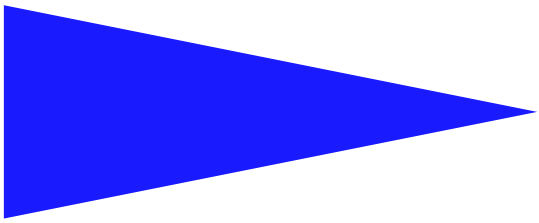
I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

P U B L I C A T I O N
I N T E R N E
N° 1569



SUPERVISORY CONTROL OF STRUCTURED DISCRETE EVENT SYSTEMS

BENOIT GAUDIN, HERVÉ MARCHAND

ISSN 1166-8687

I R I S A
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

Supervisory Control of Structured Discrete Event Systems

Benoit Gaudin, Hervé Marchand

Thème 1 — Réseaux et systèmes
Projet VerTeCs

Publication interne n1569 — November 2003 — 27 pages

Abstract: In this paper, the control of a class of Discrete Event Systems is investigated. Discrete event systems are modeled by a collection of loosely synchronous Finite State Machines. The basic problem of interest is to ensure the invariance of a set of particular configurations in the system. We provide algorithms that, based on a particular decomposition of the set of forbidden configurations, locally solve the control problem (i.e. on each component without computing the whole system) and produce a global supervisor ensuring the desired property.

Key-words: Discrete Event Systems, Supervisory Control Problem, structured Finite State Machines, Optimal control

(Resume : tsvp)

Centre National de la Recherche Scientifique
(UPRESSA 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Contrôle de systèmes à événements discrets structurés

Résumé : Dans ce papier, nous nous intéressons au contrôle de systèmes à événements discrets modélisés par des machines à états finis structurées, partageant des événements communs. Le problème du contrôle que nous nous posons est d'assurer l'interdiction d'un ensemble particulier de configurations dans le système. Nous présentons des algorithmes qui, basés sur une décomposition particulière de cet ensemble, résolvent localement les problèmes de contrôle (i.e. sur chaque composant du système sans avoir à calculer explicitement le système) et produisent un contrôleur global assurant la propriété attendue. Ce type objectifs peut être utilisé pour décrire/assurer des interactions entre différents sous-systèmes.

Mots clés : Systèmes à événements discrets, contrôle, systèmes structurés, commande optimale

1 Introduction

In this paper, we are interested in the Supervisory Control of structured Discrete Event Systems (DES). Our control methodology follows the theory of Ramadge & Wonham [18]. The system to be controlled can be described by means of Finite State Machines, that generates sequences of events (or actions) through its execution. Given a plant (P) and a specification of the expected behavior (S), the control of the plant is then performed by inhibiting some events in Σ_c, the set of controllable events in such a way that the behavior of the controlled plant is included (in the sense of language) in (S).

In many applications (as e.g. manufacturing system, control-command system, protocol network, etc) and control problems, FSMs are the starting point to model fragments of a large scale system, which usually consists of the composition of many different interacting sub-systems. The global behavior of the system is then given by the parallel composition of the different system components (they evolve concurrently while synchronizing on the shared events). The standard way of applying the supervisor synthesis methodology to such systems is by expanding them to ordinary state machines and by using classical synthesis tools on the resulting FSM. However, knowing that the synthesis algorithms are polynomial in the number of states of the systems and that the number of states of the global systems grows exponentially with the number of parallel and nested sub-systems, it is important to design algorithms that perform the controller synthesis phase by taking advantage of the structure of the system without expanding it. In other words, given the modular structure of the system, it becomes of interest, for computational reasons, to be able to synthesize a supervisor on each sub-part of the system and then to infer a global supervisor from the local ones.

Several approaches have been considered in the literature to deal with reducing the complexity of supervisor synthesis.

- In [25], the methods consists in decomposing the global control objectives into sub-objectives and to perform the controller synthesis phase separately w.r.t. these sub-objectives.
- In [6, 7], the authors considers product systems (i.e. systems composed of asynchronous subsystems, not sharing common events). Given a control objective, a local system is built from the sub-systems that are coordinated by the control objective (i.e. all the sub-systems that share some events used to express it). It is then sufficient to compute the local supervisor ensuring the control objective with respect to this local system in order to obtain the result on the whole system. Finally, the modularity property [25] ensures that a maximal solution of the control problem can be obtained by computing the local supervisors w.r.t. to the local control objectives[1]. A similar work based on an incremental synthesis methodology can be found in [3]. The fact that only a sub-part of the whole system has to be built in order to compute a supervisor obviously decreases the complexity of its computation. However, it may happen that given a control objective, the whole plant need to be computed (e.g. if all the events of the alphabet of the plant are necessary to express the objective). In [10], the authors provide an elegant way to solve the SCP when both the plant and the specification (or control objective) are given in a modular way. The controller synthesis is then performed by computing supervisors that are further added to the specification, itself seen as the initial supervisor). Note that in the above approach, no particular connection between the components of the plant and of the specification are required. In contrast, we adopt a state-based approach in which the control objective exactly fits with the structure of the plant.
- Closely related to the decentralized theory (see e.g. [13, 26]), in [23], the authors consider Concurrent Discrete Event Systems $G_1 \parallel \cdots \parallel G_n$ and provide some solutions allowing to compute local modular supervisors $\mathcal{S}_i$ on G_i (based on a notion of separable control objective) and to operate the individually controlled system $\mathcal{S}_i/G_i$ concurrently. However, when the control objective is not separable (e.g.

[1] Note that the authors also give necessary and sufficient conditions under which the obtained controlled system is non-blocking

it expresses (forbidden) interactions between all sub-systems, the method proposed in [23] cannot be efficiently used since the whole system may have to be built.

- In [1, 2], the authors also investigate the control of Concurrent DESs and propose a methodology based on the decomposition of the control objective according to the structure of the system, leading to a reduction of the complexity, when the events that are shared by the different components are controllable (this is performed by adding an interaction component that restricts the concurrent behavior of the components). A possibly blocking solution is then obtained.

In this paper, we consider the control of Concurrent DESs (i.e. the whole plant is modeled as a collection of plants $(G_i)_{i\leq n}$, sharing common events (some of them may be uncontrollable)). Compared to the previous approaches [1, 23, 7], that are considering control objective expressed by means of languages, we present a methodology that solves the Supervisory Control Problem for a particular class of control objectives that particularly fit with the structure of the plant. More precisely, the problem of interest is to ensure the invariance of a set of particular configurations in the global system (or dually to solve the State Avoidance Control Problem). This problem is obviously simpler than the one considered in [1, 23, 7]. However, this allows to overcome some of the problems that arise in the previously mentioned papers. In particular, the objectives are, in general, not separable as assumed in [23]. It will not be necessary to built sub-parts of the system (i.e. to perform the product between some original components of the plant) as in [7], and we do not suppose that the shared events are controllable as in [1, 23].

Our methodology is the following: based on a particular decomposition of the forbidden set of states in terms of set products, we provide algorithms that locally solve the control problem (i.e. on each component without computing the whole system) and produce a global supervisor ensuring the desired property. This supervisor can be seen as an oracle that will activate/deactivate local supervisors according to the current configuration of the global system and some conditions that can be easily computed on line. Moreover, we make the necessary efforts to keep the structure of the plant in the global supervisor, hence improving the readability and the understanding of the supervisor effect as well as its memory storage. Note that the way we combine them makes that the global supervisor is the most permissive one. To illustrate this aspect, let us consider a plant composed of a press and an articulated arm, that has to place (to remove) a plate inside (from) the press. Each sub-system can be modeled in an independent way. One can see that global configurations such as "*the press is closed*" while "*the arm is located inside in the press*" have to be avoided during the execution of the system. In practice, most of the control objectives will be to prevent the system from reaching particular configurations of the global system, that can be locally decomposed according to each sub-system (in the press-arm example: "*the position inside the press*" for the articulated arm and "*closed position*" for the press). Separately, they do not correspond to a dangerous configuration. It is only when all sub-systems are simultaneously in these particular configurations that the global system is itself in a dangerous situation that has to be avoided. Coming back to our example, the intuitive idea of our method is then the following: we first compute a supervisor avoiding the press to be closed and a supervisor avoiding the arm to be inside the press. Finally, we combine the two supervisors in a way that avoids the illegal global configurations to be reachable. In other words, a supervisor will be used to coordinate the evolution between the components. Hence, the interactions between the various components G_i of a plant G will be reinforced by a supervisor that will allow or not events to be triggered in the different components of G.

The remainder of the paper is organized as follows: In section 2, we introduce the model and give a brief overview of the State Avoidance Control Problem (SACP). In section 3, we provide algorithms that, based on a particular decomposition of the set of forbidden configurations, locally solve the control problem (i.e. on each component without computing the whole system) and produce a global supervisor ensuring the desired property. In Section 3.2, we solve the SACP to the control of modular systems (i.e. when the sub-systems are mutually asynchronous). This technique can be used in parallel with the one of [7] to overhead the complexity issue). In Section 3.3, we extend these results to systems modeled by concurrent sub-systems sharing common

events. Finally, in Appendix A, we define algorithms that solve the Optimal Control Problem on a product system without expanding it.

2 Preliminaries

In this section, the main concepts and notations are defined. More definitions will be given in the following sections. The reader is referred to [5, 24] for any undefined concept.

2.1 The basic model.

The basic structures from which the plant will be built are Finite State Machines (FSMs) [5], that are defined as follows:

Definition 1 *A Finite State Machine (FSM) is defined by a 5-tuple $G = \langle \Sigma, \mathcal{X}, \mathcal{X}_o, \mathcal{X}_f, \delta \rangle$, where Σ is the finite alphabet of G. $\mathcal{X}$ is the finite set of states, $\mathcal{X}_o \subseteq \mathcal{X}$ is the set of initial states, whereas $\mathcal{X}_f$ is the set of final (marked) states of G, δ is the partial transition function defined over $\Sigma \times \mathcal{X} \longrightarrow \mathcal{X}$.* •

We can think of G as an uncontrolled plant that starts at $x_o \in \mathcal{X}_o$ and executes/generates a sequence of events that are accepted by δ. The notation $\delta(\sigma, x)!$ means that $\delta(\sigma, x)$ is defined, i.e., there is a transition labeled by an event σ out of state x in machine G. Likewise, $\delta(s, x)$ denotes the state reached by taking the sequence of events defined by trace s from state x in machine G. $\delta(x)$ denotes the active event set of x. Similarly, $\delta^{-1}(x)$ denotes the set of events that lead to x. The behavior of the system is described by the language generated by G (i.e. $\mathcal{L}(G) = \{s \in \Sigma^* \mid \exists x_o \in \mathcal{X}_o,\ \delta(s, x_o)!\}$).

Given an FSM G and a subset of event $A \subseteq \Sigma$, we denote by

$$Pre_A^G(E) = \{x' \in \mathcal{X} \mid \exists \sigma \in A, \delta(\sigma, x') \in E\} \cup E \tag{1}$$

the set of states x' of G from which E can be reached after at least one transition labelled by $\sigma \in A \subseteq \Sigma$. Based on (1), it is easy to show that:

Proposition 1 *Let G be an FSM and $E \subseteq \mathcal{X}$ a set of states, then*

1. $Pre_A^G(E) \cup Pre_A^G(E') = Pre_A^G(E \cup E')$.
2. $Pre_A^G(E) \cup Pre_{A'}^G(E) = Pre_{A \cup A'}^G(E)$ *and if* $A' \subseteq A$ *then* $Pre_{A'}^G(E) \subseteq Pre_A^G(E)$
3. *if* $E \subseteq E'$, *then* $Pre_A^G(E) \subseteq Pre_A^G(E')$
4. *Finally, the set of states from which E is reachable firing only events $\sigma \in A$, noted $Pre_A^{G^*}(E)$, is given by:*

$$Pre_A^{G^*}(E) = \bigcup_{i \in \mathbb{N}} Pre_A^{G^{(i)}}(E) \tag{2}$$

where $Pre_A^{G^{(i)}}(E) = Pre_A^G(Pre_A^{G^{(i-1)}}(E))$. ◇

Submachines. We now introduce the notion of submachines of an FSM [5]. This notion will be useful for control purposes.

Definition 2 *An FSM $H = \langle \Sigma_H, \mathcal{X}_H, \mathcal{X}_{H_o}, \mathcal{X}_{H_f}, \delta_H \rangle$ is a submachine of G, denoted $H \subseteq G$, if $\Sigma_H = \Sigma$, $\mathcal{X}_H \subseteq \mathcal{X}$, $\mathcal{X}_{H_o} \subseteq \mathcal{X}_o$, $\mathcal{X}_{H_f} \subseteq \mathcal{X}_f$, $\forall \sigma \in \Sigma_H, \forall x \in \mathcal{X}_H\ \ \delta_H(\sigma, x)! \Rightarrow (\delta_H(\sigma, x) = \delta(\sigma, x))$.* •

2.2 Review of the Supervisory Control Problem

Supervisory control theory deals with control of Discrete Event Systems. The idea of this theory is that the plant models the uncontrolled behavior, which is not fully satisfactory. Hence, its behavior has to be modified by means of a feedback control (named Supervisor) in order to achieve a given set of requirements that the initial DES did not satisfy [18]. In practice, one of the main control problem is the invariance control problem (or dually the state avoidance control problem), i.e. the supervisor has to control the plant so that the controlled plant remains in a safe set of states (or dually do not reach a set of forbidden states)[2].

Assume a plant G is given and modeled as an FSM and a set of states E, we recall how to synthesize a supervisor that will ensure the avoidance of E. As usual in the R & W framework, one have to take into account the possibility that certain events cannot be disabled by the supervisor. Therefore, some of the events in Σ are said to be uncontrollable (Σ_{uc}), i.e., their occurrence cannot be prevented by a controller, while the others are controllable (Σ_c). An uncontrollable event can for example model a change of sensor readings, the tick of a clock, etc (see [5] for more details). According to the duality between the event status, we first recall the definition of a *controllable submachine* [18].

Definition 3 *Let G be a FSM and H be a submachine of G, then H is controllable w.r.t. G and Σ_{uc}, whenever*

$$\forall x \in \mathcal{X}_H \subseteq \mathcal{X}, \forall \sigma \in \Sigma_{uc}, \sigma \in \delta(x) \Rightarrow \sigma \in \delta_H(\sigma). \quad \bullet$$

Intuitively, H is controllable w.r.t. G whenever it can obtained by only removing transitions labelled by a controllable event. Such a submachine can be obtained by means of a supervisor $\mathcal{S} = (S, \mathcal{X}'_o)$, which is given by a function $S : \mathcal{X} \to 2^{\Sigma_c}$, delivering the set of actions that are disabled in state x of G by the control, and the new set of valid initial states $\mathcal{X}'_o \subseteq \mathcal{X}_o$ (it could be the case, that in order to ensure an objective, we need to reduce $\mathcal{X}_o$).

Remark 1 *In a more general framework, S is a function from $\mathcal{L}(G)$ into 2^{Σ} [18]. However, for the control objectives we have in mind (ensuring the invariance/avoidance of a set of states), memory-less supervisors are sufficient since the resulting controlled system happens to be a sub-machine of the original plant.* $\diamond$

Write $\mathcal{S}/G$ for the system consisting of the initial plant G controlled by the supervisor $\mathcal{S}$ (see Figure 1).

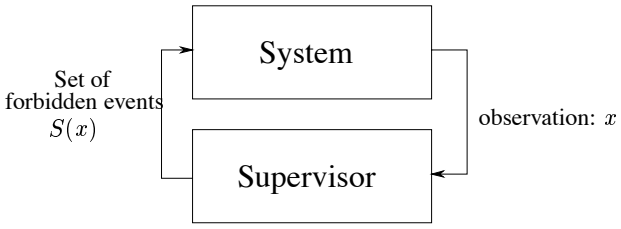


Figure 1: The closed-loop system

Given a memory-less supervisor, the closed-loop system $\mathcal{S}/G$ is actually a submachine of the initial plant such that the transition relation δ_s of $\mathcal{S}/G$ is obtained by restricting δ according to the control policy of the supervisor, i.e.

$$\forall x, x' \in \mathcal{X},\ \forall \sigma \in \Sigma,\ \delta_s(\sigma, x) = x' \Leftrightarrow \delta(\sigma, x) = x' \ \wedge \sigma \notin S(x)$$

and by keeping only the accessible part of this submachine.
In this paper, we focus on the State Avoidance Control Problem, namely how to avoid the plant to reach some particular states that are identified as illegal.

[2] In the literature, this invariance problem is often expressed using predicates over the states of the plant [17, 22, 27]. The control problem is then to force the system to remain in the states that satisfy the predicates.

The State Avoidance Control Problem *(SACP)*: *given G and E a set of states, the problem is to build a supervisor $\mathcal{S}$ such that (1) the traversed states of $\mathcal{S}/G$ do not belong to E and (2) $\mathcal{S}/G$ is the most permissive solution, i.e. for all other supervisor $\mathcal{S}'$ satisfying (1), $\mathcal{S}'/G$ is a submachine of $\mathcal{S}/G$ (i.e. $\mathcal{S}'/G \subseteq \mathcal{S}/G$). Such a supervisor is said to be maximal.*

In order to compute such a supervisor, we classically introduce two sets of states: the *weak forbidden set* of states and the *border set* that will be intensively used in the remainder of this paper.

Definition 4 *Given an FSM $G = \langle \Sigma, \mathcal{X}, \mathcal{X}_o, \mathcal{X}_f, \delta \rangle$, and a set of states $E \subseteq \mathcal{X}$, we denote by $\mathcal{I}(E)$ and $\mathcal{F}(E)$ the weak forbidden states and the border set of E, that are formally defined by:*

$$\mathcal{I}(E) = Pre^{G^*}_{\Sigma_{uc}}(E) = \{x \in \mathcal{X} \mid \exists s \in \Sigma^*_{uc}, \delta(s,x) \in E\} \tag{3}$$

$$\mathcal{F}(E) = Pre^{G}_{\Sigma}(\mathcal{I}(E)) \setminus \mathcal{I}(E) = \{x \in \mathcal{X} \setminus \mathcal{I}(E) \mid \exists \sigma \in \Sigma, s.t.\ \delta(\sigma,x) \in \mathcal{I}(E)\} \tag{4}$$

•

$\mathcal{I}(E)$ corresponds to the set of states from which it is possible to evolve into E by a trace of uncontrollable events, whereas $\mathcal{F}(E)$ corresponds to the set of states from which it is still possible to perform a control on G before evolving into $\mathcal{I}(E)$. The next proposition is adapted from [17].

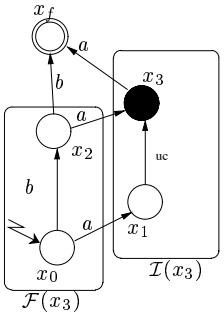
Proposition 2 *Given an FSM G and $E \subseteq \mathcal{X}$, a set of states to be forbidden by control, the supervisor $\mathcal{S}$ of G given by the pair $(S, \mathcal{X}'_o)$, such that*

$$\begin{array}{rcl} \forall x \in \mathcal{X},\ S(x) & = & \left\{ \begin{array}{l} \{\sigma \in \Sigma_c \mid \delta(\sigma,x) \in \mathcal{I}(E)\} \textit{ if } x \in \mathcal{F}(E) \\ \emptyset \textit{ Otherwise} \end{array} \right. \\ \mathcal{X}'_o & = & \mathcal{X}_o \setminus \mathcal{I}(E) \end{array} \tag{5}$$

ensures the invariance of $\mathcal{X} \setminus E$ in G and is maximal. ◇

If $\mathcal{X}'_o = \emptyset$, then it means that the BSACP has no solution (except the empty behavior). The corresponding supervisor $\mathcal{S} = (S, \emptyset)$ will be called the *trivial supervisor*. This notion will be useful in the next section.

Example 1 *Let us consider, the following FSM, where the state x_3 is a forbidden state, $\Sigma_c = \{a, b\}$ and $\Sigma_{uc} = \{uc\}$. The weak forbidden set of states $\mathcal{I}(\{x3\}) = \{x_3, x_1\}$, whereas the border of $\{x_3\}$ is given by $\mathcal{F}(\{x_3\}) = \{x_0, x_2\}$. The resulting Supervisor is given by $\mathcal{S} = (S, \{x_o\})$, where S si defined by: $S(x_o) = S(x_2) = \{a\}$ and for all the other states x, $S(x) = \emptyset$.*



Proposition 3 *[Modularity] let G be a plant and E_1, E_2 be two set of forbidden states. Let $\mathcal{S}_1 = (S_1, \mathcal{X}_{o_1})$ (resp. $\mathcal{S}_2 = (S_2, \mathcal{X}_{o_2})$) be the maximal supervisor ensuring the avoidance of E_1 (resp. E_2), then $\mathcal{S} = (S, \mathcal{X}'_o)$, s.t.*

$$\left\{ \begin{array}{l} \forall x \in \mathcal{X},\ S(x) = S_1(x) \cup S_2(x) \\ \mathcal{X}'_o = \mathcal{X}_{o_1} \cap \mathcal{X}_{o_2} \end{array} \right. \tag{6}$$

ensures the avoidance of $E_1 \cup E_2$ and is maximal. ◇

The result is due to [17] and basically comes from the fact that $\mathcal{I}(E_1 \cup E_2) = \mathcal{I}(E_1) \cup \mathcal{I}(E_2)$ and $\mathcal{F}(E_1 \cup E_2) = (\mathcal{F}(E_1) \cup \mathcal{F}(E_2)) \setminus (\mathcal{I}(E_1 \cup E_2))$.

3 Control of Structured Finite States Machines

3.1 The model and the SACP presentation

Let us consider a plant G modeled as a set of FSMs $G_i = \langle \Sigma_i, \mathcal{X}_i, \mathcal{X}_{oi}, \delta_i \rangle$. The global behavior of the system is given by $G = \langle \Sigma, \mathcal{X}, \mathcal{X}_o, \delta \rangle = G_1 \parallel \cdots \parallel G_n$, where the operation $\parallel$ is defined by:

Definition 5 *Let $G_i = (\Sigma_i, \mathcal{X}_i, \mathcal{X}_{o_i}, \mathcal{X}_{f_i}, \delta_i)$, $i = 1, 2$ be FSMs. The* synchronous product $G_1 \parallel G_2$ *of G_1 and G_2 is the FSM $G = (\Sigma, \mathcal{X}, \mathcal{X}_o, \delta)$ where $\Sigma = \Sigma_1 \cup \Sigma_2$, $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2$, $\mathcal{X}_o = \mathcal{X}_{o_1} \times \mathcal{X}_{o_2}$, $\mathcal{X}_f = \mathcal{X}_{f_1} \times \mathcal{X}_{f_2}$, and δ is defined by: for all $x = \langle x_1, x_2 \rangle \in \mathcal{X}$ and $\sigma \in \Sigma$*

$$\delta(\sigma, \langle x_1, x_2 \rangle) = \begin{cases} \langle \delta_1(\sigma, x_1), x_2 \rangle & \text{if } \sigma \in \Sigma_1 \setminus \Sigma_2 \\ \langle x_1, \delta_2(\sigma, x_2) \rangle & \text{if } \sigma \in \Sigma_2 \setminus \Sigma_1 \\ \langle \delta_1(\sigma, x_1), \delta_2(\sigma, x_2) \rangle & \text{if } \sigma \in \Sigma_1 \cap \Sigma_2 \\ \textit{Undefined} & \text{otherwise} \end{cases} \tag{7}$$

•

In the sequel, the states of G will be denoted by $x = \langle x_1, \ldots, x_n \rangle$, where each $x_i \in \mathcal{X}_i$ of G_i. For convenience, we will call a state, any element $x_i \in \mathcal{X}_i$ of a particular FSM G_i, and a *global state* a tuple of the form $\langle x_1, \ldots, x_n \rangle$, i.e. a "state" of the resulting FSM G. We call $\mathcal{X}$ the set of reachable global states of G.
Given the set of FSMs $(G_i)_{i \leq n}$ modeling G, we introduce the function

$$\begin{array}{rl} \textsc{In} : & \Sigma \to 2^n \\ & \sigma \to \{i \in [1..n] \,|\sigma \in \Sigma_i\} \end{array} \tag{8}$$

$\textsc{In}(.)$ is a function, which for each $\sigma \in \Sigma$ gives the set of components of G that share this event. Given $\sigma \in \Sigma$, $\textsc{In}(\sigma)^c$ denotes the complementary of $\textsc{In}(\sigma)$ w.r.t. $[1..n]$. $\textsc{In}(\sigma)^c$ corresponds to the set of indices i such that $\sigma \notin \Sigma_i$ (i.e. the components of G that do not have σ in their alphabet).
Moreover, we denote by Σ_s the set of shared events of G, i.e

$$\Sigma_s \;=\; \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j) = \{\sigma \in \Sigma | \; \exists i \neq j, \; \sigma \in \Sigma_i \cap \Sigma_j\}. \tag{9}$$

Note that $\sigma \in \Sigma_s$ if and only if $|\textsc{In}(\sigma)| \geq 2$. The set $\Sigma \setminus \Sigma_s$ represents the set of *local events* of G (σ is local if and only if $\exists i, \; \text{In}(\sigma) = \{i\}$).

Finally, $\forall \; i, \; 1 \leq i \leq n$, the set of controllable events in G_i is denoted by $\Sigma_{i,c}$, and the set of uncontrollable events by $\Sigma_{i,uc}$. Moreover,

$$\Sigma_c = \bigcup_i \Sigma_{i,c} \text{ and } \Sigma_{uc} = \bigcup_i \Sigma_{i,uc}$$

respectively correspond to the set of controllable/uncontrollable events of the whole system G.

The State Avoidance Control Problem (SACP) presentation. In the remainder of this paper, our aim is to solve the SACP for a set of forbidden global states E. In this framework, the purpose of a supervisor will be to coordinate the evolution between each sub-system in a way that they do no evolve into a set of illegal global states of G. In practice, this set that can be locally decomposed according to each sub-system. Separately, they do not correspond to a dangerous state. It is only when all sub-systems are simultaneously in these particular states that the system is itself in a dangerous situation that has to be avoided.
In more formal terms, given a product systems $G = G_1 \parallel \cdots \parallel G_n$, our aim is to solve the SACP for a set of forbidden global states E. This has to be done locally on each component of G. Hence, we first need to

decompose this set according to the structure of G. In fact, any set of global states E can be represented as a union of Cartesian products of local sets, i.e.

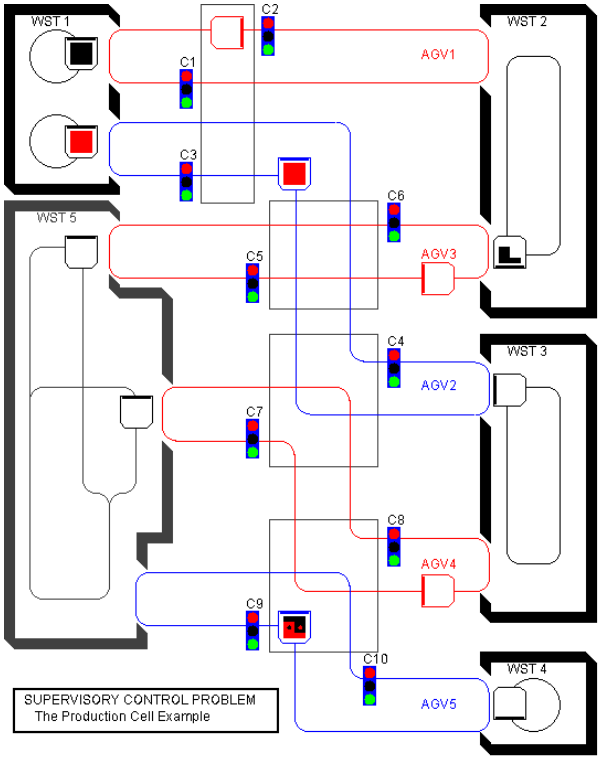
$$E = \bigcup_{1 \leq j \leq m} E^j, \tag{10}$$

where

$$\forall 1 \leq j \leq m,\ E^j = E_1^j \times \cdots \times E_n^j \text{ and } \forall 1 \leq i \leq n,\ E_i^j \subseteq \mathcal{X}_i \tag{11}$$

In the sequel, $E^j = E_1^j \times \cdots \times E_n^j$ (also noted $E^j = \Pi_{1 \leq i \leq n} E_j^i$ for short) will be called a product set. This decomposition in terms of product sets will be the basis for the expression of global sets that will have to be forbidden by control.

Example 2 *To illustrate this aspect, we consider the classical example of the flexible manufacturing cell control [11].This manufacturing cell is composed of five workstations (three processing workstations, a part-receiving station (Work Station 1) and one completed parts station (Work Station 4)).*



Five Automated Guided Vehicles (AGV's) transport materials between pairs of stations, passing through conflict zones shared with other AGV's. We assume that the controller receives signals from the AGV's indicating their current positions in the manufacturing cell. We also assume that we can stop the AGV's before they enter in some conflict zones (C_i events). The control synthesis problem is to coordinate the movement of the various AGV's in order to avoid collisions in the conflict zones (i.e., *it is required that all AGV's be controlled so that each zone be occupied by no more than one AGV). Each components of the system can be modeled by an FSM (AGV_i, $i = 1, \ldots, 5$ for the Automated Guided Vehicles, and WST_i, $i = 1, \ldots, 5$ for the workstations). The global system is given by the concurrent system $G = AGV_1 \parallel \cdots \parallel AGV_5 \parallel WST_1 \parallel \cdots \parallel WST_5$ (the manufacturing cell is represented by an FSM with more than 3.10^7 global states). Note that, in this example, $\Sigma_c = \{C_i, i \leq 10\}$ and $\Sigma_s \subseteq \Sigma_{uc}$*

Now from a control point of view, the goal of a supervisor will be to avoid the plant to be in particular global states that belongs to $E = \cup_{1 \leq i \leq 4} Zone_i$, such that

$$\begin{aligned}
Zone_1 &= E_1^1 \times E_1^2 \times \mathcal{X}_{AGV_3} \times \mathcal{X}_{AGV_4} \times \mathcal{X}_{AGV_5} \times \Pi_{1 \leq i \leq j} \mathcal{X}_{WST_i} \\
Zone_2 &= \mathcal{X}_{AGV_1} \times E_2^2 \times E_2^3 \times \mathcal{X}_{AGV_4} \times \mathcal{X}_{AGV_5} \times \Pi_{1 \leq i \leq j} \mathcal{X}_{WST_i} \\
Zone_3 &= \mathcal{X}_{AGV_1} \times E_3^2 \times \mathcal{X}_{AGV_3} \times E_3^4 \times \mathcal{X}_{AGV_5} \times \Pi_{1 \leq i \leq j} \mathcal{X}_{WST_i} \\
Zone_4 &= \mathcal{X}_{AGV_1} \times \mathcal{X}_{AGV_2} \times \mathcal{X}_{AGV_2} \times E_4^4 \times E_4^5 \times \Pi_{1 \leq i \leq j} \mathcal{X}_{WST_i}
\end{aligned}$$

where E_j^i is the set of states of AGV_i modeling the fact the i^{th} Automated Guided Vehicles is located inside the j^{th} conflict zone (i.e. $Zone_j$ is a product set encoding all the global states of the plant in which two AGVs are located inside the conflict zone j whatever the position of the other components of the plant. ◇

Remark 2 *In [15], a more restrictive state-based approach was considered for product systems. The control objective was assumed to be separable, i.e. the set of forbidden global states was given by* $E = \bigcup_{1\leq j\leq n} E^j$, *where* $\forall 1 \leq j \leq n,\ E^j = \mathcal{X}_1 \times \cdots \times \mathcal{X}_{j-1} \times E_j \times \mathcal{X}_{j+1} \times \cdots \times \mathcal{X}_n$. *Roughly, the control was to avoid each sub-system* G_j *to enter local state of* E_j *whatever the position of the other subsystems.* ◇

Based on decomposition (10) and (11), we now define the $Pre_A^G(.)$ operator according to the local operators $Pre_A^{G_i}(.)$.

Proposition 4 *Let* $G = G_1 \parallel \cdots \parallel G_n$ *be a structured system and* E *be a set of states as defined in (10) and (11), then*

$$Pre^G_{\{\sigma\}}(E) = \bigcup_{1\leq j\leq n} \left\{ \prod_{i\in \text{IN}(\sigma)} Pre^{G_i}_{\{\sigma\}}(E_i^j) \times \prod_{k\in \text{IN}(\sigma)^c} E_k^j \right\} \text{ and } Pre^G_A(E) = \bigcup_{\sigma\in A} Pre^G_{\{\sigma\}}(E) \qquad (12)$$

Proof : We first make the proof for a product set $E^j = E_1^j \times \cdots \times E_n^j$. Let us consider a global state $x = \langle x_1, \ldots, x_n\rangle \in Pre^G_{\{\sigma\}}(E^j)$. According to (1), we have either $\delta(\sigma, x) \in E^j$ or $x \in E^j$. Assume that $\delta(\sigma, x) \in E^j$ (the other case is trivial). Now, let $x' = \langle x'_1, \ldots, x'_n\rangle$ be such that $x' = \delta(\sigma, x)$. According to Definition 5, it means that $\forall i \in \text{IN}(\sigma),\ \delta_i(\sigma, x_i) = x'_i$ (i.e. $x_i \in Pre^{G_i}_{\{\sigma\}}(x'_i)$) and $\forall k \notin \text{IN}(\sigma),\ x_k = x'_k$. Hence, we can deduce that

$$x \in \left(\Pi_{i\in \text{IN}(\sigma)} Pre^{G_i}_{\{\sigma\}}(x'_i)\right) \times \left(\Pi_{k\notin \text{IN}(\sigma)}\{x'_k\}\right) \subseteq \left(\Pi_{i\in \text{IN}(\sigma)} Pre^{G_i}_{\{\sigma\}}(E_i^j)\right) \times \left(\Pi_{k\notin \text{IN}(\sigma)} E_k^j\right)$$

The last inclusion is due to Proposition 1, *item (3)*.
Conversely, let $x = \langle x_1, \ldots, x_n\rangle \in \left(\Pi_{i\in \text{IN}(\sigma)} Pre^{G_i}_{\{\sigma\}}(E_i^j)\right) \times \left(\Pi_{k\notin \text{IN}(\sigma)} E_k^j\right)$. For $i \in \text{IN}(\sigma)$, let $x'_i \in E_i^j$ be such that $x_i \in Pre^{G_i}_{\{\sigma\}}(E_i^j)$. We then have $\forall i \in \text{IN}(\sigma),\ \delta_i(\sigma, x_i) = x'_i$ (we omit the case $x_i = x'_i$). Now, for $k \notin \text{IN}(\sigma)$, we note $x'_k = x_k$, and let us call x' the global state $x' = \langle x'_1, \ldots, x'_n\rangle$. We then have $\delta(\sigma, x) = x'$. Hence, from (1), we can deduce that $x \in Pre^G_{\{\sigma\}}(x')$ and therefore $x \in Pre^G_{\{\sigma\}}(E^j)$.
The generalization to an arbitrary number of product is immediate (Proposition 1, *item (1)*). Finally, the fact that $Pre^G_A(E) = \bigcup_{\sigma\in A} Pre^G_{\{\sigma\}}(E)$ is due to Proposition 1, *item (2)*. ◇

In the next sections, we provide a methodology that locally solves the control problem (i.e. on each component of G without computing the whole system) but produces a global supervisor ensuring the global avoidance of E. In Section 3.2, we first focus on the control of product DES, i.e. systems composed of components not sharing common events.

3.2 SACP for product systems: the case $\Sigma_s = \emptyset$

In this section, we assume that the plant that has to be controlled is modeled as a collection of asynchronous FSMs $(G_i)_{i\leq n}$ (with $\forall i \neq j,\ \Sigma_i \cap \Sigma_j = \emptyset$). Our aim is to solve the state avoidance control problem for a set of forbidden global states of the form $E = \cup_{1\leq j\leq m} E^j$, as described in Section 3.1. We first focus on the case where the set of forbidden global states E is reduced to a product set of the form $E_1 \times \cdots \times E_n$ (with $E_i \subseteq \mathcal{X}_i$ of G_i).
Based on Definition 4, one can compute $\mathcal{I}(E_i)$ and $\mathcal{F}(E_i)$, $i \leq n$ as well as $\mathcal{S}_i = (S_i, \mathcal{X}'_{o_i})$ the maximal supervisors avoiding states in E_i to be reachable in G_i (resp. G_2). These supervisors are computed according to (5). At this point, one can see that $G' = (\mathcal{S}_1/G_1) \parallel \cdots \parallel (\mathcal{S}_n/G_n)$ would solve the problem (i.e. the global states that belong to $E_1 \times \cdots \times E_n$ are not reachable in G') but would not be maximal (i.e. the control policy is too restrictive). This basically comes from the fact that this control objective is not separable (according to the definition of [23]). However, the next propositions (5 and 6) show that by combining the informations previously computed, it is possible to obtain a supervisor that is maximal.

Proposition 5 *Let $G = G_1 \parallel \cdots \parallel G_n$ and the set of forbidden states $E = E_1 \times \cdots \times E_n$, then the set of weak forbidden global states is given by*

$$\mathcal{I}(E) = \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n) \tag{13}$$

and, the border set is given by:

$$\mathcal{F}(E) = \bigcup_{1 \leq i \leq n} \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_{i-1}(E_{i-1}) \times \mathcal{F}_i(E_i) \times \mathcal{I}_{i+1}(E_{i+1}) \times \cdots \times \mathcal{I}_n(E_n) \tag{14}$$

where $\mathcal{I}_i(E_i)$ (resp. $\mathcal{F}_i(E_i)$) correspond the weak forbidden set of states (resp. border set) of E_i in G_i.

In order to prove Proposition 5, we need the following lemma, which is a direct consequence of Proposition 4, when G is a product system.

Lemma 1 *Let $G = G_1 \parallel \cdots \parallel G_n$ be a product system, $E = E_1 \times \cdots \times E_n$ be a set of states of G, and $A \subseteq \Sigma_i$, then $Pre_A^G(E) = E_1 \times \cdots \times E_{i-1} \times Pre_A^{G_i}(E_i) \times E_{i+1} \times \cdots \times E_n$.*

Proof of Proposition 5: Let $x = \langle x_1, \cdots, x_n \rangle \in \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n)$. As $x_i \in \mathcal{I}_i(E_i)$, $\exists s_i \in \Sigma_{i,uc}^*$, s.t. $\delta_i(s_i, x_i) = x_i' \in E_i$. Hence, according to Definition 5, we have that $\delta(s_1 \cdots s_n, x) = \langle x_1', \cdots, x_n' \rangle \in E_1 \times \cdots \times E_n$, which implies that $x \in \mathcal{I}(E_1 \times \cdots \times E_n) = \mathcal{I}(E)$.

Conversely, let $x = \langle x_1, \cdots, x_n \rangle \in \mathcal{I}(E)$. Then it exists $s \in \Sigma_{uc}^*$, s.t. $\delta(s, x) = x' = \langle x_1', \cdots, x_n' \rangle \in E$. Now let us consider the traces $s_i = P_i[s] \in \Sigma_{i,uc}$, where P_i is the natural projection over $\Sigma_{i,uc}$ (see e.g. [5]). Now, as $\Sigma_{i,uc} \cap \Sigma_{j,uc} = \emptyset$ and based on Definition 5, we also have that $\delta(s_1 \cdots s_n, x) = x'$, which entails that $\forall i,\ \delta_i(s_i, x_i) = x_i' \in E_i$. According to the definition of the weak forbidden set, we then have $\forall i,\ x_i \in \mathcal{I}_i(E_i)$ (as $x' \in E_1 \times \cdots \times E_n$). Finally we obtain $x \in \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n)$, and overall $\mathcal{I}(E) = \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n)$.

Let us now prove that $\mathcal{F}(E) = \bigcup_{1 \leq i \leq n} \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_{i-1}(E_{i-1}) \times \mathcal{F}_i(E_i) \times \mathcal{I}_{i+1}(E_{i+1}) \times \cdots \times \mathcal{I}_n(E_n)$. According to (4), $\mathcal{F}(E)$ is defined by

$$\mathcal{F}(E) \quad = \quad Pre_\Sigma^G(\mathcal{I}(E)) \setminus \mathcal{I}(E)$$

According to Proposition 1, *item (2)*, this can be rewritten:

$$= \quad \bigcup_{1 \leq i \leq n} Pre_{\Sigma_i}^G(\mathcal{I}(E)) \setminus \mathcal{I}(E) = \bigcup_{1 \leq i \leq n} Pre_{\Sigma_i}^G(\mathcal{I}(E_1) \times \cdots \times \mathcal{I}(E_n)) \setminus \mathcal{I}(E)$$

Now, based on Lemma 1, this entails that

$$\begin{aligned} \mathcal{F}(E) \quad &= \quad \bigcup_{1 \leq i \leq n} [\mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_{i-1}(E_{i-1}) \times Pre_{\Sigma_i}^{G_i}(\mathcal{I}_i(E_i)) \times \mathcal{I}_{i+1}(E_{i+1}) \times \cdots \times \mathcal{I}_n(E_n)] \setminus \mathcal{I}(E) \\ &= \quad \bigcup_{1 \leq i \leq n} \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_{i-1}(E_{i-1}) \times (Pre_{\Sigma_i}^{G_i}(\mathcal{I}_i(E_i)) \setminus \mathcal{I}_i(E_i)) \times \mathcal{I}_{i+1}(E_{i+1}) \times \cdots \times \mathcal{I}_n(E_n) \end{aligned}$$

Finally, as $Pre_{\Sigma_i}^{G_i}(\mathcal{I}_i(E_i) \setminus \mathcal{I}_i(E_i) = \mathcal{F}_i(E_i)$ (according to (4) applied on G_i and E_i), we obtain

$$\mathcal{F}(E) \quad = \quad \bigcup_{1 \leq i \leq n} \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_{i-1}(E_{i-1}) \times \mathcal{F}_i(E_i) \times \mathcal{I}_{i+1}(E_{i+1}) \times \cdots \times \mathcal{I}_n(E_n)$$

◇

The interest of this methods is that $\mathcal{I}(E)$ and $\mathcal{F}(E)$ are computed without building G, i.e. they may be computed according to the local border/weak forbidden sets. Further, based on $\mathcal{I}(E)$ and $\mathcal{F}(E)$, a maximal supervisor $\mathcal{S}_E$ ensuring the avoidance of E in G can be computed as in Proposition 2. Note that in this case, even if all the computations have been done locally, we obtain a global supervisor that does not reflect the structure of the plant to be controlled. Moreover, it is not interesting to effectively compute these two sets, since they may be of the size of the global system and then unfeasible to compute. In order to avoid to store in memory large set of states (i.e. $\mathcal{I}(E)$ and $\mathcal{F}(E)$), we prefer to store in memory local set of states (i.e. the $\mathcal{I}_i(E_i)$ and $\mathcal{F}_i(E_i)$) and to perform some on-line computations. This is the aim of the next next proposition.

Proposition 6 *Let $G = G_1 \parallel \cdots \parallel G_n$ with $G_i = \langle \Sigma_i, \mathcal{X}_i, \mathcal{X}_{o_i}, \mathcal{X}_{f_i}, \delta_i \rangle,\ i = 1, \ldots, n$ with $\Sigma_i \cap \Sigma_j = \emptyset, i \neq j$ and $E = E_1 \times \cdots \times E_n$. Consider the sets $\mathcal{F}(E_i),\ \mathcal{I}(E_i)$ as defined in Definition 4 as well as the corresponding maximal supervisors $\mathcal{S}_i = (S_i, \mathcal{X}'_{o_i})$ (possibly trivial).*
Let $\mathcal{S}_E = (S, \mathcal{X}_{o_E})$ be such that

$$\forall x = \langle x_1, \ldots, x_n \rangle,\ S_E(x) = \begin{cases} S_1(x_1) \text{ if } \forall j \neq 1,\ x_j \in \mathcal{I}_j(E_j) \\ \ldots \\ S_i(x_i) \text{ if } \forall j \neq i,\ x_j \in \mathcal{I}_j(E_j) \\ \ldots \\ S_n(x_n) \text{ if } \forall j \neq n \in \mathcal{I}_j(E_j) \\ \\ \emptyset \text{ Otherwise} \end{cases} \tag{15}$$
$$\mathcal{X}_{o_E} = \{\langle x_{o_1}, \ldots, x_{o_n} \rangle \in \Pi_{i \leq n} \mathcal{X}_{o_i} \ / \ \exists i,\ x_{o_i} \in \mathcal{X}'_{o_i}\}$$

The supervisor $\mathcal{S}_E$ ensures the avoidance of $E_1 \times \ldots \times E_n$ in $G = G_1 \parallel \cdots \parallel G_n$ and is maximal.

Proof : Let S be the maximal supervisor ensuring the avoidance of E w.r.t. G computed according to Proposition 2. Let $x = \langle x_1, \cdots, x_n \rangle \in \mathcal{X}$, then

$$S(x) = \begin{cases} \{\sigma \in \Sigma_c \mid \delta(\sigma, x) \in \mathcal{I}(E)\} \text{ if } x \in \mathcal{F}(E)\} \\ \emptyset \end{cases} = \begin{cases} \bigcup_{1 \leq i \leq n} \{\sigma \in \Sigma_{i,c} \mid \delta(\sigma, x) \in \mathcal{I}(E)\} \text{ if } x \in \mathcal{F}(E)\} \\ \emptyset \end{cases} \tag{16}$$

According to (13), this is equivalent to

$$S(x) = \bigcup_{1 \leq i \leq n} \begin{cases} \{\sigma \in \Sigma_{i,c} \mid \delta(\sigma, x) \in \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n)\} \text{ if } x \in \mathcal{F}(E)\} \\ \emptyset \end{cases} \tag{17}$$

Now if $\delta(\sigma, x) \in \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n)$ and if $\sigma \in \Sigma_{i,c}$, it means that $\delta_i(\sigma, x_i) \in \mathcal{I}_i(E_i)$ and $\forall j \neq i$, $x_j \in \mathcal{I}_j(E_j)$ (as the only component that may evolve via σ is G_i). Hence, (17) is equivalent to

$$S(x) = \bigcup_{1 \leq i \leq n} \begin{cases} \{\sigma \in \Sigma_{i,c} \mid \delta_i(\sigma, x_i) \in \mathcal{I}_i(E_i)\} \text{ if } x \in \mathcal{F}(E) \text{ and } \forall j \neq i,\ x_j \in \mathcal{I}_j(E_j)\} \\ \emptyset \end{cases} \tag{18}$$

Now if $x \in \mathcal{F}(E)$ and $\delta_i(\sigma, x_i) \in \mathcal{I}_i(E_i)$ and $\forall j \neq i,\ x_j \in \mathcal{I}_j(E_j)$, because of (14), we have that $x_i \in \mathcal{F}_i(E_i)$ and $\forall j \neq i,\ x_j \in \mathcal{I}_j(E_j)$, which entails that (18) is equivalent to

$$\begin{aligned} S(x) &= \bigcup_{1 \leq i \leq n} \begin{cases} \{\sigma \in \Sigma_{i,c} \mid \delta_i(\sigma, x_i) \in \mathcal{I}_i(E_i)\}) \text{ if } x_i \in \mathcal{F}(E_i) \text{ and } \forall j \neq i,\ x_j \in \mathcal{I}_j(E_j)\} \\ \emptyset \end{cases} \\ &= \bigcup_{1 \leq i \leq n} \{S_i(x_i) \text{ if } \forall j \neq i,\ x_j \in \mathcal{I}_j(E_j)\} \end{aligned} \tag{19}$$

Finally, one can see that for $i \neq i'$ $\{S_i(x_i) \text{ if } \forall j \neq i,\ x_j \in \mathcal{I}_j(E_j)\} \cap \{S_{i'}(x_{i'}) \text{ if } \forall j \neq i',\ x_j \in \mathcal{I}_j(E_j)\} = \emptyset$, hence we have $S(x) = S_E(x)$. Dealing with the initial states of the controlled system, it is easy to show that $\mathcal{X}_{o_E} = \mathcal{X}_o \setminus \mathcal{I}(E)$. ◇

let S_E be a supervisor as in Proposition 6, one can see that at most one local supervisor is active at a time. It is the one for which the sub-plant has evolved in its border set of states when the other sub-plants are in a weak forbidden state.

The interest of such a method is that the supervisor $\mathcal{S}_E$ is locally computed according to the local supervisors $\mathcal{S}_i$. Therefore, this method avoids to build the whole system and the computation of $\mathcal{S}$ on the resulting system. Hence, it reduces the complexity of the algorithm (see the next paragraph) as well as the memory storage of the supervisor. Moreover, the supervisor itself somehow keeps the structure of the plant as it is represented as a

collection of local supervisors. Hence, the way $\mathcal{S}_E$ is built may improve the readability and the understanding of the control effect. However, on-line computations are still needed to know whether an event has to be disabled or not, i.e. to be activated, a local supervisor need to have some information coming from the different components of the system. In particular it has to know whether the other components are in a weak forbidden (or border) state or not.

Example 3 *Consider the two following FSMs G_1 and G_2, the objective is to avoid the global state $x = \langle x_1, x'_1 \rangle$ to be reachable in $G_1 \parallel G_2$. We first compute the maximal supervisors $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) ensuring the avoidance of x_1 in G_1 (resp. x'_1 in G_2). The action of the supervisors is represented by dash arrows in the Figure 2(b). Now,*

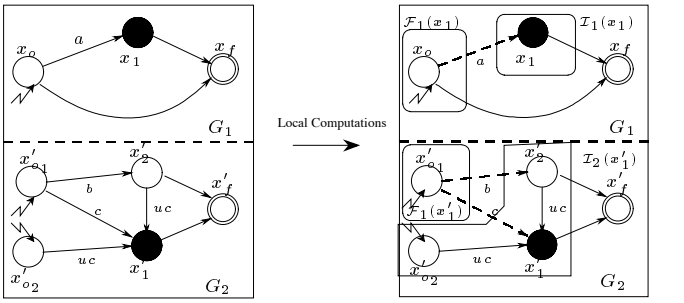


Figure 2: A simple Example

following Proposition 6, the supervisor $\mathcal{S} = (S_x, \mathcal{X}_{o_{\{x\}}})$ is given by

- $\mathcal{X}_{o_{\{x\}}} = \{\langle x_o, x'_{o_1} \rangle, \langle x_o, x'_{o_2} \rangle\}$
- $S_x(\langle x_o, x'_{o_2} \rangle) = S_x(\langle x_o, x'_1 \rangle) = S_x(\langle x_o, x'_2 \rangle) = \{a\}$ *because G_2 is in $\mathcal{I}_2(x'_1)$. Hence, $\mathcal{S}_1$ becomes active. $S(\langle x_1, x'_{o_1} \rangle) = \{b, c\}$ (G_1 is in $\mathcal{I}_1(x_1)$. Therefore $\mathcal{S}_2$ is active). For all the other states, $S(x0 = \emptyset$.* $\diamond$

Let us now present the main theorem of this section. It shows hows to avoid a set of global states E (as defined by (10)) to be reachable in a product system.

Theorem 1 *Let $G = G_1 \parallel \cdots \parallel G_n$ be the plant to be controlled and a set $E = \bigcup_{1 \leq j \leq m} E^j$ where $\forall 1 \leq j \leq m,\ E^j = E^j_1 \times \cdots \times E^j_n$ and $E^j_i \subseteq \mathcal{X}_i$ for $1 \leq i \leq n$. Let $\mathcal{S}_{E^j} = (S_{E^j}, \mathcal{X}_{o_{E^j}})$ be the maximal supervisors computed w.r.t. G and E^j, then $\mathcal{S}_E = (S, \mathcal{X}_{o_E})$ s.t.*

$$\begin{cases} \forall x = \langle x_1, \ldots, x_n \rangle \in \mathcal{X},\ S(x) = S_{E^1}(x) \cup \cdots \cup S_{E^m}(x) \\ \mathcal{X}_{o_E} = \mathcal{X}_{o_{E^1}} \cap \cdots \cap \mathcal{X}_{o_{E^m}} \end{cases} \tag{20}$$

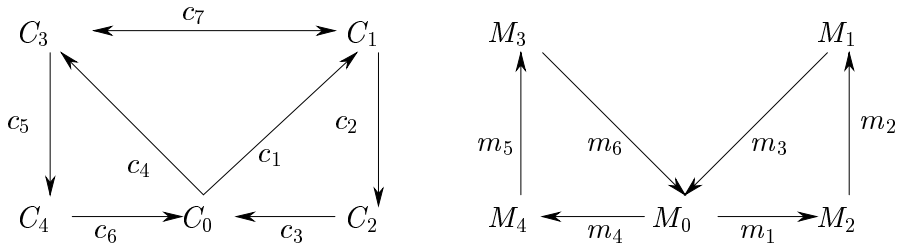
ensures the avoidance E in G and is maximal.

Proof : Based on Prop. 6, $\forall 1 \leq j \leq m, \mathcal{S}_{E^j}$ is a controllable and maximal supervisor with respect to $G,\ \Sigma_{uc}$ and the control objective E^j. The modularity result of Prop. 3 gives the results. $\diamond$

Finally, note that this kind of control objectives cannot be efficiently solved using the method presented in [6, 10] since G has to be built (i.e. the FSM $G_1 \parallel \cdots \parallel G_n$ has to be computed), as the objectives concern the whole system. However, our method can be used in complement to the one of [6, 10] whenever the control objective does not concern the whole objective (i.e our method can be used to compute the controller on the sub-machines for which forbidden interactions are required). The global supervisor can then be inferred using the methods of [6, 10].

The complexity aspect. Let us now discuss about complexity of the control synthesis phase. Given an FSM with N states and a set of states to be avoided by control, the complexity of the controller synthesis phase for the SACP is in $\mathcal{O}(N|\Sigma|)$. Let us now consider a system G of the form $G_1 \parallel \cdots \parallel G_n$ where each G_j contains N states (we assume that $max_i(|\Sigma_i|) = k$). The number of states of G is in $\mathcal{O}(N^n)$. Hence, using classical techniques, the state avoidance control problem is in $\mathcal{O}(n.k.N^n))$. In our case, given a product set $E = E_1 \times \cdots \times E_n$, to be avoided by control, we locally compute the supervisor on each local component of G. Hence the global complexity is in $\mathcal{O}(n.k.N)$. More generally, given a set of forbidden global states E, then this set can be decomposed into a union of product sets. Assume that E is composed of m product sets, then the supervisor computation complexity is in $\mathcal{O}(m.n.k.N)$. However one has also to take into account the computations that have to be done on-line when controlling the plant. Indeed, deciding which supervisor have to be activated given one global state, is done at execution time. This can be done in $\mathcal{O}(m.n.N)$, which is an acceptable complexity. However, if the objectives are not well structured, the number m of product sets to be forbidden can be important. In this case, the on-line computation may be important as well and one has to find a good set of states representation as Binary Decision Diagrams [4] for example[3].

Example 4 *Let us consider the well known Cat & Mouse example [18]. The cat and the mouse movements are respectively modeled by the FSMs* CAT *and* MOUSE *for which the states are respectively C_i and M_i, for $i = 0 \cdots 4$ corresponding to the room in which the animals are (the events c_i and m_i model the movements of the animal from one room to another). The goal of supervisor is to avoid the cat and the mouse to be at the same time in the same room.*



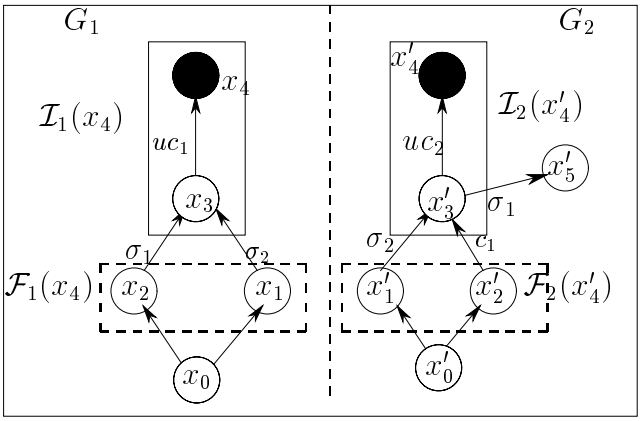
Following (10), the set of forbidden global states can be decomposed in 5 product sets $E_i = \langle C_i, M_i \rangle$, each one modeling the fact that the cat and the mouse are in the same room. Now according to Theorem 1, it is sufficient to compute the maximal supervisors $\mathcal{S}_{E_i}$ ensuring the avoidance of E_i, $i = 0, \ldots 4$ (C.f. Proposition 6), i.e. each supervisor $\mathcal{S}_{E_i}$ ensures that the cat and the mouse are not in the room i at the same time. Finally, by combining the action of each supervisor (C.f. Theorem 1), we obtain a global supervisor ensuring the desired property. ◇

3.3 SACP for Concurrent Systems: the case $\Sigma_s \neq \emptyset$

Let us now consider a plant $G = G_1 \parallel \cdots \parallel G_n$, with $\Sigma_s \neq \emptyset$ (i.e. the FSMs modeling the plant share common events). Compared to Section 3.2, it will not be possible to express the global supervisor as a function of the local supervisors (this is due to the shared events). However, the methodology remains the same. Given a set of global states E to be forbidden, local forbidden and border sets will be computed and the supervisor will be computed according to these sets of states. Hence, the FSM G will never be built.
Given a set of global states E of G, as defined in section 3.1, that has to be forbidden by control, we first need to extend the weak forbidden set of states and the border sets definitions in order to take into account the fact that the different components of G share common events. Indeed, the weak forbidden set of states and border sets of each G_i computed as in Section 3.2, would lead to a supervisor that does not ensure the desired property (sometimes too restrictive, sometimes too permissive). To illustrate this point, let us consider the following example:

[3] see e.g. [27, 14] for controller synthesis tools based on a BDD implementation.

Example 5



Let us consider a system $G = G_1 \parallel G_2$, where G_1 and G_2 are sharing the events $\Sigma_s = \{\sigma_1, \sigma_2\}$ and $\Sigma_{uc} = \{uc_1, uc_2\}$. Assume we want to avoid the global state $\langle x_4, x'_4\rangle$ to be reachable in G. Let us consider the supervisor $\mathcal{S}$ computed as in Section 3.2, according to the sets $\mathcal{I}_i$ and $\mathcal{F}_i$.
It is easy to show that $\mathcal{S}$ does not solve the problem since in $\langle x_1, x'_1\rangle$, $\mathcal{S}(\langle x_1, x'_1\rangle) = \emptyset$. However by triggering the shared event σ_2, G will evolve into the global state $\langle x_3, x'_3\rangle$, that happens to be a weak forbidden global state.
A contrario, in the global state $\langle x_2, x'_3\rangle$, $\mathcal{S}$ disables event $\sigma 1$ (because $x_2 \in \mathcal{F}_1(x_4)$ and $x'_3 \in \mathcal{I}_2(x'_4)$). However, this event does not have to be disabled since by triggering it, the reached global state is $\langle x_3, x'_5\rangle$, which is not a forbidden global state. $\diamond$

3.3.1 Local forbidden and border sets

Given a component of G, say G_i and a set of states $E_i \subseteq \mathcal{X}_i$, we first introduce the *local weak forbidden set of states with respect to E_i*, denoted by $\mathcal{I}_{i,loc}(E_i)$. It is defined by:

$$\mathcal{I}_{i,loc}(E_i) = {Pre^{G_i}}^*_{\Sigma_{i,uc}\setminus\Sigma_s}(E_i) = \{x \in \mathcal{X}_i |\ \exists s \in (\Sigma_{i,uc} \setminus \Sigma_s)^*,\ \delta_i(s,x) \in E_i\} \tag{21}$$

$\mathcal{I}_{i,loc}(E_i)$ represents the set of states of G_i from which it is possible to evolve into E_i by triggering a sequence of events which are uncontrollable and only local (i.e. with no shared event).
Now, dealing with the local border set, the control policy differs according to whether an event is shared or not (See example 5). This is formalized as follows:

$$\begin{aligned} \mathcal{F}_{i,loc}(E_i) &= Pre^{G_i}_{\Sigma_i\setminus\Sigma_s}(\mathcal{I}_{i,loc}(E_i)) \setminus \mathcal{I}_{i,loc}(E_i) \\ &= \{x \in (\mathcal{X}_i \setminus \mathcal{I}_{i,loc}(E_i)) \mid \exists \sigma \in \Sigma_i \setminus \Sigma_s,\ \delta_i(\sigma, x) \in \mathcal{I}_{i,loc}(E_i)\} \end{aligned} \tag{22}$$

$$\forall \sigma \in \Sigma_s,\ \mathcal{F}_i^{\sigma}(E_i) = Pre^{G_i}_{\{\sigma\}}(\mathcal{I}_{i,loc}(E_i)) = \{x \in \mathcal{X}_i |\ \delta_i(\sigma, x) \in \mathcal{I}_{i,loc}(E_i)\} \tag{23}$$

$\mathcal{F}_{i,loc}(E_i)$ corresponds to the states that lead to the local weak forbidden set via a local event. For the shared events, we specialize the border set to a particular event $\sigma \in \Sigma_s$ (one for each shared event). $\mathcal{F}_i^{\sigma}(E_i)$ corresponds to the set that lead to the weak local forbidden set via the event σ.

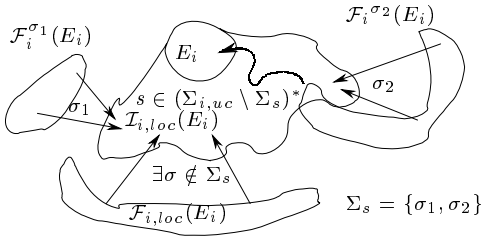
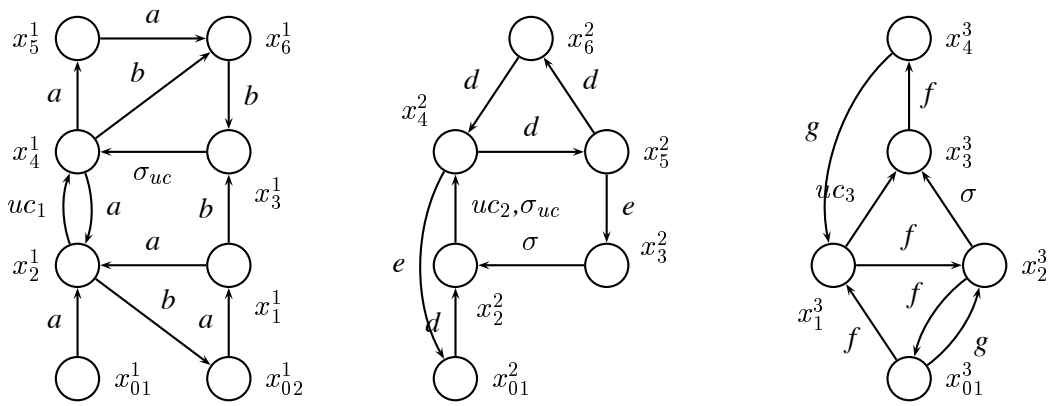


Figure 3: Sets $\mathcal{I}_{i,loc}(E_i), \mathcal{F}_{i,loc}(E_i)$ and $\mathcal{F}_i^{\sigma}(E_i)$ of G_i w.r.t. E_i

It may be the case that $\mathcal{I}_{i,loc}(E_i) \cap \mathcal{F}_i^{\sigma}(E_i) \neq \emptyset$. Finally, note that the distinction made between (22) and (23) is important since by removing a shared event in a component, we may avoid some other components to evolve through this event.

Example 6 *To illustrate these new sets of states, we consider the following example. The system G we want to control is given by the three following FSMs G_1, G_2, G_3, where $\Sigma_s = \{\sigma_{uc}, \sigma\}$ and $\Sigma_{uc} = \{uc_1, uc_2, uc_3, \sigma_{uc}\}$.*



The set of forbidden configurations is given by the union of two product sets: $E = E^1 \cup E^2$ with

$$E^1 = \{x_1^1, x_4^1\} \times \{x_4^2\} \times \{x_3^3\} = \{\langle x_1^1, x_4^2, x_3^3\rangle, \langle x_4^1, x_4^2, x_3^3\rangle\}$$

$$E^2 = \{x_3^1, x_2^1\} \times \{x_2^2, x_3^2\} \times \{x_2^3, x_3^3\}$$

$$= \{\langle x_3^1, x_2^2, x_2^3\rangle, \langle x_3^1, x_2^2, x_3^3\rangle, \langle x_3^1, x_3^2, x_2^3\rangle, \langle x_3^1, x_3^2, x_3^3\rangle, \langle x_2^1, x_2^2, x_2^3\rangle, \langle x_2^1, x_2^2, x_3^3\rangle, \langle x_2^1, x_3^2, x_2^3\rangle, \langle x_2^1, x_3^2, x_3^3\rangle\}$$

The (local) weak forbidden states and the (local) border sets are summarized in Table 1.

	$E_1^1 = \{x_1^1, x_4^1\}$	$E_2^1 = \{x_4^2\}$	$E_3^1 = \{x_3^3\}$	$E_1^2 = \{x_3^1, x_2^1\}$	$E_2^2 = \{x_2^2, x_3^2\}$	$E_3^2 = \{x_2^3, x_3^3\}$
$\mathcal{I}_{i,loc}$	$\{x_1^1, x_2^1, x_4^1\}$	$\{x_2^2, x_4^2\}$	$\{x_1^3, x_3^3\}$	$\{x_2^1, x_3^1\}$	$\{x_2^2, x_3^2\}$	$\{x_1^3, x_2^3, x_3^3\}$
$\mathcal{F}_{i,loc}$	$\{x_{01}^1, x_{02}^1\}$	$\{x_{01}^2, x_6^2\}$	$\{x_{01}^3, x_4^3\}$	$\{x_{01}^1, x_1^1, x_4^1, x_6^1\}$	$\{x_{01}^2, x_5^2\}$	$\{x_{01}^3, x_4^3\}$
$\mathcal{F}^{\sigma_{uc}}$	$\{x_3^1\}$	$\{x_2^2\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{F}_i^{\sigma}$	$\emptyset$	$\{x_3^2\}$	$\{x_2^3\}$	$\emptyset$	$\{x_3^2\}$	$\{x_2^3\}$

Table 1.

3.3.2 Global Forbidden and border Sets

Based on the sets $\mathcal{I}_{i,loc}(E_i), \mathcal{F}_{i,loc}(E_i)$ and $\mathcal{F}_i^{\sigma}(E_i)$ that can be locally computed, we now define the global weak forbidden set of states as well as the global border set. In a first step, we consider a product set $E = E_1 \times \cdots \times E_n$. We first introduce $\mathcal{I}_{loc}(.)$ as the set of global states that may lead by a sequence of local uncontrollable events to E. This set can be computed on G as follows:

$$\mathcal{I}_{loc}(E) = Pre^{G^*}_{\Sigma_{uc} \setminus \Sigma_s}(E) \tag{24}$$

In fact, from a computational point of view, it is not interesting to compute $\mathcal{I}_{loc}(E)$ using directly the operator $Pre^{G^*}_{\Sigma_{uc} \setminus \Sigma_s}$. Instead, it can be shown that $\mathcal{I}_{loc}(E)$ can be computed using the local weak forbidden sets $\mathcal{I}_{i,loc}(.)$.

Proposition 7 $\mathcal{I}_{loc}(E) = \mathcal{I}_{1,loc}(E_1) \times \cdots \times \mathcal{I}_{n,loc}(E_n)$

The proof of Proposition 7 can be mimicked from the one of Proposition 5, knowing that the sequences that we consider only contain uncontrollable events that are local to each component of the plant. Coming back to Example 6, $\mathcal{I}_{loc}(E^1) = \{x_1^1, x_2^1, x_4^1\} \times \{x_2^2, x_4^2\} \times \{x_1^3, x_3^3\}$.

$\mathcal{I}_{loc}(E)$ actually constitutes a first approximation of the global states that have to be forbidden by control. Indeed, one can easily note that $\mathcal{I}_{loc}(E) \subseteq \mathcal{I}(E)$, where $\mathcal{I}(E)$ corresponds to the weak forbidden set of

states that would be computed on the whole system G. Equality is not met (in general) since the states that are traversed by an uncontrollable trajectory leading to E, having at least one shared event are not taken into account.
Now, depending on the status of the events (shared or not shared, controllable or not controllable), there actually exist different ways to reach $\mathcal{I}_{loc}(E)$.

1. Either it is reached by triggering a local controllable event,
2. or it is reached by triggering a controllable shared event,
3. or it is reached by triggering an uncontrollable shared event.

For point (1), we denote by $\mathcal{F}_G^i(E)$ the set of states that can lead to $\mathcal{I}_{loc}(E)$ by triggering a local event that belongs to $\Sigma_i \setminus \Sigma_s$. For point (2) and (3), we denote by $\mathcal{F}_G^\sigma(E)$,with $\sigma \in \Sigma_s$, the set of states of G from which it is possible to reach $\mathcal{I}_{loc}(E)$ by triggering $\sigma \in \Sigma_s$.

According to these informal definitions, one can remark that $\mathcal{F}_G^i(E) \subseteq \mathcal{F}(E)$ and for $\sigma \in \Sigma_s \cap \Sigma_c$ $\mathcal{F}_G^\sigma(E) \subseteq \mathcal{F}(E)$[4], i.e. these global states will belong to the border of E. However, this is not the case for the global states that belong to $\mathcal{F}_G^\sigma(E)$, with $\sigma \in \Sigma_s \cap \Sigma_{uc}$, as they may lead into $\mathcal{I}_{loc}(E) \subseteq \mathcal{I}(E)$ by triggering an uncontrollable event.

We now show how this sets can be locally computed.
In a first step, we are interested in computing the states that lead to $\mathcal{I}_{loc}(E)$ by triggering only local events. As previously said, they will be denoted by $\mathcal{F}_G^i(E)$, whenever the concerned local events belongs to the alphabet of the component G_i. In fact, by definition, we have that $\mathcal{F}_G^i(E) = Pre^G_{\Sigma_i \setminus \Sigma_s}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$. This set can be efficiently reorganized, without building G, as follows:

Proposition 8 $\mathcal{F}_G^i(E) = \mathcal{F}_{i,loc}(E_i) \times (\Pi_{k \neq i} \mathcal{I}_{k,loc}(E_k))$

Proof : By definition, we have $\mathcal{F}_G^i(E) = Pre^G_{\Sigma_i \setminus \Sigma_s}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$. Hence

$$\begin{array}{rcll}
\mathcal{F}_G^i(E) & = & Pre^G_{\Sigma_i \setminus \Sigma_s}(\mathcal{I}_{1,loc}(E_1) \times \cdots \times \mathcal{I}_{n,loc}(E_n)) \setminus \mathcal{I}_{loc}(E) & \text{(because of Prop. 7)} \\
& = & \left(Pre^{G_i}_{\Sigma_i \setminus \Sigma_s}(\mathcal{I}_{i,loc}(E_i)) \times (\Pi_{k \neq i} \mathcal{I}_{k,loc}(E_k))\right) \setminus \mathcal{I}_{loc}(E) & \text{(because of Prop. 4)} \\
& = & \left(Pre^{G_i}_{\Sigma_i \setminus \Sigma_s}(\mathcal{I}_{i,loc}(E_i)) \setminus \mathcal{I}_{i,loc}(E_i) \times (\Pi_{k \neq i} \mathcal{I}_{k,loc}(E_k))\right) & \\
& = & \mathcal{F}_{i,loc}(E_i) \times (\Pi_{k \neq i} \mathcal{I}_{k,loc}(E_k)) & \text{(because of (22))}
\end{array}$$

◇

Next, for $\sigma \in \Sigma_s$, we introduce the set $\mathcal{F}_G^\sigma(E)$, which somehow represents the border set of E according to σ in G, i.e. the set of states from which it is possible to reach $\mathcal{I}_{loc}(E)$ by triggering the shared event σ. This set is formally defined by $\mathcal{F}_G^\sigma(E) = Pre^G_{\{\sigma\}}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$, which in turn can be rewritten as follows:

Proposition 9 $\mathcal{F}_G^\sigma(E) = \left[\left(\Pi_{i \in \mathbf{IN}(\sigma)} \mathcal{F}_i^\sigma(E_i)\right) \times \left(\Pi_{k \notin \mathbf{IN}(\sigma)} \mathcal{I}_{k,loc}(E_k)\right)\right] \setminus \mathcal{I}_{loc}(E)$

Proof: $\begin{array}{rcl}
\mathcal{F}_G^\sigma(E) & = & Pre^G_{\{\sigma\}}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) \\
& = & Pre^G_{\{\sigma\}}(\mathcal{I}_{1,loc}(E_1) \times \cdots \times \mathcal{I}_{n,loc}(E_n)) \setminus \mathcal{I}_{loc}(E) \\
& = & \left[\left(\Pi_{i \in \mathbf{IN}(\sigma)} Pre^{G_i}_{\{\sigma\}}(\mathcal{I}_{i,loc}(E_i))\right) \times \left(\Pi_{k \notin \mathbf{IN}(\sigma)} \mathcal{I}_{k,loc}(E_k)\right)\right] \setminus \mathcal{I}_{loc}(E) \\
& = & \left[\left(\Pi_{i \in \mathbf{IN}(\sigma)} \mathcal{F}_i^\sigma(E_i)\right) \times \left(\Pi_{k \notin \mathbf{IN}(\sigma)} \mathcal{I}_{k,loc}(E_k)\right)\right] \setminus \mathcal{I}_{loc}(E)
\end{array}$

[4] where $\mathcal{F}(E)$ is the border set of G w.r.t E.

In Example 6, $\mathcal{F}_G^{\sigma_{uc}}(E^1) = \{x_3^1\} \times \{x_2^2\} \times \{x_3^3, x_1^3\}$, and $\mathcal{F}_G^{\sigma}(E^1) = \{x_1^1, x_2^1, x_4^1\} \times \{x_3^2\} \times \{x_2^3\}$.
Intuitively, to evolve into $\mathcal{I}_{loc}(E)$, via the event $\sigma \in \Sigma_s$, it is sufficient for the components not sharing this event to be in a local weak forbidden state whereas the other components are in a state of the border according to σ (i.e. in $\mathcal{F}_i^{\sigma}(E_i)$). If σ is triggered, then all the components sharing this event will simultaneously evolve into a local weak forbidden state and overall G will also evolve into a weak forbidden global state, belonging to $\mathcal{I}_{loc}(E)$ (see e.g. Example 5 for the state $\langle x_1, x_1' \rangle$). Note that we make no distinction between the status of the event (controllable or not). This will be made further in the section.

Finally, we define $\mathcal{F}_G(E)$, which represents the set of states of G from which it is possible to reach $\mathcal{I}_{loc}(E)$, triggering an event $\sigma \in \Sigma_c$, i.e. $\mathcal{F}_G(E) = Pre_{\Sigma_c}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$. Using the set of states $\mathcal{F}_G^i(E)$ and $\mathcal{F}_G^{\sigma}(E)$ previously defined, we obtain:

Proposition 10 $\mathcal{F}_G(E) = [\bigcup_{1 \leq i \leq n} \mathcal{F}_G^i(E)] \bigcup [\bigcup_{\sigma \in \Sigma_s \cap \Sigma_c} \mathcal{F}_G^{\sigma}(E))]$

Lemma 2 $Pre_{\Sigma_{i,c} \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) = Pre_{\Sigma_i \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$

Proof : First we have that $Pre_{\Sigma_{i,c} \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) \subseteq Pre_{\Sigma_i \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$ (Proposition 1, *item (2)*). Now, let $x \in Pre_{\Sigma_i \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$, then $\exists \sigma \in \Sigma_i \setminus \Sigma_s$ s.t. $\delta(\sigma, x) \in \mathcal{I}_{loc}(E)$. Now σ is obviously controllable (otherwise x would have belong to $\mathcal{I}_{loc}(E)$), hence $x \in Pre_{\Sigma_{i,c} \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$.

Proof of Proposition 10 :

$$
\begin{aligned}
\mathcal{F}_G(E) &= Pre_{\Sigma_c}^G(\mathcal{I}(E)) \setminus \mathcal{I}(E) = Pre_{\Sigma_c}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) = [Pre_{\Sigma_c \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \cup Pre_{\Sigma_c \cap \Sigma_s}^G(\mathcal{I}_{loc}(E)) \\
&= [\textstyle\bigcup_{1 \leq i \leq n} Pre_{\Sigma_{i,c} \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)] \cup [\bigcup_{\sigma \in \Sigma_c \cap \Sigma_s} Pre_{\sigma}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)] \\
&= [\textstyle\bigcup_{1 \leq i \leq n} Pre_{\Sigma_i \setminus \Sigma_s}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)] \cup [\bigcup_{\sigma \in \Sigma_c \cap \Sigma_s} Pre_{\sigma}^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)] \text{ (according to Lemma 2)} \\
&= [\textstyle\bigcup_{1 \leq i \leq n} \mathcal{F}_G^i(E)] \bigcup [\bigcup_{\sigma \in \Sigma_s \cap \Sigma_c} \mathcal{F}_G^{\sigma}(E))] \cup \bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^{\sigma}(E)) \\
&= [\textstyle\bigcup_{1 \leq i \leq n} \mathcal{F}_G^i(E)] \bigcup [\bigcup_{\sigma \in \Sigma_s \cap \Sigma_c} \mathcal{F}_G^{\sigma}(E))]
\end{aligned}
$$

So far, we gave the definitions for a set of forbidden global states modeled as a product set. Next we extend these definition to the general case (i.e. a union of product sets).

Proposition 11 *Let $G = G_1 \parallel \cdots \parallel G_n$ be the plant to be controlled according to the set of forbidden global states $E = \bigcup_{1 \leq j \leq m} E^j$ where for all $1 \leq j \leq m$, E^j represents a product set. We then have,*

$$\mathcal{I}_{loc}(E) = \bigcup_{1 \leq j \leq m} \mathcal{I}_{loc}(E^j) \tag{25}$$

$$\forall \sigma \in \Sigma_s,\ \mathcal{F}_G^{\sigma}(E) = \bigcup_{1 \leq j \leq m} \mathcal{F}_G^{\sigma}(E^j) \setminus \mathcal{I}_{loc}(E),\ and \tag{26}$$

$$\mathcal{F}_G(E) = \bigcup_{1 \leq j \leq m} \mathcal{F}_G(E^j) \setminus \mathcal{I}_{loc}(E). \tag{27}$$

3.3.3 Supervisor Computation

Before describing the general method, we here give an intermediate result that makes the link with the classical notion of weak forbidden/border sets defined by (3), (4).

Proposition 12 *Let $G = G_1 \parallel \cdots \parallel G_n$ be a concurrent system and $E = E_1 \times \cdots \times E_n$ be a product set. Then, if $\forall \sigma \in \Sigma_s \cap \Sigma_{uc},\ \mathcal{F}_G^\sigma(E) = \emptyset$ then $\mathcal{I}(E) = \mathcal{I}_{loc}(E)$ and $\mathcal{F}_G(E) = \mathcal{F}(E)$, where $\mathcal{I}(E)$ and $\mathcal{F}(E)$ are the set of weak forbidden states and the border set of E in G (i.e. computed as in Definition 4).*

Proof : One can easily show that $\mathcal{I}_{loc}(E) \subseteq \mathcal{I}(E)$. Now, let $x \in \mathcal{I}(E) \setminus \mathcal{I}_{loc}(E)$ and $s \in \Sigma_{uc}^*$ be one of the uncontrollable sequences starting from x and leading to E (this sequence exists since $x \in \mathcal{I}(E)$). Then s can be decomposed into $s_1 \sigma s_2$ where $s_1 \in \Sigma_{uc}^*$, $\sigma \in \Sigma_s \cap \Sigma_{uc}$ and $s_2 \in (\Sigma_{uc} \setminus \Sigma_s)^*$ (i.e. σ is the last shared event of s). Let $x' = \delta(x, s_1\sigma)$. Then, $x' \in \mathcal{I}_{loc}(E)$ since E can be reached from x' triggering only local uncontrollable events. Moreover, we have that $\delta(x, s_1) \in \mathcal{F}_G^\sigma(E)$ as $\delta(x, s_1\sigma) = x' \in \mathcal{I}_{loc}(E)$, which contradicts the fact that $\mathcal{F}_G^\sigma(E) = \emptyset$. We can then conclude that $x \notin \mathcal{I}(E) \setminus \mathcal{I}_{loc}(E)$ and that $\mathcal{I}(E) = \mathcal{I}_{loc}(E)$.
Finally, by definition,

$$
\begin{aligned}
\mathcal{F}(E) &= Pre_\Sigma^G(\mathcal{I}(E)) \setminus \mathcal{I}(E) = Pre_\Sigma^G(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) \\
&= [Pre^G_{\Sigma_c \setminus \Sigma_s}(\mathcal{I}_{loc}(E)) \cup Pre^G_{\Sigma_c \cap \Sigma_s}(\mathcal{I}_{loc}(E)) \cup Pre^G_{\Sigma_{uc} \cap \Sigma_s}(\mathcal{I}_{loc}(E))] \setminus \mathcal{I}_{loc}(E) \\
&= [\textstyle\bigcup_{1 \le i \le n} Pre^G_{\Sigma_{i,c} \setminus \Sigma_s}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)] \cup [\bigcup_{\sigma \in \Sigma_c \cap \Sigma_s} Pre^G_\sigma(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)] \\
&\qquad \cup [\textstyle\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} Pre^G_\sigma(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)] \\
&= [\textstyle\bigcup_{1 \le i \le n} \mathcal{F}_G^i(E)] \bigcup [\bigcup_{\sigma \in \Sigma_s \cap \Sigma_c} \mathcal{F}_G^\sigma(E))] \cup \bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^\sigma(E)) \\
&= [\textstyle\bigcup_{1 \le i \le n} \mathcal{F}_G^i(E)] \bigcup [\bigcup_{\sigma \in \Sigma_s \cap \Sigma_c} \mathcal{F}_G^\sigma(E))] = \mathcal{F}_G(E)
\end{aligned}
$$

◇

Based on the previous proposition, we can easily derived a supervisor ensuring the avoidance of the product set E in G.

Proposition 13 *If $\forall \sigma \in \Sigma_s \cap \Sigma_{uc},\ \mathcal{F}_G^\sigma(E) = \emptyset$ then the supervisor $\mathcal{S}_E = (S_E, \mathcal{X}_{oE})$, such that*

$$
\begin{aligned}
\forall x \in \mathcal{X},\ S_E(x) &= \begin{cases} \{\sigma \in \Sigma_c |\ \delta(\sigma, x) \in \mathcal{I}_{loc}(E)\} \text{ if } x \in \mathcal{F}_G(E) \\ \emptyset \text{ Otherwise} \end{cases} \\
\mathcal{X}_{oE} &= \mathcal{X}_o \setminus \mathcal{I}_{loc}(E)
\end{aligned}
$$

ensures the avoidance of E in G and is maximal. ◇

Proposition 2 and 12 immediately give the proof. As all the computations are performed locally, the complexity of this synthesis phase is in $\mathcal{O}(n.k.N)$ (using notations of Section 3.2). The fact that we need to compute specialized borders w.r.t. some shared events does not increase the complexity, as all these computations can be done on the fly when the graph is being explored. Moreover, from a computational point of view, $\mathcal{I}_{loc}$ and $\mathcal{F}_G$ does not need to be computed. It is sufficient and more convenient to store the different local sets $\mathcal{I}_{i,loc}$, $\mathcal{F}_i^\sigma$, and $\mathcal{I}_{i,loc}$ in memory and to perform some tests on-line in order to know whether a configuration belongs to one of these sets (as done in Section 3.2). This increases the on-line complexity, but avoids to store sets of states that may be too large to be (efficiently) represented.
Next we show how to extent this result to any set of global states E.

Theorem 2 *Let $G = G_1 \parallel \cdots \parallel G_n$ be the plant to be controlled and a set $E = \bigcup_{1 \le j \le m} E^j$ where $\forall 1 \le j \le m,\ E^j = E_1^j \times \cdots \times E_n^j$ and $E_i^j \subseteq \mathcal{X}_i$ for $1 \le i \le n$.*

Assume $\forall \sigma \in \Sigma_s \cap \Sigma_{uc},\ \mathcal{F}_G^\sigma(E) = \emptyset$, then let $\mathcal{S}_{E^j} = (S_{E^j}, \mathcal{X}_{o_{E^j}})$ be the maximal supervisors computed w.r.t. G and E^j (Proposition 13), then $\mathcal{S}_E = (S, \mathcal{X}_{o_E})$, where $\forall x = \langle x_1, \ldots, x_n \rangle \in \mathcal{X}$,

$$
\begin{cases} S(x) = S_{E^1}(x) \cup \cdots \cup S_{E^m}(x) \\ \mathcal{X}_{o_E} = \mathcal{X}_{o_{E^1}} \cap \cdots \cap \mathcal{X}_{o_{E^m}} \end{cases} \tag{28}
$$

ensures the avoidance E in G and is maximal.

The proof is immediate and is due to Proposition 13 and Proposition 3. Moreover, the controller synthesis phase is in $\mathcal{O}(m.n.k.N)$.

General Case.
Now, if $\exists \sigma \in \Sigma_s \cap \Sigma_{uc}$ such that $\mathcal{F}_G^\sigma(E) \neq \emptyset$, then $\mathcal{I}(E) \neq \mathcal{I}_{loc}(E)$. However, based on the previous results, we can remark that the weak forbidden global states that are not taken into account are the ones that lead into E via an uncontrollable sequence having shared events. In order to capture these states, we introduce the following function Φ

Definition 6 *Let $E \subseteq \mathcal{X}$, then the function $\Phi : 2^{\mathcal{X}} \rightarrow 2^{\mathcal{X}}$ is defined by*

$$\Phi(E) = E \bigcup \bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^\sigma(E) \tag{29}$$

The set $\Phi(E)$ contains E and the set of states that may lead to E triggering a sequence of uncontrollable events such that its first event is a shared uncontrollable event. We now consider the sequence $(\Phi^n(E))_{n \geq o}$ (with $\Phi^o(E) = E$). For all $n \geq 1$, elements of $\Phi^n(E)$ can lead to E triggering a sequence of uncontrollable events which contains less than n elements of Σ_s. We denote by $\Phi^*(E)$ the limit of the sequence $(\Phi^n(E))_{n \geq o}$ (see Figure 4). This limit always exists and is reached in a finite number of iterations, since the number of states of the system is finite.

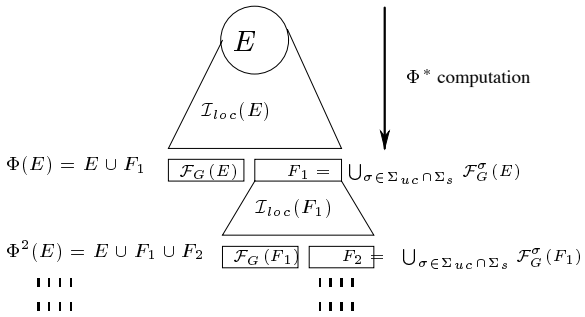


Figure 4: Φ^* Intuition from the point of view of G

Based on Φ and Φ^*, we can show the two following lemmas

Lemma 3 $\forall \sigma \in \Sigma_s \cap \Sigma_{uc},\ \mathcal{F}_G^\sigma(\Phi^*(E)) = \emptyset$

Proof : According to Definition 6, $\Phi(\Phi^*(E)) = \Phi^*(E) \cup \bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^\sigma(\Phi^*(E))$. Moreover, from definition of Φ^*, $\Phi(\Phi^*(E)) = \Phi^*(E)$. Hence, $\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^\sigma(\Phi^*(E)) = \emptyset$. $\diamond$

Lemma 4 $\mathcal{I}(\Phi^*(E)) = \mathcal{I}(E)$

Proof : First we show that $\Phi^*(E) \subseteq \mathcal{I}(E)$ The proof proceed by induction over n. We want to prove that $\forall n \geq 0$, $\Phi^n(E) \subseteq \mathcal{I}(E)$. The result is clear for $n = 0$ (in this case, $\Phi^0(E) = E$). Now let us consider $n \geq 0$ and let us suppose that $\Phi^n(E) \subseteq \mathcal{I}(E)$. This easily entails that $\Phi^{n+1}(E) \subseteq \Phi(\mathcal{I}(E))$. Moreover, according to (3) and Definition 6, we have $\Phi(\mathcal{I}(E)) = \mathcal{I}(E)$. Therefore, we have $\Phi^{n+1}(E) \subseteq \mathcal{I}(E)$, which, once the fix point reached, gives $\Phi^*(E) \subseteq \mathcal{I}(E)$. This easily entails that $\mathcal{I}(\Phi^*(E)) \subseteq \mathcal{I}(E)$. Conversely, since $\mathcal{X}' \subseteq \mathcal{X}'' \Longrightarrow \mathcal{I}(\mathcal{X}') \subseteq \mathcal{I}(\mathcal{X}'')$, and $E \subseteq \Phi^*(E)$, we have $\mathcal{I}(E) \subseteq \mathcal{I}(\Phi^*(E))$. Hence $\mathcal{I}(E) = \mathcal{I}(\Phi^*(E))$. $\diamond$

From Property 12, generalized to a union of product set we can deduce from the previous lemma that $\mathcal{I}(E) = \mathcal{I}_{loc}(\Phi^*(E))$, from which we conclude that:

Theorem 3 *Given $G = G_1 \parallel \cdots \parallel G_n$ and E a set of forbidden configurations. Let $\Phi^*(E) = \cup_{1 \leq j \leq m} E^j$ be the decomposition in terms of product sets of $\Phi^*(E)$, then the supervisor $\mathcal{S}_E = (S_E, \mathcal{X}_{oE})$, such that $\forall x \in \mathcal{X}$*

$$\begin{array}{rcl} S_E(x) & = & S_{E^1}(x) \cup \cdots \cup S_{E^m} \\ \mathcal{X}_{oE} & = & \mathcal{X}_o \setminus \mathcal{I}_{loc}(\Phi^*(E)) \end{array}$$

ensures the avoidance of E in G and is maximal.

Proof : According to lemma 3, $\forall \sigma \in \Sigma_s \cap \Sigma_{uc}$, $\mathcal{F}_G^{\sigma}(\Phi^*(E)) = \emptyset$. We then obtain the result by applying Theorem 2. ◇

The result of Theorem 3 gives us a method to solve SCP without explicitly building G. We made the necessary effort to perform the computation locally on each component of G, hence reducing the global complexity of the algorithm. However, it is worthwhile noting that the complexity of the supervisor computation heavily depends on the number of product sets that is used to express $\Phi^*(E)$. Indeed, at each iteration, new product sets need to be forbidden (i.e. the one that belongs to $\mathcal{F}_G^{\sigma}(\Phi^i(E))$ for $\sigma \in \Sigma_{uc} \cap \Sigma_s$. In particular, this number obviously depends on the cardinal of the shared uncontrollable events. Hence, this method is more suitable and effective for loosely synchronous systems, for which most of the behavior of the components is local and is synchronized occasionally with other components).

Example 7 *Coming back top the previous example, one can show that* $\Phi(E) = E^1 \cup E^2 \cup E^3$ *where* $E^3 = \{x_3^1\} \times \{x_2^2\} \times \{x_3^3, x_1^3\}$. *The computation of the weak forbidden and border sets for* $E1$ *and* E^2 *had already been done. So we only need to perform the computations for* E^3. *The results are summarized in the next table.*

	E_1^3	E_2^3	E_3^3
$\mathcal{I}_{i,loc}$	$\{x_3^1\}$	$\{x_2^2\}$	$\{x_1^3, x_3^3\}$
$\mathcal{F}_{i,loc}$	$\{x_1^1, x_6^1\}$	$\{x_{01}^2\}$	$\{x_{01}^3, x_4^3\}$
$\mathcal{F}_i^{\sigma_{uc}}$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{F}_i^{\sigma}$	$\emptyset$	$\{x_3^2\}$	$\{x_2^3\}$

At this stage, it is easy to see that $\Phi(E) = \Phi^*(E)$. *Hence,* $\mathcal{S}$ *can be deduced from the Table 1 and 2 following Theorem 3.* ◇

4 Conclusion

In this paper, we investigated the State Avoidance Control Problem for loosely synchronous systems. The methodology is based upon a decomposition of the set of forbidden configurations E according to the structure of the system. Based on this decomposition, it is then possible to locally compute the set of weak locally forbidden states and some local border sets on each component leading to a global supervisor ensuring the avoidance of E. We provide different methods depending whether the components share common events or not. Note that at this point, Theorem 2 and 3 do not require that $\Sigma_s \subseteq \Sigma_c$. Hence compared to [2], we have less restrictive assumptions. However, we focused so far on the State avoidance Control Problem, whereas in [2], their methodology based on a language formalism allows to handle more intricate control objectives (this aspect is currently under investigation). We are also looking for an algorithm that will force the plant to be 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 [15, 9].

A Optimal control of Product Systems

The aim of the optimal control (see e.g. [20]) is to generate a controller which constrains the system to a desired behavior according to quantitative and qualitative aspects. This is performed by the addition of quantitative measures in the form of cost functions to capture the fact that some legal behaviors are better than others. In the basic setup of supervisory control theory (see [18, 5]), optimality is with respect to set inclusion and thus all legal behaviors are equally good (zero cost) and illegal behaviors are equally bad (infinite cost). The work in [20] enriches this setup by the addition of quantitative measures in the form of occurrence and control cost functions, to capture the fact that some legal behaviors are better than others. The problem is then to synthesize a controller that is not only legal, but also "good" in the sense of given quantitative measures[5].

[5] Some other studies appear in [12, 16, 21].

In this section, after a brief presentation of the theory developed in [20], we extend these results to the control of plant modeled as a product system.

A.1 The Optimal Control Problem.

We here recall some results for the optimal control of FSM (with a *unique initial and final state*) that can be found in [20]. Our aim here is not to describe in detail all the theory, but to present the principal notations and results that we will use in the sequel. In order to take into account the numerical aspect of the optimal control problem, two cost values are associated to each event of Σ. To this effect, we introduce an occurrence cost function $c_e : \Sigma \rightarrow \boldsymbol{R}^+$ and a control cost function $c_c : \Sigma \rightarrow \{0, \infty\}$. The control cost function is used to encode the status of the events. The control cost function is infinity for events in Σ_{uc}. The cost functions are then used to introduce a cost on the trajectories of a submachine H of G (note that in [20], the control cost function is not trivial). We introduce $\Sigma_d^G(H, x)$ as the set of disabled events at state x for the system to remain in submachine H of G as well as $\mathcal{X}(s, x)$ the set of states crossed on the way when traversing s.

Definition 7 *Let H be a submachine of G and $\mathcal{L}_m(H)$ be the marked language generated by H, then*

- *For any state $x \in \mathcal{X}_H$ and string $s = \sigma_1^s \sigma_2^s \dots \sigma_{\|s\|}^s$, such that $\delta_H(s, x)$ exists, the cost of s is given by:*

$$c^g(x, H, s) = \sum_{j=1}^{\|s\|} c_e(\sigma_j^s) \; + \sum_{x' \in \mathcal{X}(s,x)} \sum_{\sigma \in \Sigma_d^G(H,x')} c_c(\sigma) \tag{30}$$

- *The objective function denoted by $c(.)$ is given by: $c(H) = \sup_{s \in \mathcal{L}_m(H)} c^g(x_{0A}, H, s)$.*

Basically, the cost of a trajectory is the sum of the occurrence costs of the events composing it. If an uncontrollable event is disabled, the cost of a trajectory becomes infinite because the second term of (30) becomes infinity. Finally, $c(H)$ represents the worst case behavior possible in submachine H.
We now define the optimization problem.

Definition 8 *For all $x \in \mathcal{X}$, $H \subseteq G(x)$ is an optimal submachine if $c(H) = \min_{H' \subseteq G(x)} c(H') < \infty$.*

In particular, an optimal solution $H \subseteq G(x_0)$ is an optimal submachine of the plant G and represents a solution of the optimal control problem, which in general has more than one solution. For such a submachine H, $c(H)$ represents the optimal cost (in fact, the worst inevitable cost) necessary to reach x_f from x_0. It means that a submachine with a lower cost could not ensure the accessibility of x_f from x_0. The following lemma (lemma (2.15) in [19]) is stated to note that optimal solutions lie within the class of controllable submachines.

Lemma 5 *Let $H \subseteq G$. If $c(H) < \infty$ then H is controllable.* ◇

The next theorem ([20]) gives necessary and sufficient conditions for the existence of optimal submachines.

Theorem 4 *An optimal submachine of G exists if and only if there exists a submachine H of G such that H is trim, controllable with no cycles.* ◇

Intuitively, in this theorem, the trim assumption ensures that the final state can be still reached. the controllability assumption ensures that the positive cost cycles can be broken using controllable events alone. Finally the acyclic assumption together with the acyclic assumption ensure that the cost is finite. We now introduce the notion of DP-Optimal submachines. This kind of submachine will be used intensively for algorithm purposes.

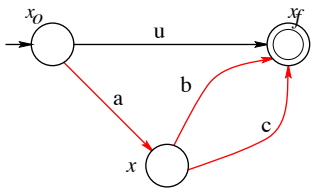
Definition 9 *A submachine $H \subseteq G(x)$ is DP-Optimal if it is optimal and for all $x' \in \mathcal{X}_H$, $H(x')$ is an optimal submachine of $G(x')$.*

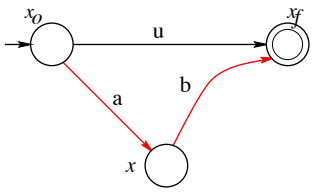
If a particular DP-Optimal FSM includes all other DP-Optimal FSMs as submachines, then we call it the *maximal DP-Optimal submachine*. Its existence is given by the following theorem ([20]).

Theorem 5 *If an optimal submachine of G exists, then the unique maximal DP-Optimal submachine of G w.r.t. the final state x_f also exists.* ◇

In [20], the authors provide an algorithm (DP-OPT), that, given an FSM G with a unique final state, constructs the maximal DP-Optimal submachine, with a worst-case complexity $\mathcal{O}(|X|^2|\Sigma|log(|\Sigma|) + |X|^3|\Sigma|)$ (Theorem 6.10 of [20]).

Example 8 *We here give a simple example that allows to grasp the notion of DP-Optimality versus Optimality. We have already seen that optimality actually exists when the worst-case cost from the initial state x_0 is to x_f is finite once minimized. DP-Optimality is obtained when the terminal path from any state of a submachine to the final state x_f is optimal in the previous sense. This is illustrated by the following example. We can observe*





Event	c_e	Remarks
a,b	1	Controllable
c	3	Controllable
u	5	Uncontrollable

Figure 5: A simple example giving the difference between Optimality and DP-Optimality

that, in both submachines, the cost for going from x_0 to x_f is optimal since there exist no other controllable path with a lower cost. However, in the first submachine, the worst-case cost to go from state x to state x_f is 3 (through event c); whereas the worst-case cost in the second submachine is optimized to be reduced to 1. Consequently, we can say that the second submachine is DP-Optimal (thereby optimal), but the first submachine is not DP-Optimal, but is optimal. ◇

A.2 The optimal control of product systems

We here extend the theory of [20] to a plant G modeled as a product system $G_12 \parallel \cdots \parallel G_n$, with $\Sigma_i \cap \Sigma_j = \emptyset$, for $i \neq j$. We restrict ourselves to FSMs G_i with a single final state.
Given two FSMs G_1 and G_2, the purpose of this section is to show that whenever there exist two (DP-)optimal submachines H_1 and H_2 of G_1 and G_2, then $H_1 \parallel H_2$ is also (DP-)optimal w.r.t. $G_1 \parallel G_2$.
In the first part of this section, we will consider FSMs with only one initial state as carried out in [20]. First, we need to introduce the following technical lemmas:

Lemma 6 [8] *Let G_1' and G_2' be submachines of G_1 and G_2, then $c(G_1' \parallel G_2') = c(G_1') + c(G_2')$.* ◇

Note that this lemma is true only because we consider the special case, where the control cost function is trivial (i.e. either equal to 0 or ∞) and because G_1 and G_2 are asynchronous FSMs. this lemma also entails that it is sufficient to consider elements of $\mathcal{L}(G_1 \parallel G_2)$ of the form $s_1 s_2$ with $s_1 \in \Sigma_1^*$ and $s_2 \in \Sigma_2^*$ when we are interested in the cost of $G_1 \parallel G_2$.

Lemma 7 [8] *Let $A \subseteq G_1 \parallel G_2$. Then, there exist $G_1' \subseteq G_1$ and $G_2' \subseteq G_2$, s.t. $c(G_1' \parallel G_2') = c(A)$.* ◇

Using the previous results, we are now able to prove the following property:

Proposition 14 *If H_1 and H_2 are optimal submachines of G_1 and G_2, then $H_1 \parallel H_2$ is an optimal submachine of $G_1 \parallel G_2$. The result is still valid when dealing with DP-Optimality.*

Proof : Let A be an optimal submachine of $G_1 \parallel G_2$. According to Lemma 7, there exist $G'_1 \subseteq G_1$ and $G'_2 \subseteq G_2$, s.t. $c(A) = c(G'_1 \parallel G'_2)$. Next, it is easy to show that the worst case cost of $H_1 \parallel H_2$ is minimal among the submachines of G of the form $G'_1 \parallel G'_2$ (otherwise, either G'_1 or G'_2 would have a worst case cost lower than the one of H_1 or H_2). Hence we have $c(A) = c(G'_1 \parallel G'_2) \geq c(H_1 \parallel H_2)$, which proves that $H_1 \parallel H_2$ is optimal w.r.t. $G_1 \parallel G_2$. The DP-Optimality can be easily deduced from the previous point. $\diamond$

So far, we were interested in composing FSMs having only one initial state. Knowing that the FSMs we consider have several initial states, we need to extend these results to FSMs having this property.

Definition 10 *Let G be a FSM, and H a submachine of G such that $\mathcal{X}_{H_o} \subseteq \mathcal{X}_{G_o}$ and $\mathcal{X}_{H_o} \neq \emptyset$. H is (DP-)optimal if $\forall x_0 \in \mathcal{X}_{H_o}$, $H(x_0)$ is (DP-)optimal w.r.t. $G(x_0)$, and $\forall x_0 \in \mathcal{X}_{G_o} \setminus \mathcal{X}_{H_o}$, $G(x_0)$ has no optimal submachine.* $\bullet$

With the preceding notations, consider the maximal DP-Optimal submachine $H(x_o)$ of $G(x_o)$. We denote by $\mathcal{X}'_o$ the set of initial states for which such a submachine exists. It is then easy to show that $H = \oplus_{x_o \in \mathcal{X}'_o}(H(x_o))$[6] is DP-optimal w.r.t. G. Based on this result, we can state the following proposition:

Proposition 15 *Let H_1 and H_2 be DP-optimal submachines of G_1 and G_2, then $H_1 \parallel H_2$ is a DP-optimal submachine of $G_1 \parallel G_2$.* $\diamond$

Proof : We just give here the sketch of the proof. For all $x_{o_i} \in \mathcal{X}'_{o_1}$ and $x_{o_j} \in \mathcal{X}'_{o_2}$, using Proposition 14, we know that $H_{ij} = H_1(x_{o_i}) \parallel H_2(x_{o_j})$ is DP-optimal w.r.t. $G_1(x_{o_i}) \parallel G_2(x_{o_j})$. Now, based on the fact that the merge of two DP-Optimal submachines is still a DP-Optimal submachine, we can deduce that $\oplus_{ij} H_1(x_{o_i}) \parallel H_2(x_{o_j}) = H_1 \parallel H_2$ is a DP-Optimal of $G_1 \parallel G_2$. Consider now a state $x_o = \langle x_{o_1}, x_{o_2} \rangle \notin \mathcal{X}'_{o_1} \times \mathcal{X}'_{o_2}$. Assume $x_{o_1} \notin \mathcal{X}'_{o_1}$. It entails that there exists no DP-Optimal submachine of $G_1(x_{o_1})$, and that $\forall H(x_{o_1}) \subseteq G_1(x_{o_1})$, $c(H(x_{o_1})) = \infty$. Finally, using Lemma 6 and 7, we can show that $\forall H'(x_{o_j}) \subseteq G_2(x_{o_j})$, where $x_{o_j} \in \mathcal{X}_{o_2}$, $c(H(x_{o_1}) \parallel H'(x_{o_j})) = \infty$, which means that there is no (DP-)optimal submachine of $G_1(x_{o_1}) \parallel G_2(x_{o_j}) = G_1 \parallel G_2(\langle x_{o_1}, x_{o_j} \rangle)$. $\diamond$

References

[1] A. Abdelwahed. *Interacting Discrete event systems: modeling, verification an supervisory control*. PhD thesis, University of Toronto, 2002.

[2] 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.

[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] M.H. deQueiroz and J.E.R. Cury. Modular supervisory control of large scale discrete-event systems. In *Discrete Event Systems: Analysis and Control. Proc. WODES'00*, pages 103–110. Kluwer Academic, 2000.

[6] $\oplus$ denotes the merge of two FSMs (see [20]).

[7] M.H. deQueiroz and J.E.R. Cury. Synthesis and implementation of local modular supervisory control for a manufacturing cell. In *Proceedings of the 6th International Workshop on Discrete Event Systems*, pages 377–382, October 2002.

[8] B. Gaudin. Optimal control of hierarchical finite state machines. Master's thesis, University of rennes I, 2001. (In French).

[9] B. Gaudin and H. Marchand. Modular supervisory control of asynchronous and hierarchical finite state machines. In *European Control Conference, ECC 2003*, Cambridge, UK, September 2003.

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

[11] B. H. Krogh. Supervisory control of Petri nets. In *Belgian-French-Netherlands' Summer School on Discrete Event Systems*, June 1993.

[12] R. Kumar and V. Garg. Optimal supervisory control of discrete event dynamical systems. *SIAM Journal of Control and Optimization*, 33(2):419–439, March 1995.

[13] F. Lin and W.M. Wonham. Decentralized control and coordination of discrete-event systems with partial observation. *IEEE Transactions of Automatic Control*, 35(12):1330–1337, December 1990.

[14] 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.

[15] H. Marchand and B. Gaudin. Supervisory control problems of hierarchical finite state machines. In *41th IEEE Conference on Decision and Control*, Las Vegas, USA, December 2002.

[16] K. M. Passino and P. J. Antsaklis. On the optimal control of discrete event systems. In *Proc. of 28th Conf. Decision and Control*, pages 2713–2718, Tampa, Floride, December 1989.

[17] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM J. Control Optim.*, 25(5):1202–1218, September 1987.

[18] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems*, 77(1):81–98, 1989.

[19] R. Sengupta and S. Lafortune. A deterministic optimal control theory for discrete event systems: Computational results. Technical Report n° CGR-93-16, Control Group, College of Engineering, University of Michigan, USA, December 1993.

[20] R. Sengupta and S. Lafortune. An optimal control theory for discrete event systems. *SIAM Journal on Control and Optimization*, 36(2), March 1998.

[21] E. Tronci. Optimal state supervisory control. In *Proc. of* 35^th *IEEE conf. on Decision and Control*, Kobe, Japon, December 1996.

[22] T. Ushio. On controllable predicates and languages in discrete-event systems. In *Proc. of the* 28^{th} *Conference on Decision and Control*, pages 123–124, Tampa, Floride, December 1989.

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

[24] W. M. Wonham. Notes on control of discrete-event systems. Technical Report ECE 1636F/1637S, Department of Electrical and Computer EngineeringUnivertsity of Toronto, 2002.

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

[26] 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.

[27] Z.H. Zhang and W.M. Wonham. Stct: An efficient algorithm for supervisory control design. In *Symposium on Supervisory Control of Discrete Event Systems (SCODES2001)*, Paris (France), July 2001.

Contents