (i.e. $L(G_i)$) in order to enforce $P_i(K)$, we have chosen to perform the computations from $P_i^{-1}(L(G_i))$ in order to enforce K in a modular fashion. Hence our problem is to find conditions under which we are able to synthesize supervisors such that

$$L(S_1/P_1^{-1}(G_1)) \cap \cdots \cap L(S_n/P_n^{-1}(G_n)) \;=\; (K \cap L(G))^{\uparrow c}$$

Unfortunately, it is not sufficient to compute a supervisor S_i w.r.t. $P_i^{-1}(G_i)$ that restricts the behavior $P_i^{-1}(L(G_i))$ to the supremal controllable sublanguage of $K \cap P_i^{-1}(L(G_i))$ w.r.t. $\Sigma_{i,uc}$ and $P_i^{-1}(L(G_i))$ and to coordinate the supervisors S_i according to Figure 3 to obtain the supremal controllable sublanguage of $K \cap L(G)$. The result may be not controllable[9]. This is basically due to the duality between the local and global controllable events. The idea is then to refine the notion of controllability in order to take into account the fact that uncontrollable events may be local to a component. The property that we ensure on each $P_i^{-1}(G_i)$ according to K is called *the partial controllability condition* and is defined in Section 3.2.

3.2 Partial Controllability Condition

Definition 4 *Let $M \subseteq L \subseteq \Sigma^*$ be prefix-closed languages. Let $\Sigma'_{uc} \subseteq \Sigma_{uc} \subseteq \Sigma$ be two sub-alphabets of Σ. Let $M' \subseteq M$, then M' is partially controllable with respect to Σ'_{uc}, Σ_{uc}, M and L if*

[9]If for all $i \leq n$ we compute the supremal controllable sublanguage of $K \cap P_i^{-1}(L(G_i))$ w.r.t. Σ_{uc} and $P_i^{-1}(L(G_i))$, then the result may be not maximal.

(i) M' *is controllable w.r.t* Σ'_{uc} *and* L.

(ii) M' *is controllable w.r.t* Σ_{uc} *and* M. •

Intuitively, L will be seen as an approximation of the system w.r.t. one of its components and M as the initial specification (C.f. Section 3.3.1 and Theorem 2). Now given a sub-behavior of M, the idea is that we may allow the violation of the controllability condition by triggering an uncontrollable event σ that is not local (i.e. $\sigma \in \Sigma_{uc} \setminus \Sigma'_{uc}$), because L only constitutes an approximation of the system and because at least one of the other supervisors, computed from the other approximations, will avoid these events to be admissible. However, we still want to enforce the controllability of M' w.r.t. M and Σ_{uc} because M will constitute the actual specification and not an approximation.

In general, M is not partially controllable with respect to Σ'_{uc}, Σ_{uc}, M and L (e.g. if M is not controllable w.r.t. Σ'_{uc} and L). However, it can be shown that there exists a supremal sub-language of M that has this property.

Proposition 1 *Let* $M \subseteq L \subseteq \Sigma^*$ *be prefix-closed languages,* $\Sigma'_{uc} \subseteq \Sigma_{uc}$. *There exists a unique supremal language, denoted by* $M^{\uparrow pc}$, *which is partially controllable w.r.t* Σ'_{uc}, Σ_{uc}, M *and* L. *Moreover*

$$M^{\uparrow pc} = \overline{M^{\uparrow pc}} = SupC(SupC(M, \Sigma'_{uc}, L), \Sigma_{uc}, M) \tag{6}$$

Proof: First of all, the set of partially controllable languages w.r.t Σ'_{uc}, Σ_{uc}, M and L is not empty, since $\emptyset$ is partially controllable w.r.t Σ'_{uc}, Σ_{uc}, M and L. Now let $M' \subseteq M$ be a partially controllable language w.r.t Σ'_{uc}, Σ_{uc}, M and L. Since $M' \subseteq M$ is controllable w.r.t Σ'_{uc} and L, $M' \subseteq \mathrm{SupC}(M, \Sigma'_{uc}, L)$, and by monotony of the $\mathrm{SupC}(\bullet, \Sigma_{uc}, M)$ operator, we have that

$$\mathrm{SupC}(M', \Sigma_{uc}, M) \subseteq \mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$$

Now, $M' \subseteq M$ is controllable w.r.t Σ_{uc} and M. This entails that $M' = \mathrm{SupC}(M', \Sigma_{uc}, M)$, and finally

$$M' \subseteq \mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$$

which entails that $\mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$ contains any sub-language of M that is partially controllable w.r.t Σ'_{uc}, Σ_{uc}, M and L.

Let us now show that $\mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$ is partially controllable w.r.t Σ'_{uc}, Σ_{uc}, M and L.

(i) let $s \in \mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$ and $\sigma \in \Sigma'_{uc}$ such that $s\sigma \in L$. As

$$\mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M) \subseteq \mathrm{SupC}(M, \Sigma'_{uc}, L)$$

we obtain $s\sigma \in \mathrm{SupC}(M, \Sigma'_{uc}, L)\Sigma'_{uc} \cap L$. Hence, by definition of controllability, $s\sigma \in \mathrm{SupC}(M, \Sigma'_{uc}, L)$ and as $\mathrm{SupC}(M, \Sigma'_{uc}, L) \subseteq M$, we also have that $s\sigma \in M$.

Moreover, since $\Sigma'_{uc} \subseteq \Sigma_{uc}$, we obtain that $s\sigma \in \mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)\Sigma_{uc} \cap M$. Hence, by definition of controllability again,

$$s\sigma \in \mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$$

Thus $\mathrm{SupC}(\mathrm{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$ is controllable w.r.t. Σ'_{uc} and L.

(ii) By definition, $\text{SupC}(\text{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$ is controllable w.r.t Σ_{uc} and M.

Finally $\text{SupC}(\text{SupC}(M, \Sigma'_{uc}, L), \Sigma_{uc}, M)$ is partially controllable w.r.t Σ'_{uc}, Σ_{uc}, M and L. Moreover, it contains any partially controllable language w.r.t Σ'_{uc}, Σ_{uc}, M and L. Therefore, it is the supremal (and we denote it by $M^{\uparrow pc}$). The fact that $M^{\uparrow pc}$ is prefix-closed is trivial since the SupC() operator preserves the prefix closure. ◇

Proposition 1 offers a practical way to compute $M^{\uparrow pc}$, the supremal partially controllable sub-language of M w.r.t. to Σ'_{uc}, Σ_{uc}, M and L. This language will be sometimes denoted $\text{SupPC}(M, \Sigma'_{uc}, \Sigma_{uc}, L)$ in the sequel. It can then be shown that the required complexity for the computation of $M^{\uparrow pc}$ is in $O(N_M.N_L)$, where N_M (resp N_L) is the size of the automaton generating M (resp. L).

3.3 Control of Concurrent DES

Given a Concurrent DES G modeled by FSM $\{G_i\}_{1 \leq i \leq n}$, and a specification K, we want to compute a controllable sub-language of $K \cap L(G)$ w.r.t. $L(G)$ and Σ_{uc}, without having to build G itself.

3.3.1 Modular Computation of a controllable sub-language of K w.r.t. $L(G)$

Based on the concept of partial controllability applied on K and on the approximations of the system $P_i^{-1}(L(G_i))$ derived from each of its components, the next theorem provides a modular way to compute a sub-language of K that is controllable with respect to the system.

Theorem 2 *Let G a concurrent system modeled by FSM $\{G_i\}_{1 \leq i \leq n}$ and let $K \subseteq \Sigma^*$ be a prefix-closed language modeling the specification. For $i \leq n$, we denote*

- $K_i = K \cap P_i^{-1}(L(G_i))$, *and*
- $K_i^{\uparrow pc}$ *the supremal sublanguage of K_i partially controllable with respect to $\Sigma_{i,uc}$, Σ_{uc}, K_i and $P_i^{-1}(L(G_i))$.*

Then, $K_1^{\uparrow pc} \cap \cdots \cap K_n^{\uparrow pc}$ is controllable with respect to Σ_{uc} and $L(G)$.

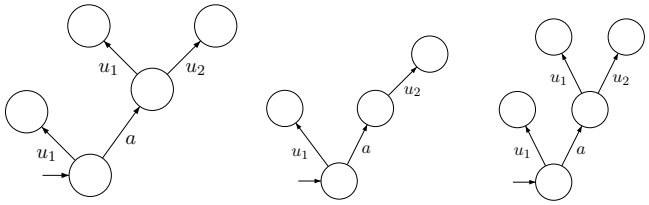
Proof: First, since K and $L(G_i)$ are prefix-closed, then for $i \leq n$ the languages $P_i^{-1}(L(G_i))$ as well as K_i and $K_i^{\uparrow pc}$ are prefix-closed (Proposition 1). Now, according to Definition 1, we have to show that

$$(\bigcap_{i \leq n} K_i^{\uparrow pc})\Sigma_{uc} \cap L(G) \subseteq \bigcap_{i \leq n} K_i^{\uparrow pc}$$

Let $s \in \bigcap_{i \leq n} K_i^{\uparrow pc}$ and $\sigma \in \Sigma_{uc}$ be such that $s.\sigma \in L(G)$. We thus have to show that $s\sigma \in \bigcap_{i \leq n} K_i^{\uparrow pc}$. Without loss of generality we can assume that $\sigma \in \Sigma_{1,uc}$.
Since $s\sigma \in L(G)$, $s\sigma \in P_1^{-1}(L(G_1))$. Hence we have $s\sigma \in K_1^{\uparrow pc}\Sigma_{1,uc} \cap P_1^{-1}(L(G_1))$. Moreover, according to Definition 4, $K_1^{\uparrow pc}$ is controllable w.r.t. $\Sigma_{1,uc}$ and $P_1^{-1}(L(G_1))$, from which we can conclude that $s\sigma \in K_1^{\uparrow pc}$.
As $s\sigma \in K_1^{\uparrow pc}$, we have that $s\sigma \in K$. And as $s\sigma \in L(G)$, then $\forall i \leq n,\ s\sigma \in P_i^{-1}(L(G_i))$, which entails that $\forall i \leq n, s\sigma \in K_i = K \cap P_i^{-1}(L(G_i))$. Hence, $\forall i \leq n, s\sigma \in K_i{}^{\uparrow pc}\Sigma_{uc} \cap K_i$. Now, according to Definition 4, $\forall i \leq n,\ K_i^{\uparrow pc}$ is controllable w.r.t. Σ_{uc} and K_i. Hence $\forall i \leq n,\ s\sigma \in K_i^{\uparrow pc}$, which entails that $s\sigma \in \bigcap_{i \leq n} K_i^{\uparrow pc}$. ◇

Theorem 2 gives us a modular method to compute a sub-language of K that is controllable w.r.t. $L(G)$ and Σ_{uc}. Moreover, this computation is performed without building the system G (i.e. there is no need to perform the parallel composition between the components of G). From a computational point of view, based on Section 3.2, for $i \leq n$, the computation of $K_i^{\uparrow pc}$ is in $O(N.N_K)$ (where N represents the number of states of any sub system G_i and N_K the one of the specification K).

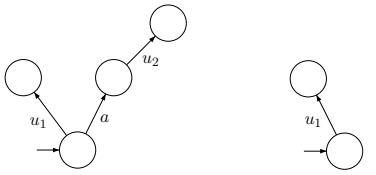
Example 2 *Let $G = G_1 \parallel G_2$, as described in Example 1 with $\Sigma_{1_{uc}} = \{u_1\}$ and $\Sigma_{2_{uc}} = \{u_2\}$ and let us consider the specification given by the language $K \subseteq \Sigma^*$ as described in Figure 4(a)[10]. Our aim is to compute a sub-language of $K \cap L(G)$ that is controllable w.r.t. $L(G)$ and Σ_{uc}. Following Theorem 2, we first compute the languages $K_i = K \cap P_i^{-1}(L(G_i))$, $i = 1, 2$. The FSM generating K_1 and K_2 are represented in Figures 4(b) and 4(c).*



(a) K (b) K_1 (c) K_2

Figure 4: K and the derived specifications K_1 and K_2

Now based on (6), we compute $K_1^{\uparrow pc}$ (resp $K_2^{\uparrow pc}$), the supremal language of K_1 (resp. K_2) that is partially controllable w.r.t. Σ_{uc}, $\Sigma_{1,uc}$ and $P_1^{-1}(L(G_1)$, (resp $\Sigma_{2,uc}$ and $P_2^{-1}(L(G_2))$) (c.f. Figure 5). The intersection of these two languages is the language $K_1^{\uparrow pc} \cap K_2^{\uparrow pc} = \{\epsilon, u_1\}$, that is obviously controllable w.r.t. $L(G)$ and Σ_{uc}.



(a) $K_1^{\uparrow pc}$ (b) $K_2^{\uparrow pc}$

Figure 5: The resulting supremal partially controllable languages

Let us now describe the way a supervisor can be extracted from the previously computed languages and how it can act upon G in order to achieve the specification K. With the notations of Theorem 2, $\bigcap_{i \leq n} K_i^{\uparrow pc}$ is controllable with respect to Σ_{uc} and $L(G)$. However, it is not interesting

[10]Note that b belongs to the alphabet of K even though it can not be triggered.

to perform the intersection between these languages and to derive a supervisor from the result (all the computational advantages of our method would be lost).

Each $K_i^{\uparrow pc}$ can be seen as a supervisor S_i which is able to restrict behaviors of $P_i^{-1}(G_i)$ to a controllable one with respect to $\Sigma_{i,uc}$. Since $P_i^{-1}(G_i)$ is an over-approximation of G, it is also possible to apply S_i to G. Doing this for all i leads to the concept of modularity described in [28]. But since the controlled systems generated by two different S_i are controllable with respect to two different sets of uncontrollable events, the results of [28] can not be used. But Theorem 2 justifies the use of a modular architecture. Hence, from a given behavior, only events enabled by all the S_i (derived from $K_i^{\uparrow pc}$) are effectively enabled. In other words, the global supervisor acting upon G is given by $S(s) = S_1(s) \cap \cdots \cap S_n(s)$. The supervisor architecture is summarized in Figure 3.

3.4 Computation of $(K \cap L(G))^{\uparrow c}$

The above methodology allows us to compute a controllable sub-language of $K \cap L(G)$. Indeed, according to Theorem 2, we have that $\bigcap_{i\leq n} K_i^{\uparrow pc}$ is a controllable sub-language of $(K \cap L(G))^{\uparrow c}$. However, it may happen that this language is not the maximal permissive one (As in Example 1 in which the supremal controllable sublanguage of $K \cap L(G)$ is given by $(K \cap L(G))^{\uparrow c} = \{\epsilon, a, u_1, au_2\}$). In this section, we present some conditions under which Theorem 2 gives access to the supremal solution.

First, it is worthwhile noting that, in general, uncontrollable shared events are not adequate to perform local computations. Indeed, in order to ensure the partial controllability of K_i w.r.t. $\Sigma_{i,uc}$, Σ_{uc}, $P_i^{-1}(L(G_i))$, we may need to disable a shared uncontrollable event by control, even though this event is not fireable in the global system (this is due to the fact that we are working on approximations and thus with local informations). This leads us to restrict the class of the concurrent DES to the class of concurrent DES that do not share uncontrollable events (i.e. $\Sigma_s \subseteq \Sigma_c$). Next, we show that, under this new assumption, $(K \cap L(G))^{\uparrow c}$ verifies point (i) of the partial controllability definition.

Lemma 3 *Let G be a concurrent system modeled by FSM $\{G_i\}_{1\leq i\leq n}$ and $K \subseteq \Sigma^*$ a prefix-closed language modeling the specification, then if $\Sigma_s \subseteq \Sigma_c$, then $\forall i \leq n$, $(K \cap L(G))^{\uparrow c}$ is controllable w.r.t $\Sigma_{i,uc}$, $P_i^{-1}(L(G_i))$.*

Proof: Let $i \leq n$. Let us consider $s \in (K \cap L(G))^{\uparrow c}$ and $\sigma \in \Sigma_{i,uc}$ such that $s\sigma \in P_i^{-1}(L(G_i))$. We have to show that $s\sigma \in (K \cap L(G))^{\uparrow c}$. Since $\sigma \in \Sigma_{i,uc}$ and $\Sigma_s \subseteq \Sigma_c$, we have $\sigma \in \Sigma_i \setminus \Sigma_s$. Moreover, $s \in P_i^{-1}(L(G_i))$, $s \in L(G)$ and $s\sigma \in P_i^{-1}(L(G_i))$, which entails that $s\sigma \in L(G)$ (Lemma 2). Now, as $\Sigma_{i,uc} \subseteq \Sigma_{uc}$, we have that $s\sigma \in (K \cap L(G))^{\uparrow c}.\Sigma_{uc} \cap L(G)$. Since $(K \cap L(G))^{\uparrow c}$ is controllable w.r.t Σ_{uc} and $L(G)$, this entails that $s\sigma \in (K \cap L(G))^{\uparrow c}$. $\diamond$

3.4.1 The specification is a subset of $L(G)$

Based on this lemma, Theorem 3.4.1 shows that whenever the specification models sub-behaviors of the system (i.e $K \subseteq L(G)$), then applying Theorem 2 provides a maximal permissive supervisor.

Theorem 3 *If $\Sigma_s \subseteq \Sigma_c$ and $K \subseteq L(G)$, then with the notations of Theorem 2, $\bigcap_{i\leq n} K_i^{\uparrow pc} = K^{\uparrow c}$.*

Proof: From Theorem 2, $\bigcap_{i\leq n} K_i^{\uparrow pc} \subseteq K^{\uparrow c}$. It is then sufficient to show that $K^{\uparrow c} \subseteq \bigcap_{i\leq n} K_i^{\uparrow pc}$ or, equivalently that $\forall i \leq n$, $K^{\uparrow c} \subseteq K_i^{\uparrow pc}$. To do so, we choose $i \leq n$ and prove that $K^{\uparrow c}$ is partially controllable w.r.t. $\Sigma_{i,uc}$, Σ_{uc}, K_i and $P_i^{-1}(L_i)$.

(i) First, according to Lemma 3, $K^{\uparrow c}$ is controllable w.r.t $\Sigma_{i,uc}$ and $P_i^{-1}(L(G_i))$.

(ii) Let us now show that $K^{\uparrow c}$ is controllable w.r.t. Σ_{uc} and K_i. Let us consider $s \in K^{\uparrow c}$, $\sigma \in \Sigma_{uc}$ such that $s\sigma \in K_i$, we have to prove that $s\sigma \in K^{\uparrow c}$. Since $s\sigma \in K_i \subseteq L(G)$, $s\sigma \in L(G)$. We then have $s \in K^{\uparrow c}$, $\sigma \in \Sigma_{uc}$ and $s\sigma \in L(G)$. Hence, because $K^{\uparrow c}$ is controllable w.r.t. Σ_{uc} and $L(G)$, $s\sigma \in K^{\uparrow c}$.

This proves that $K^{\uparrow c}$ is partially controllable w.r.t. $\Sigma_{i,uc}$, Σ_{uc}, K_i, $P_i^{-1}(L(G_i))$. Now, as $K_i^{\uparrow pc}$ is supremal and partially controllable w.r.t. $\Sigma_{i,uc}$, Σ_{uc}, K_i, $P_i^{-1}(L(G_i)$, we can deduce that $\forall i \leq n,\ K^{\uparrow c} \subseteq K_i^{\uparrow pc}$. Finally, $K^{\uparrow c} \subseteq \bigcap_{i\leq n} K_i^{\uparrow pc}$ and the proof is done. ◇

Theorem 3 states that whenever the shared events are controllable and the language of the specification is included in that of the system, our method computes the supremal controllable sub-language of K w.r.t. $L(G)$ and Σ_{uc}. We recall that the complexity of our method is in $O(n.N.N_K)$. This has to be opposed to the complexity $O(N^n.N_K)$ of computing $(K \cap L(G))^{\uparrow c}$ when G is seen as a unique FSM (of course this complexity only stands for prefix-closed specification). Finally note that it is sufficient to check that $\forall 1 \leq i \leq n,\ K \subseteq P_i^{-1}(L(G_i))$ in order to check that $K \subseteq L(G)$. Hence, it is not necessary to compute $L(G)$.

Example 3 *Let us consider the system given in Figure 1 again, and the specification given by* $K = \overline{\{u_1, au_2\}}$. *It is easy to show that* $K \subseteq L(G)$. *Moreover, for this specification, we have* $K = {K_1}^{\uparrow pc} = {K_2}^{\uparrow pc}$, *which implies that* ${K_1}^{\uparrow pc} \cap {K_2}^{\uparrow pc} = K$. *Hence according to Theorem 3,* $K = SupC(K, \Sigma_{uc}, L)$. ◇

3.4.2 The specification is locally consistent w.r.t. $L(G)$

In some situations, modeling the specification by a language included in the language of the system may lead to a language that is too complex to be efficiently represented. Moreover, requiring the inclusion of languages makes that the specification of K may be itself relatively difficult to identify insofar as $L(G)$ is not known (the inclusion can only be checked *a posteriori*). One can consider a specification that requires the system to trigger only once a particular event, say a. As such, this specification can be modeled by an FSM with two states. However, if we request this specification to be included in the behavior of the system, we would have to unfold the system in order to only take into account the behaviors that match this specification.

To alleviate these problems (size, inclusion and difficulty of modeling), we now introduce a new condition under which our methodology gives access to the supremal controllable sub-language of K w.r.t. L and Σ_{uc}. This condition does not require the specification to be included in the one of the system and allows us to have specifications that are relatively independent of the system. This condition is called *local consistency* and is given by Definition 6. But first, we introduce the notion of *consistency*:

Definition 5 *Let* $\Sigma' \subseteq \Sigma$ *be two alphabets and let* $M \subseteq \Sigma^*$ *be a prefix-closed language. Let us consider alphabets* $\Sigma_{uc} \subseteq \Sigma$ *and* $\Sigma'_{uc} = \Sigma_{uc} \cap \Sigma'$. M *is said to be consistent with respect to* Σ_{uc} *and* $P_{\Sigma'}$ *if* $\forall s \in M,\ \forall s' \in M(s, \Sigma_{uc})$ [11], $\forall \sigma \in \Sigma'_{uc}$,

[11] See Equation (1).

$$P_{\Sigma'}(s')\sigma \in M(s, \Sigma'_{uc}) \Rightarrow s'\sigma \in M(s, \Sigma_{uc}).$$

•

Definition 5 captures a certain interleaving between the local (Σ'_{uc}) and global uncontrollable events. In particular, among other aspects, this condition induces that if after a sequence s of M, there is a local uncontrollable event that is admissible, then this event is admissible whenever M triggers uncontrollable events that belong to $\Sigma_{uc} \setminus \Sigma'_{uc}$ (C.f. Figure 6)).

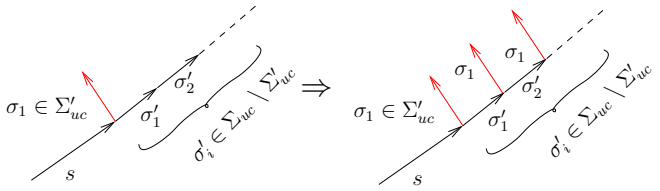


Figure 6: An aspect of the Definition 5 reflecting the interleaving

Definition 6 *Let us consider a concurrent system G modeled by FSM $\{G_i\}_{1\leq i\leq n}$ and K be a prefix-closed language over Σ. K is said to be* locally consistent *with respect to Σ_{uc} and G if* $\forall i \in \{1, \ldots, n\}$,

$$K \cap P_i^{-1}(L(G_i)) \text{ is consistent with respect to } \Sigma_{uc} \text{ and } P_i.$$

•

Therefore, a language is said to be locally consistent for a concurrent systems whenever it is consistent according to each local subsystems. Intuitively, if K is *locally consistent* with respect to Σ_{uc} and $L(G)$, then it means that the possible interleaving between the local/global uncontrollable events w.r.t. each approximation of the system are taken into account in the specification K. Roughly speaking, it means that K respects the interleaving between the local and global uncontrollable events as long as they happen in the approximations.

Example 4 *Let us consider the concurrent system given by the parallel composition of G_1 and G_2 respectively represented in Figures 7 (a) and 7 (b). The sets of uncontrollable events are respectively given by $\Sigma_1 = \{u_1, u'_1\}$ and $\Sigma_2 = \{u_2, u'_2\}$. In this example, we consider several specifications,*

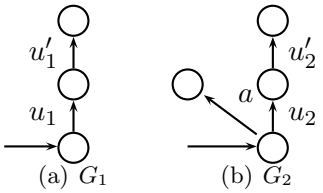


Figure 7: The two components G_1 and G_2 of G.

given by Figures 8(a) to 8(d). In order to check the local consistency of K^j we need to check the consistency of $K^j \cap P_i^{-1}(L(G_i))$ w.r.t. Σ_{uc} and $P_i^{-1}(L(G_i))$.

It is easy to show that both K^1 and K^2 are locally consistent w.r.t. G. However, one can check that K^3 is not locally consistent because $K^3 \cap P_2^{-1}(L(G_2))$ is not consistent. Indeed, in $K^3 \cap P_2^{-1}(L(G_2))$ $(= K^3)$, we have that $P_2(u_1).u_2 \in K^3$ (since $P_2(u_1).u_2 = u_2$) while $u_1.u_2 \notin K^3$.

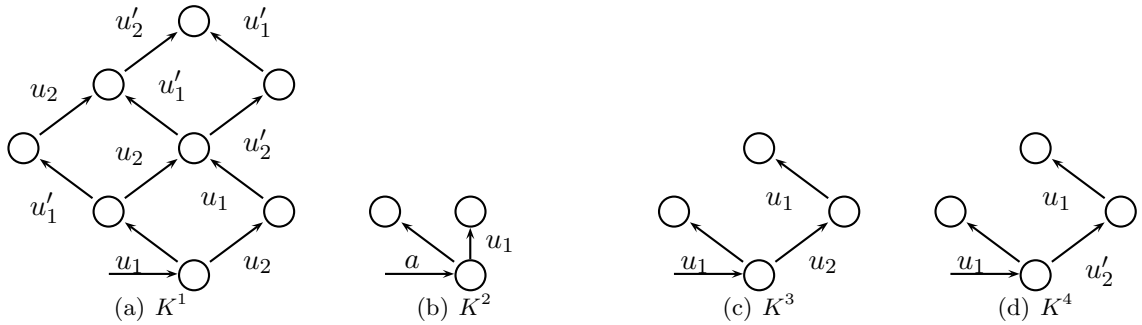


Figure 8: Different specifications

Finally, whereas K^4 seems very similar to K^3, K^4 is locally consistent. Indeed, $K^4 \cap P_2^{-1}(L(G_2)) = \{u_1\}$ and is consistent. The difference between K^3 and K^4 is that the sequence u_2 belongs to $P_2^{-1}(L(G_2))$ whereas u_2' does not. $\diamond$

Theorem 4 *If $\Sigma_s \subseteq \Sigma_c$ and K is locally consistent, then with the notations of Theorem 2,*

$$\bigcap_{1 \leq i \leq n} K_i^{\uparrow pc} = (K \cap L(G))^{\uparrow c}$$

Proof: Let $K \subseteq \Sigma^*$ be a locally consistent language w.r.t. Σ_{uc} and G. According to Theorem 2, we have that $\bigcap_{i \leq n} K_i^{\uparrow pc}$ is controllable w.r.t. Σ_{uc} and $L(G)$ and is included in $(K \cap L(G))^{\uparrow c}$. We thus have to show that $\forall i \leq n,\ (K \cap L(G))^{\uparrow c} \subseteq K_i^{\uparrow pc}$.
To do so, let us consider $i \leq n$ and $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ (the infimal prefix-closed and controllable superlanguage of $(K \cap L(G))^{\uparrow c}$ w.r.t. Σ_{uc} and K_i). We now prove that $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ is partially controllable w.r.t $\Sigma_{i,uc}$, Σ_{uc}, K_i and $P_i^{-1}(L(G_i))$. Indeed, if $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ is partially controllable w.r.t $\Sigma_{i,uc}$, Σ_{uc}, K_i and $P_i^{-1}(L(G_i))$, then we will have that $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c} \subseteq {K_i}^{\uparrow pc}$. Moreover, as $(K \cap L(G))^{\uparrow c} \subseteq ((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$, we will also have that $(K \cap L(G))^{\uparrow c} \subseteq {K_i}^{\uparrow pc}$ and the proof will be done.

Let us now show that $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ is partially controllable w.r.t $\Sigma_{i,uc}$, Σ_{uc}, K_i and $P_i^{-1}(L(G_i))$. According to Definition 4, we have to show that $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ is *(i)* controllable w.r.t. $\Sigma_{i,uc}$ and $P_i^{-1}(L(G_i))$ and *(ii)* controllable w.r.t. Σ_{uc} and K_i. Point *(ii)* is obvious since by the definition of the infimal controllable language w.r.t. Σ_{uc} and K_i, $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ is controllable w.r.t. Σ_{uc} and K_i. Let us now prove the point *(i)*.

Let us consider $s \in ((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ and $\sigma \in \Sigma_{i,uc}$ such that $s\sigma \in P_i^{-1}(L(G_i))$. We have

$$s\sigma \in ((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}.\Sigma_{i,uc} \cap P_i^{-1}(L(G_i))$$

Moreover, according to (2),

$$((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c} = (K \cap L(G))^{\uparrow c}.\Sigma_{uc}^* \cap K_i$$

Thus, s is of the form $s = s't$ with $s' \in (K \cap L(G))^{\uparrow c}$ and $t \in \Sigma_{uc}^*$. Now, as $s\sigma = s't\sigma \in P_i^{-1}(L(G_i))$, we also have that $s'P_i(t\sigma) \in P_i^{-1}(L(G_i))$ (Lemma 1).

We now have that $s' \in (K \cap L(G))^{\uparrow c}$, $P_i(t\sigma) \in \Sigma_{i,uc}^*$ and $s'P_i(t\sigma) \in P_i^{-1}(L(G_i))$. We then have that $s'P_i(t\sigma) \in (K \cap L(G))^{\uparrow c}$ as $(K \cap L(G))^{\uparrow c}$ is controllable w.r.t. $\Sigma_{i,uc}$ and $P_i^{-1}(L(G_i))$

(Lemma 3). Finally, as $s'P_i(t\sigma) \in (K \cap L(G))^{\uparrow c} \subseteq K_i$, we obtain $P_i(t\sigma) = P_i(t)\sigma \in K_i(s', \Sigma_{i,uc})$. But K_i is consistent since K is locally consistent (Definition 6). We thus obtain that $t\sigma \in K_i(s', \Sigma_{uc})$ (Definition 5). Hence $s\sigma$ $(= s't\sigma) \in K_i$ and it entails that

$$s\sigma \in ((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}.\Sigma_{uc} \cap K_i$$

As $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ is (by definition) controllable w.r.t. Σ_{uc} and K_i, $s\sigma \in ((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$. Thus $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ is controllable w.r.t. $\Sigma_{i,uc}$ and $P_i^{-1}(L(G_i))$.

Finally, $((K \cap L(G))^{\uparrow c})^{\downarrow K_i, c}$ is partially controllable w.r.t $\Sigma_{i,uc}$, Σ_{uc}, K_i and $P_i^{-1}(L(G_i))$, which concludes the proof. $\diamond$

Theorem 4 states that the *local consistency* condition together with $\Sigma_s \subseteq \Sigma_c$ are sufficient conditions under which our approach solves the *Basic Supervisory Control Problem*[12]. Now, given prefix-closed languages K and $L(G)$, the complexity of checking *local consistency* is $O(n.N^2.N_K^2)$ (where N_K denotes the number of states of the FSM generating K and N represents the number of states of any sub system G_i (See [9] for details regarding how the local consistency check is performed). Moreover, as previously mentioned, the complexity of computing $(K \cap L(G))^{\uparrow c}$ with our method is $O(n.N_K.N)$. Therefore, whenever our method can be applied, its overall complexity is $O(n.N^2.N_K^2)$.

3.5 How to relax the $\Sigma_s \subseteq \Sigma_c$ assumption

According to Theorems 3 or 4 in order to solve the Basic Supervisory Control Problem, we do require (1) the specification to be locally consistent or to be included in $L(G)$ and (2) the shared events to be controllable. However, none of these conditions is necessary (only sufficient) to obtain a maximal solution. In the next corollary we actually show that it is possible not to consider the second condition as far as the uncontrollable shared events are not involved during the computation phase. Moreover, the modular computation approach gives an efficient way to check for this. If this property holds, then our approach provides a maximal solution to the BSCP, even if some shared event are uncontrollable[13].

Corollary 1 *Consider a concurrent system G modeled by FSM $\{G_i\}_{1 \leq i \leq n}$ s.t. $L(G_i) \subseteq \Sigma_i^*$ and $K \subseteq \Sigma^*$. If $K \subseteq L(G)$ or K is locally consistent w.r.t $\Sigma_{uc} \setminus \Sigma_s$ and G and if*

$$\forall i, SupPC(K_i, \Sigma_{i,uc}, \Sigma_{uc}, P_i^{-1}(L(G_i))) = SupPC(K_i, (\Sigma_{i,uc} \setminus \Sigma_s), (\Sigma_{uc} \setminus \Sigma_s), P_i^{-1}(L(G_i))) \quad (7)$$

Then

$$\bigcap_i SupPC(K_i, \Sigma_{i,uc}, \Sigma_{uc}, P_i^{-1}(L(G_i))) = SupC(K \cap L(G), \Sigma_{uc}, L(G))$$

If Condition 7 holds then it simply says that uncontrollable shared events are not involved in the computation of the supremal partially controllable sub-language of K with respect to $\Sigma_{i,uc}$ and Σ_{uc} (i.e. considering $\Sigma_{i,uc} \cap \Sigma_s$ as controllable or not gives access to the same solution). Now let us prove Corollary 1.

Proof: Let us first consider the language $\mathrm{SupC}(K \cap L(G), (\Sigma_{uc} \setminus \Sigma_s), L(G))$ which can be seen as the supremal controllable sub-language of $K \cap L(G)$, considering that the shared events are

[12] In [10], a similar result was obtained but with a more restrictive condition named G-observability.

[13] This property can be helpful in some non trivial cases. See section 5, in which the the AGV problem is treated.

controllable. Then as $K \subseteq L(G)$ (resp. K is locally consistent w.r.t $\Sigma_{uc} \setminus \Sigma_s$), based on Theorem 3 (resp. Theorem 4), we have that

$$\bigcap_i \text{SupPC}(K_i, (\Sigma_{i,uc} \setminus \Sigma_s), (\Sigma_{uc} \setminus \Sigma_s), P_i^{-1}(L(G_i))) = \text{SupC}(K \cap L(G), (\Sigma_{uc} \setminus \Sigma_s), L(G)) \quad (8)$$

But by assumption,

$$\bigcap_i \text{SupPC}(K_i, \Sigma_{i,uc}, \Sigma_{uc}, P_i^{-1}(L(G_i))) = \bigcap_i \text{SupPC}(K_i, (\Sigma_{i,uc} \setminus \Sigma_s), (\Sigma_{uc} \setminus \Sigma_s), P_i^{-1}(L(G_i))) \quad (9)$$

Then according to equations (8) and (9), we have

$$\bigcap_i \text{SupPC}(K_i, \Sigma_{i,uc}, \Sigma_{uc}, P_i^{-1}(L(G_i))) = \text{SupC}(K \cap L(G), \Sigma_{uc} \setminus \Sigma_s, L(G))$$

Now, according to Theorem 2, $\bigcap_i \text{SupPC}(K_i, \Sigma_{i,uc}, \Sigma_{uc}, P_i^{-1}(L(G_i)))$ is controllable w.r.t Σ_{uc} and $L(G)$, which entails that

$$\bigcap_i \text{SupPC}(K_i, \Sigma_{i,uc}, \Sigma_{uc}, P_i^{-1}(L(G_i))) \subseteq \text{SupC}(K \cap L(G), \Sigma_{uc}, L(G)) \quad (10)$$

As $(\Sigma_{uc} \setminus \Sigma_s) \subseteq \Sigma_{uc}$, we have $\text{SupC}(K \cap L(G), \Sigma_{uc}, L(G)) \subseteq \text{SupC}(K \cap L(G), \Sigma_{uc} \setminus \Sigma_s, L(G))$. Hence, $\text{SupC}(K \cap L(G), \Sigma_{uc}, L(G)) \subseteq \bigcap_i \text{SupPC}(K_i, \Sigma_{i,uc}, \Sigma_{uc}, P_i^{-1}(L(G_i)))$. Overall, we obtain,

$$\bigcap_i \text{SupPC}(K_i, \Sigma_{i,uc}, \Sigma_{uc}, P_i^{-1}(L(G_i))) = \text{SupC}(K \cap L(G), \Sigma_{uc}, L(G))$$

$\diamond$

4 Some comments regarding the specification

The previous section provides a modular method that efficiently computes the supremal controllable sub-language of a specification K with respect to a concurrent system G. This computation is performed without having to build the whole system G. To capture the maximality property, we request K to be either included in $L(G)$ or locally consistent w.r.t. G and Σ_{uc}. However, one of the drawback of our method is that we required so far that the specification is modeled by a unique language and it is worthwhile noting that the complexity of our method depends on the size of the specification, which in turn may be, for example, modeled

(1) either by a collection of specifications $K^i \subseteq \Sigma^*$ that have to be ensured at the same time in a modular way (Note that $K_i \not\subseteq \Sigma_i^*$).

(2) or modeled itself as a separable specification $K \subseteq \Sigma^*$, obtained by synchronizing several local specifications $K_i \subseteq \Sigma_i^*$, i.e. $K = K_1 \| \cdots \| K_n$.

Next, we investigate how to apply our methodology when the specification is either modular or separable. To simplify the proof and the notations, we assume that the shared events are controllable knowing that we may further relax this condition as shown in Corollary 1.

4.1 Modular Specifications

Assume that the global specification is given in a modular way e.g. $K \cap K'$, with $K \subseteq \Sigma^*$, $K' \subseteq \Sigma^*$ and K, K' prefix-closed, we then have the following result:

Proposition 2 *Let G be a concurrent system modeled by FSM $\{G_i\}_{1\leq i\leq n}$ such that $\Sigma_s \subseteq \Sigma_c$, K and K' be two prefix-closed languages over Σ^*. For $i \leq n$, we denote*

- $K_i = K \cap P_i^{-1}(L(G_i))$, *and* $K'_i = K' \cap P_i^{-1}(L(G_i))$
- $K_i^{\uparrow pc}$ *(resp.* $K_i'^{\uparrow pc}$*) the supremal sublanguage of* K_i *(resp.* K'_i*) partially controllable with respect to* $\Sigma_{i,uc}$, Σ_{uc}, K_i *(resp.* K'_i*) and* $P_i^{-1}(L(G_i))$.

If K and K' are either included into $L(G)$ or locally consistent, then

$$\bigcap_{i\leq n} K_i^{\uparrow pc} \cap \bigcap_{i\leq n} K_i'^{\uparrow pc} = ((K \cap K') \cap L(G))^{\uparrow L(G),c}$$

Indeed, from [28], we have that

$$(K \cap L(G))^{\uparrow c} \cap (K' \cap L(G))^{\uparrow c} = (K \cap K')) \cap L(G))^{\uparrow L(G),c}$$

then in the case of prefix-closed languages, the modularity results in [28] together with the results of Theorem 3 or 4, ensure the results as far as the conditions of these theorems are satisfied by the two specifications, which is the case by assumptions.

Remark 2 *Note that this is only a sufficient condition, it may be the case that $K \cap K'$ respects either the inclusion condition or the local consistency condition but not K or K'. In this case, we still need to compute the whole specifications.*

4.2 Separable Specification

In [26] the authors introduced the notion of separable languages and proved that when the specification is separable then there exists a set of local supervisors acting upon each component so that the controlled behavior is equal to $(K \cap L(G))^{\uparrow c}$ (see Section 3.1.1). It is interesting to compare the class of languages that are separable with the class of languages that are locally consistent. This is the aim of the next proposition.

Proposition 3 *Consider a concurrent system G modeled by FSM $\{G_i\}_{1\leq i\leq n}$ such that $L(G_i) \subseteq \Sigma_i^*$ and $K \subseteq \Sigma^*$. Then, under the assumption that $\Sigma_s \subseteq \Sigma_c$, if K is separable w.r.t. $\{\Sigma_i\}_{1\leq i\leq n}$ then K is locally consistent w.r.t Σ_{uc} and G.*

Proof: Let $K \subseteq \Sigma^*$ be a separable (Definition 3) language w.r.t $\{\Sigma_i\}_{1\leq i\leq n}$. Then for $i \leq n$, let $L'_i \subseteq \Sigma_i^*$ be such that $K = \|_{1\leq i\leq n} L'_i$. Now, for any $i \leq n$, let us consider the languages $K_i = K \cap P_i^{-1}(L(G_i))$. According to Definition 6, we have to show that K_i is consistent w.r.t Σ_{uc} and P_i. Let $s \in K_i$, $s' \in K_i(s, \Sigma_{uc})$ and $\sigma \in \Sigma_{i,uc}$ be such that $P_i(s')\sigma \in K_i(s, \Sigma_{uc})$. It is then sufficient to show that $s'\sigma \in K_i(s, \Sigma_{uc})$. Now, as $s' \in K_i(s, \Sigma_{uc})$, we have $ss' \in K_i \subseteq K$. Now as $\forall j$, $K \subseteq P_j^{-1}(L'_j)$ we also have that $\forall j$, $P_j(ss') \in L'_j$. Moreover, $\sigma \in \Sigma_i \setminus \Sigma_s$, which entails that $\forall j \neq i$, $P_j(ss'\sigma) = P_j(ss')$ from which we deduce that $P_j(ss'\sigma) \in L'_j$. Therefore,

$\forall j \neq i$, $ss'\sigma \in P_j^{-1}(L'_j)$. It is finally sufficient to show that $ss'\sigma \in P_i^{-1}(L'_i)$. We have that $sP_i(s'\sigma) \in K_i$ as $P_i(s')\sigma \in K_i(s, \Sigma_{uc})$ and $P_i(s')\sigma = P_i(s'\sigma)$. Hence $sP_i(s'\sigma) \in K$ which implies that $sP_i(s'\sigma) \in P_i^{-1}(L'_i)$. Finally, we can deduce from Lemma 1 that $ss'\sigma \in P_i^{-1}(L'_i)$. Hence the result. $\diamond$

The previous proposition states that under the hypothesis that the shared events are controllable, whenever the specification is separable w.r.t. $\{\Sigma_i\}_{1\leq i\leq n}$ then it is *locally consistent* w.r.t. any concurrent system having $\{\Sigma_i\}_{1\leq i\leq n}$ as local alphabets. Moreover, one can easily find a language that is *locally consistent* but not separable (See e.g. K^4 in Example 4). This implies that the class of *locally consistent* languages is strictly greater than the class of separable ones. Based on this result, one can note that if K is separable (but not directly given as a concurrent specification), then our methodology theoretically offers an alternative way to compute $(K \cap L(G))^{\uparrow c}$. However, if K is a separable control objective modeled as $K^1 \parallel \cdots \parallel K^n$, then applying our approach would require the computation of K which once computed can be of the size of the system. This means that if our method is naively applied then it is less efficient than that of [26]. Nevertheless, we now demonstrate how our methodology can be adapted in order to obtain the same complexity.

First we can remark that if $K = K^1 \parallel \cdots \parallel K^n$, then is it true that $K = P_1^{-1}(K^1) \cap \cdots \cap P_n^{-1}(K^n)$. The idea is then to consider the n $P_i^{-1}(K^i)$ as n specifications that have to be ensured on G and then to apply proposition 2. To that purpose, we first need to check that these specifications respect the local consistency condition. Proposition 4 shows that this is useless.

Proposition 4
If $\Sigma_s \subseteq \Sigma_c$ and $K^i \subseteq \Sigma_i^*$ then $P_i^{-1}(K^i)$ is locally consistent w.r.t Σ_{uc} and $G = G_1 \parallel \cdots \parallel G_n$.

Proof: According to Definition 6, we have to show that, $\forall j \leq n$, K_j^i, defined by $K_j^i = P_i^{-1}(K_i) \cap P_j^{-1}(L(G_j))$, is consistent w.r.t Σ_{uc} and P_j. To do so, let us consider $s \in \Sigma^*$, $s' \in \Sigma_{uc}^*$ and $\sigma \in \Sigma_{j,uc}$ s.t $ss' \in K_j^i$ and $sP_j(s')\sigma \in K_j^i$. We thus have to prove that $ss'\sigma \in K_j^i$.
According to the definition of K_j^i, we have that $sP_j(s')\sigma \in P_j^{-1}(L(G_j))$ and $sP_j(s')\sigma \in P_i^{-1}(K^i)$. However we have that

$$\begin{aligned} sP_j(s')\sigma \in P_j^{-1}(L(G_j)) &\Rightarrow P_j(sP_j(s')\sigma) \in L(G_j) \\ &\Rightarrow P_j(ss'\sigma) \in L(G_j) \Rightarrow ss'\sigma \in P_j^{-1}(L(G_j)) \qquad (a) \end{aligned}$$

Let us now show that $ss'\sigma \in P_i^{-1}(K^i)$.

- If $i = j$, then as $sP_j(s')\sigma = sP_i(s')\sigma \in P_i^{-1}(K^i)$, we have that $P_i(sP_i(s')\sigma) \in K^i$, which entails that $P_i(ss'\sigma) \in K^i$ and finally $ss'\sigma \in P_i^{-1}(K^i)$.
- If $i \neq j$, by hypothesis, we have $ss' \in K_i^j$ and therefore $ss' \in P_i^{-1}(K^i)$. It means that $P_i(ss') \in K^i$. Now as $\sigma \notin \Sigma_i$ (because $\sigma \in \Sigma_{j,uc}$ and $\Sigma_s \subseteq \Sigma_c$), we also have that $P_i(ss'\sigma) \in K^i$. We thus deduce that $ss'\sigma \in P_i^{-1}(K^i)$.

Finally, knowing that $\forall j \leq n$, $ss'\sigma \in P_j^{-1}(L(G_j))$ and $ss'\sigma \in P_i^{-1}(K^i)$, we have that $ss'\sigma \in K_j^i$ and $\forall j \leq n$, K_j^i is then consistent w.r.t. Σ_{uc} and P_j, which entails that $P_i^{-1}(K^i)$ is locally consistent w.r.t Σ_{uc} and G. $\diamond$

Proposition 4 gives us a first naive method to apply the method described in Section 3 when the specification is given by $K = K^1 \parallel \cdots \parallel K^n$. This proposition entails that it is sufficient to consider the set $(P_i^{-1}(K^i))_{i \leq n}$ as n specifications. All these specifications are locally consistent and according to Theorem 4, we can compute the supremal controllable sub-language of these specifications w.r.t. $L(G)$ and Σ_{uc} using our method. The modular result given by Proposition 2 ensures the validity (as well as the maximality) of the final result.

Nevertheless for one specification $P_i^{-1}(K^i)$, the proposed method requires the computation of n supervisors. This implies that n^2 supervisors have to be computed to solve the problem with this approach. The next proposition shows that only a part of these supervisors actually need to be computed (in fact, exactly one per specification $P_i^{-1}(K^i)$).

Proposition 5 *If $\Sigma_s \subseteq \Sigma_c$ and $K^i \subseteq \Sigma_i^*$, then for $i \neq j$, $P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$ is partially controllable w.r.t $\Sigma_{j,uc}$, Σ_{uc}, $P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$ and $P_j^{-1}(L(G_j))$.*

Proof: First of all, for $i \neq j$, $P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$ is clearly controllable w.r.t Σ_{uc} and $P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$. Hence, according to Definition 4, it is sufficient to show that $P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$ is controllable w.r.t $\Sigma_{j,uc}$ and $P_j^{-1}(L(G_j))$.
To do so, let us consider $s \in P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$ and $\sigma \in \Sigma_{j,uc}$ such that $s\sigma \in P_j^{-1}(L(G_j))$. We have to show that $s\sigma \in P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$. Now, since $s\sigma \in P_j^{-1}(L(G_j))$, it is sufficient to show that $s\sigma \in P_i^{-1}(K^i)$.
By hypothesis $s \in P_i^{-1}(K^i)$, which entails $P_i(s) \in K^i$. Moreover, since $i \neq j$, we also have that $\sigma \notin \Sigma_i$ and thus $P_i(\sigma) = \epsilon$. Therefore, $P_i(s) = P_i(s).P_i(\sigma) \in K_i$, and then $P_i(s\sigma) \in K^i$. This can be rewritten as $s\sigma \in P_i^{-1}(K^i)$. Overall, $s\sigma \in P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$ which is then controllable w.r.t. $\Sigma_{j,uc}$ and $P_j^{-1}(L(G_j))$. $\diamond$

Given a specification $P_i^{-1}(K^i)$ (that is *locally consistent* according to Proposition 4), according to Theorem 4 the BSCP can be solved by computing, for all $j \leq n$, the supremal partially controllable language w.r.t. $\Sigma_{j,uc}, \Sigma_{uc}, P_i^{-1}(K^i) \cap P_j^{-1}(L_j)$ and $P_j^{-1}(L_j)$. But, as a consequence of Proposition 5, we know that the computations are unnecessary for $i \neq j$. This implies that only one supervisor needs to be computed *per* specification $P_i^{-1}(K^i)$. Overall, only n supervisors are synthesized when the specification is given by a concurrent specification $K^1 \parallel \cdots \parallel K^n$. We thus obtain the same complexity as the one described in [26].

Example 5 *We again consider the concurrent system G described in Example 1 as well as the specification K given by $K^1 \parallel K^2$ (where K^1 and K^2 are respectively given in Figure 9 (a) and (b)).*

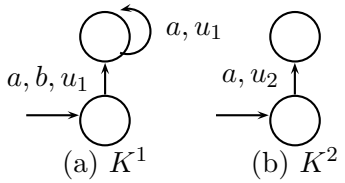
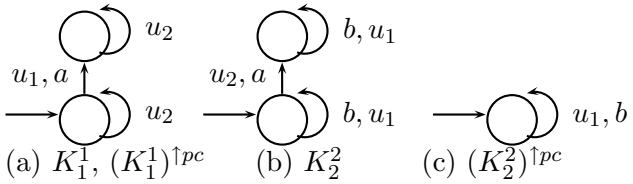


Figure 9: $K = K^1 \parallel K^2$

We know that $K = P_1^{-1}(K^1) \cap P_2^{-1}(K_2)$. Now, according to Proposition 4, for $i = 1, 2$, $P_i^{-1}(K^i)$ is locally consistent w.r.t Σ_{uc} and G and we may apply our methodology on each $P_i^{-1}(K^i)$. To do so we need to compute the languages $(K_j^i)^{\uparrow pc}$, $i, j = 1, 2$ (where K_j^i denotes $P_i^{-1}(K^i) \cap P_j^{-1}(L(G_j))$). However, according to Proposition 5, we only need to compute $(K_i^i)^{\uparrow pc}$, since $(K_j^i)^{\uparrow pc} = K_j^i$ if $i \neq j$.



Finally, based on the remarks given after Proposition 5, we have that $(K_1^1)^{\uparrow pc} \cap (K_2^2)^{\uparrow pc} = \{u_1\} = (K \cap L(G))^{\uparrow c}$.

Remark 3 *In [1], the authors consider the control Interactive Discrete Event Systems whose class encompasses the Concurrent Systems one. The system as well as the specification can be seen as a concurrent system to which is added a compensator that is used to somehow coordinate the different subsystems. In general, the approach of [1] is more general that the one presented in this paper as both the system and the specification are represented by a model that is more expressive. However, when restricting this approach to the control of concurrent discrete event system, the proposed approach is less general in the sense that the compensator, which can be seen as a global specification has to fulfill a property that is more restrictive that the one presented in this paper (Definition 6).*

5 Example

In this section, we use the quite classical AGV (Automated Guided Vehicles) example introduced in [16] to illustrate our methodology. In this example, the uncontrolled system is a manufacturing system which consists of five work-stations (WST) linked together with five automated guided vehicles. These AGVs bring some resources from one station to an other. More precisely, the figure 10 represents the way the resources move.

Let us denote by $AGV1, \ldots, AGV5$, the FSM modeling the behavior of the Automated Guided Vehicles and by $WST1, \ldots, WST5$ the FSM modeling the behavior of the work-stations. Some of these FSM are presented in Figures 11, 12 and 13. Note that $WST1$ is actually described by means of two parallel FSM $WST11$ and $WST12$[14]. The events s_i represent the synchronisations between the AGV and the different work-stations, whereas the a_i, a_i' (resp. the b_i) encode the movement of the AGV (resp. the tasks achieved by a workstation as well as the movement inside a workstation). Finally, the initial states of these FSM are represented with square in the figures.

The global manufacturing system is modeled as the parallel composition of the different FSM previously introduced. The number of states of each subsystems is less than twelve and the number of states of the entire system is about thirty millions. Moreover, in this example, most of events are assumed to be uncontrollable. Only events $(c_i)_{1 \leq i \leq 5}$ are indeed controllable and it is worthwhile noting that some shared events are uncontrollable. It means that condition (7) of Corollary 1 is required to apply the proposed methodology.

[14] In [16], the system was modeled with a cyclic Petri Net that can easily modeled by means of FSM

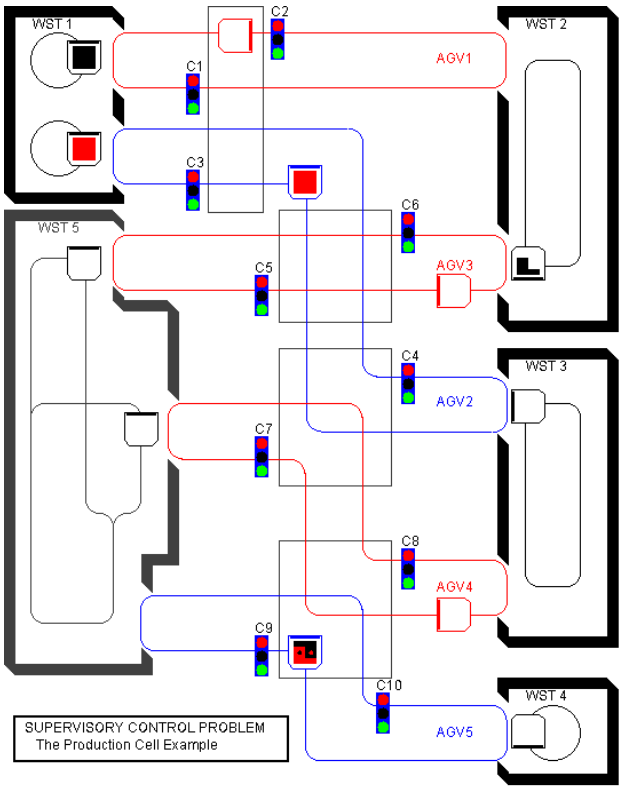


Figure 10: The AGV Example

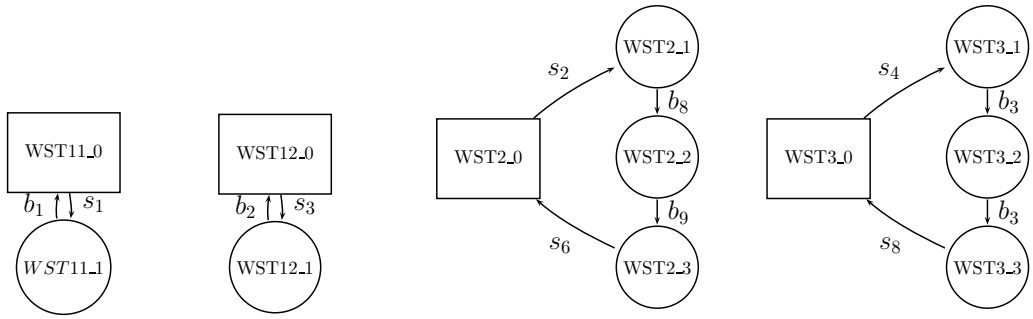


Figure 11: Work stations WST11, WST12, WST2, and WST3

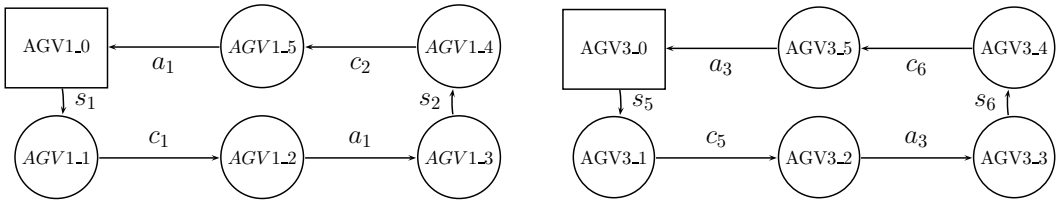


Figure 12: AGV1 & AVG3

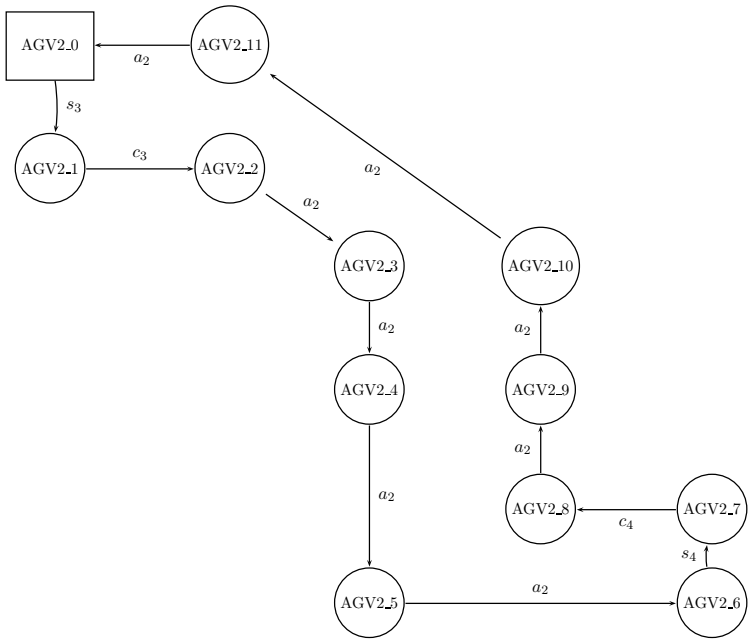


Figure 13: AGV2

The classical problem, described in [16], consists in computing one supervisor which prevents some conflicting zones from being reachable in the same time by at least two AGV. The conflicting zones are indicated with black rectangles in figure 10.

According to the FSM representation of the system, the set of behaviors avoiding each conflicting zone can be represented by an FSM $Zone_i$. Figure 14 corresponds to the conflicting zone $Zone_1$ involving AGV_1 and AGV_2. The meaning of the FSM in figure 14 is the following: it is possible for an AGV to enter the conflicting zone $Zone_1$ either when event c_1, c_2, c_3 or a_2' occurs. Then, when such an event is triggered, this FSM enters into a state from which none other of these events can be triggered. For example, looking at figure 12, one can see that after event c_1 occurs, one AGV is into the conflicting zone $Zone_1$. Now, if event a_1 occurs, then the AGV goes out of this zone.

Each conflicting zone 2,3,4 can be modeled by an FSM similar to the one represented in Figure 14. It can be shown that each specification $Zone_i$ is actually locally consistent w.r.t. to the system. Moreover, for each specification the condition (7) of Corollary 1 holds. Based on Corollary 1, we can then compute for each specification a supervisor S^i[15] such that $L(S^i/G) = SupC(L(Zone_i) \cap L(G), \Sigma_{uc}, L(G))$. Finally, the modularity result of Proposition 2 ensures that $\cap_{i\leq 4} L(S^i/G) = SupC(L(\|_i(Zone_i)) \cap L(G), \Sigma_{uc}, L(G))$.

This example was treated, using an implementation of the method proposed in this paper. Since the system consists of 12 FSM, the method provides 12 supervisors which ensure the objective $Zone_1$ acting together. Performing computation with a 3,6 Ghz Xeon processor and 2 Megabyte memory, 16 seconds were needed to compute the supervisors ensuring $Zone_1$[16].

Now, one point could be emphasis. It can be shown that the specifications we considered here are not separable (i.e. it is not possible to to model the specifications as a separable language

[15] each of them is actually given by a set of local supervisors $(S^i_j)_{j\leq n}$.

[16] About 15 seconds were actually needed to check for the local consistency of the control objective, while about one second is used to compute the supervisors.

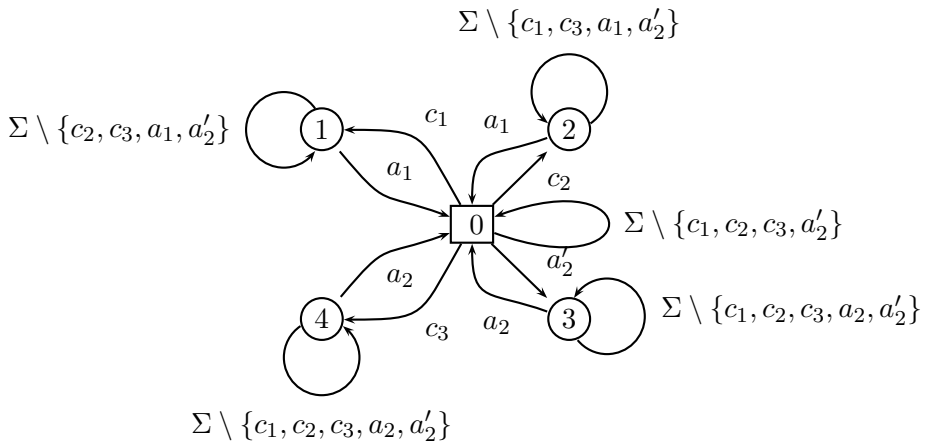


Figure 14: behaviors avoiding the first zone

over alphabets $\{(\Sigma_{AGV_i})_{1\leq i\leq 5}, (\Sigma_{WST_i})_{1\leq i\leq 5}\}$. Therefore, the methodology described in [26] can not be applied as such. However, it would be possible to perform the product of some subsystems of the plant so that the specification becomes separable w.r.t. the alphabets of the new subsystem. Anyway, this would render the previous method less efficient as the size of the new sub-systems would be greater. In the same spirit, the approach described in [8] can not be efficiently applied as this would require the computation of the whole system.

Another control objective: Let us now consider again the specification modeled by the FSM Obj2 represented in Figure 15. This control objective aims at restricting the behaviors of the AGV system so that a particular ordering of some events is ensured. This ordering concerns the events c_1, c_3, c_5, a_3, c_7 and a_5. This ordering represents one particular interleaving of the behaviors of the $(AGV_i)_{1\leq i\leq 5}$. It means for example that the occurrence of event c_5 depends on the one of event a_5. Moreover, the occurrence of these two events depends on the ones of events c_1 and c_3, and c_3 is not allowed to occur before c_1 occurred. This control objective allows to order actions of different subsystems. It can be shown that it is locally consistent and not controllable. One can note that for this example, each AGV_i for $i \in \{1, \ldots, n\}$ is involved. The method of [26] does not allow to order occurrence of events which do not belong to the same subsystem. Hence, applying this methods requires to compute $\|_{1\leq i\leq n}\ AGV_i$ (Since the alphabets of two different AGV_i are disjoint, this FSM has 34560 states (i.e the product of the number of states of each AGV_i)).

A contrario, applying our approach only requires to manage FSM with less than 72 states (i.e. the size of $AGV_2||$obj2). The global supervisor is given as the conjunction of several supervisors.

This simple scheduling specification illustrates the fact that the complexity of our method is still efficient even though some subsystems are constrained each others by the specification.

Performing computation with a 3,6 Ghz Xeon processor and 2 Megabyte memory again, about 35 seconds were needed to compute the supervisors ensuring Obj_2[17].

[17] About 34 seconds were actually needed to check for the local consistency of the control objective, while about one second is used to compute the supervisors.

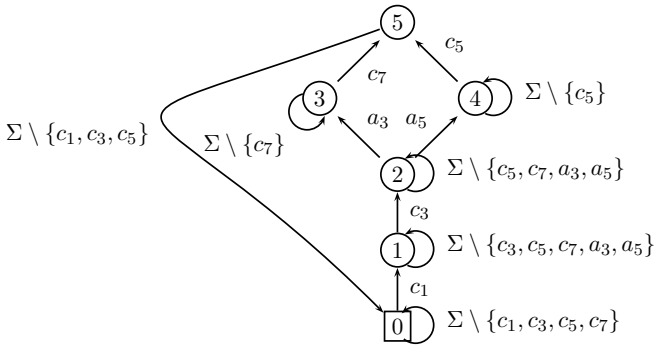


Figure 15: Another control objective: obj2

6 Conclusion

In this paper, we investigate the Supervisory Control of Concurrent Discrete Event Systems. In particular, we propose an efficient modular method that computes the supremal controllable language included into a specification K w.r.t. to the system G. From the system G and each of its components G_i, we derive a set of approximations $(P_i^{-1}(G_i))_{i \leq n}$ and we ensure by control that each of these approximations respects a new language property, called partial controllability condition that depends on K. It is then shown that whenever the original specification respects some conditions (either $K \subseteq L(G)$, or K is locally consistent) then a centralized supervisor can be extracted from the controlled approximations in such a way that the behavior of the controlled system corresponds to the supremal controllable language contained in the language of the specification. This computation is performed without having to build the whole system, hence avoiding the state space explosion induced by the concurrent nature of the system. Moreover, if one wants to change a component of G, e.g. replacing G_i by G'_i, then as far as G'_i is expressed using the same alphabet as the one of G_i with the same partitioning between the controllable/uncontrollable event, then it is sufficient to recompute $K_i'^{\uparrow pc} = (K \cap P_i^{-1}(L(G'_i))^{\uparrow pc}$ in order to obtain the new supervisor (note that only the conditions referring to G'_i have to be (re)-checked). Hence this methodology is suitable for reconfigurable systems. Finally, note that our method can be used in complement to the one of [8] and [2] whenever the sub-specifications do not concern the whole system (i.e for each sub-specification, our method can be used to compute the supervisors on the concerned sub-machines, without having to build the corresponding global FSM).

However, for some control problems, it may happen that the specification that has to be ensured is more related to the notion of states rather than to the notion of trajectories of the system (the mutual exclusion problem for example). For this class of problem, one of the main issue is the *State Avoidance Control Problem* or dually the *Invariance Control Problem*. If one wants to used a language-based approach to encode this problem, then the obtained specification may be of the size of the global system itself which renders the use of the above works intractable in the sense that the computation of the specification requires the computation of the whole system. Hence, the previous method is not suitable for this kind of control problems (See [13] for a methodology totally devoted to the state avoidance control problem).

So far we have been interested in the control of system for prefix-closed specifications modeling e.g. safety properties. We are currently looking for results ensuring that the controlled system

is non-blocking while still avoiding the computation of the whole state space. Another point of interest would be to extend theses techniques to the hierarchical model described in [12].

References

[1] S. Abdelwahed and W. Wonham. Supervisory control of interacting discrete event systems. In *41th IEEE Conference on Decision and Control*, pages 1175–1180, Las Vegas, USA, December 2002.

[2] K. Akesson, H. Flordal, and M. Fabian. Exploiting modularity for synthesis and verification of supervisors. In *Proc. of the IFAC*, barcelona, Spain, July 2002.

[3] B. Brandin, R. Malik, and P. Dietrich. Incremental system verification and synthesis of minimally restrictive behaviours. In *Proceedings of the American Control Conference*, pages 4056–4061, Chicago, Illinois, June 2000.

[4] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM computing Surveys*, pages 293–318, September 1992.

[5] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.

[6] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete–event processes with partial observations. *IEEE Transaction on Automatic Control*, 33(3):249–260, March 1988.

[7] M. H. De Queiroz and J.E.R. Cury. Modular control of composed systems. In *Proceedings of the American Control Conference*, pages 4051–4055, Chicago, Illinois, June 2000.

[8] M.H. deQueiroz and J.E.R. Cury. Modular supervisory control of large scale discrete-event systems. In *Proc of 5th Workshop on Discrete Event Systems, WODES 2000*, Ghent, Belgium, August 2000.

[9] B. Gaudin. *Synthèse de contrôleurs sur des systèmes à événements discrets structurés.* PhD thesis, Université de Rennes 1, November 2004.

[10] B. Gaudin and H. Marchand. Modular supervisory control of a class of concurrent discrete event systems. In *Proc of 7th Workshop on Discrete Event Systems, WODES'04*, September 2004.

[11] B. Gaudin and H Marchand. Supervisory control of concurrent discrete event systems. Research Report 1593, IRISA, February 2004. available at http://www.irisa.fr/vertecs/Publis/Ps/1593.ps.

[12] B. Gaudin and H Marchand. Supervisory control of product and hierarchical discrete event systems. *European Journal of Control*, 10(2), 2004.

[13] B. Gaudin and H. Marchand. Efficient computation of supervisors for loosely synchronous discrete event systems: A state-based approach. In *6th IFAC World Congress*, Prague, Czech Republic, July 2005.

[14] J. Gunnarsson. *Symbolic Methods and Tools for Discrete Event Dynamic Systems*. PhD thesis, Linköping University, 1997.

[15] G. Hoffmann and H. Wong-Toi. Symbolic synthesis of supervisory controllers. In *Proc. of 1992 American control Conference, Chicago,Il,USA*, pages 2789–2793, 1992.

[16] L.E. Holloway and B. Krogh. Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Transactions on Automatic Control*, 35(5):514–523, May 1990.

[17] S. Jiang and R. Kumar. Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(5):653–660, 2000.

[18] R.J. Leduc, B.A. Brandin, W.M. Wonham, and M. Lawford. Hierarchical interface-based supervisory control: Serial case. In *Proc. of 40th Conf. Decision Contr.*, pages 4116–4121, December 2001.

[19] S.-H. Lee and Wong K.C. Structural decentralized control of concurrent discrete-event systems. *European Journal of Control*, 8(5), 2002.

[20] C. Ma. *Non blocking supervisory control of state tree structures*. PhD thesis, Univeristy of Toronto, February 2004.

[21] H. Marchand, P. Bournai, M. Le Borgne, and P. Le Guernic. Synthesis of discrete-event controllers based on the signal environment. *Discrete Event Dynamic System : Theory and Applications*, 10(4):347–368, October 2000.

[22] Ch. Papadimitriou. *Computational complexity*. Addison-Wesley Reading, 1994.

[23] K. Rohloff and S. Lafortune. The control and verification of similar agents operating in a broadcast network environment. In *42nd IEEE Conference on Decision and Control*, Hawaii, USA, December 2003.

[24] K. Rudie and W.M. Wonham. Think globally, act locally: decentralized supervisory control. *IEEE Transaction on Automatic Control*, 31(11):1692–1708, November 1992.

[25] A. Vahidi, B. Lennarston, and M. Fabian. Efficient supervisory synthesis of large systems. In *Proc of 7th Workshop on Discrete Event Systems, WODES'04*, September 2004.

[26] Y. Willner and M. Heymann. Supervisory control of concurrent discrete-event systems. *International Journal of Control*, 54(5):1143–1169, 1991.

[27] W. M. Wonham. Notes on control of discrete-event systems. Technical Report ECE 1636F/1637S, Department of Electrical and Computer EngineeringUnivertsity of Toronto, July 2003.

[28] W. M. Wonham and P. J. Ramadge. Modular supervisory control of discrete event systems. *Mathematics of Control Signals and Systems*, 1:13–30, 1988.

[29] T. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. In *Proc of 5th Workshop on Discrete Event Systems, WODES 2000*, Ghent, Belgium, August 2000.