EFFICIENT COMPUTATION OF SUPERVISORS FOR LOOSELY SYNCHRONOUS DISCRETE EVENT SYSTEMS: A STATE-BASED APPROACH

Benoit Gaudin * **Herve Marchand** *

* *Irisa, Campus universitaire de Beaulieu, Rennes.*
First.Last@irisa.fr

Abstract: In this paper, we are interested in the control of a particular class of Concurrent Discrete Event Systems defined by a collection of components that interact with each other. We here consider the state avoidance control problem. We provide algorithms that, based on a particular decomposition of the set of forbidden states, locally solve the control problem (i.e. on each component without computing the whole system) and produce a global supervisor, that can be efficiently evaluated on the fly. *Copyright© 2005 IFAC.*

Keywords: Discrete Event Systems, Supervision and control, Concurrent Systems, State Avoidance Control Problem.

1. INTRODUCTION

In this paper, we are interested in the *Supervisory Control Problem* for Concurrent Discrete Event Systems defined by a collection of components that interact with each other. We are concerned with systems where the construction of the entire system is assumed not to be feasible (due to the state space explosion resulting from the composition), making the use of classical supervisory control methodologies impractical (See e.g. [Wonham (2003)] or [Cassandras and Lafortune (1999)]).

Several approaches have been considered, in the literature, to take into account the structure of concurrent systems [deQueiroz and Cury (2000); Akesson et al. (2002); Willner and Heymann (1991); Rohloff and Lafortune (2003)]. In most of the above works, the authors adopt a language-based approach and their methodology is characterized by the fact that the specification (i.e. the expected behavior) can be decomposed according to the structure of the plant. Under this hypothesis, they provide necessary and sufficient conditions under which it is possible to compute local modular supervisors acting upon each component and to operate the individually controlled plant concurrently in such a way that the behavior of the controlled plant corresponds to the supremal one.

However, it may happen that the specification that has to be ensured is more related to the notion of states rather than to the notion of trajectories of the system (the mutual exclusion for example). For this class of problem, one of the main issue 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 (See e.g. [Wonham (2003)]). Note that if one wants to used a language-based approach(as in e.g. Gaudin and Marchand (2004a)]) to encode this problem, then the obtained specification

- does not fit with the structure of the system (i.e. is, in general, not separable), and
- may be of the size of the global system itself,

which renders the use of the above works useless or at least intractable. This leads us to develop techniques totally devoted to the state avoidance control problem. Following the methodology described in [Minhas (2002)] and [Vahidi et al. (2004)], we decompose the computation in two phases (an off-line and an on-line computation): Based on a decomposition of the forbidden set of states in terms of set products, we provide a methodology that locally computes on each component of G the set of *bad states* (these are the states that may lead to the forbidden states via an uncontrollable trajectory). This is performed without computing the whole system. t this point, the supervisor is evaluated on the fly w.r.t. the bad states and thus requires an on-line evaluation in order to determine the set of events that has to be disabled by control (these are the events that may lead to a bad state). It is performed in such a way that the global partial transition function does not need to be built. Moreover, we make the necessary effort to obtain a good complexity during the on-line computation and off-line evaluation. For the latter, this is basically due to the fact that we give a structure to the set of bad states, similar to the one of the plant, so that the realization of the supervisor on the fly becomes easier.

2. PRELIMINARIES

We consider in this study a system represented as a Finite State Machine (FSM). An FSM is a 4-tuple

(Σ, Q, q_0, δ) where Σ is the set of events that can occur, Q is the set of states , q_0 is the initial state and $\delta : \Sigma \times Q \longrightarrow Q$ is the partial transition function. For $q \in Q$, $\delta(q)$ denotes the active event set of q. Similarly, $\delta^{-1}(q)$ denotes the set of events that lead to q. We also define the operator Pre_A^G for all $E \subseteq Q$ by

$$Pre_A^G(E) = E \cup \{q \in Q | \ \exists \sigma \in A, \ \delta(\sigma, q) \in E\} \quad (1)$$

as well as the operator

$$\mathit{CoReach}_A^G(E) = \bigcup_{n \geq 0} Pre_A^{G(n)}(E) \quad (2)$$

$\mathit{CoReach}_A^G(E)$ represents the set of states from which it is possible to reach a state of E by only triggering events of A. Note that $E \subseteq \mathit{CoReach}_A^G(E)$. We now recall the classical definition of the synchronous product of two FSM. This operation will be intensively used in the sequel to combine the different components involved in the specification of the plant we want to control.

Définition 1. Let $G_i = (\Sigma_i, Q_i, q_{0_i}, \delta_i)$, $i = 1, 2$ be FSM s.t. $\Sigma_s = \Sigma_1 \cap \Sigma_2$. The synchronous product *$G_1 \parallel G_2$ of G_1 and G_2 is the FSM $G = (\Sigma, Q, q_0, \delta)$ where $\Sigma = \Sigma_1 \cup \Sigma_2$, $Q = Q_1 \times Q_2$, $q_0 = \langle q_{0_1}, q_{0_2} \rangle$, and δ is defined by: for all $q = \langle q_1, q_2 \rangle \in Q$ and $\sigma \in \Sigma$*

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

The State Avoidance Control Problem. Let G be a plant modeled as an FSM (Σ, Q, q_0, δ). In order to control this FSM, we classically partition the alphabet into controllable events Σ_c and uncontrollable events Σ_{uc}. Given this partition, a supervisor $\mathcal{S}$ is given by a function $S : Q \rightarrow 2^{\Sigma_c}$, delivering the set of actions that are disabled in state q of G by control. Write $\mathcal{S}/G$ for the closed-loop system, consisting of the initial plant G controlled by the supervisor $\mathcal{S}$. In the sequel, we are interested in solving the *State Avoidance Control Problem* (*SACP*), where the control objective consists in states that have to be avoided by control:

SACP: *given G and E a set of states, the problem is to build a supervisor $\mathcal{S}_E$ such that (1) the traversed states do not belong to E and (2) $\mathcal{S}_E/G$ is the most permissive solution (according to the inclusion of languages).*

Remark 1. In the literature, this state avoidance control problem is often expressed using predicates over the states of the plant [Wonham (2003); Ushio (1989)]. The control problem is then to force the system to remain outside the states that satisfy the predicates. $\diamond$

In order to solve this problem, we first introduce the *weak forbidden set* $\mathcal{I}$:

$$\mathcal{I}(E) = CoReach_{\Sigma_{uc}}^G(E) \quad (3)$$

$\mathcal{I}(E)$ corresponds to the set of states from which it is possible to evolve into E by a trace of uncontrollable events. Note that this operator is monotonic and distributes over union.

Proposition 1. Given an FSM G and $E \subseteq Q$, a set of states E. If $q_o \notin \mathcal{I}(E)$, then the supervisor $\mathcal{S}_E$ of G, such that $\forall q \in Q$

$$\mathcal{S}_E(q) = \{\sigma \in \Sigma_c | \ \delta(\sigma, q)! \wedge \delta(\sigma, q) \in \mathcal{I}(E)\} \quad (4)$$

is the most permissive supervisor ensuring the avoidance of E in G (otherwise there is no supervisor ensuring the property). $\diamond$

The proof (with a predicate approach) can be found in [Wonham (2003)].

3. CONTROL OF CONCURRENT SYSTEMS

3.1 Concurrent System description

Let us now consider a plant G modeled as a collection of FSM $G_i = \langle \Sigma_i, Q_i, q_{oi}, \delta_i \rangle$. The global system is given by $G = G_1 \parallel \cdots \parallel G_n$. The resulting FSM will be noted $\langle \Sigma, Q, q_o, \delta \rangle$ and the states of G will be denoted by $q = \langle q_1, \ldots, q_n \rangle$.

Σ_s represents the set of shared events of G, i.e $\Sigma_s = \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j)$. Now, given the set of FSMs G_i modeling G, IN(.) is a function, which for each $\sigma \in \Sigma$ gives the set of indexes $i \in \{1, \ldots, n\}$ such that $\sigma \in \Sigma_i$.

Event Status. The alphabet of G_i is partitioned into the controllable event set $\Sigma_{i,c}$ and the uncontrollable event set $\Sigma_{i,uc}$, i.e. $\Sigma_i = \Sigma_{i,uc} \cup \Sigma_{i,c}$. The alphabet of the global plant G is given by:

$$\Sigma = \bigcup_i \Sigma_i, \ \Sigma_c = \bigcup_i \Sigma_{i,c}, \text{ and } \Sigma_{uc} = \Sigma \setminus \Sigma_c. \quad (5)$$

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

$$\forall i, j, \ \Sigma_{i,uc} \cap \Sigma_{j,c} = \emptyset \quad (6)$$

which simply means that the components that share an event agree on the control status of this event. Under this hypothesis, we have that $\Sigma_{uc} = \cup_i \Sigma_{i,uc}$. Note that as we will consider a monolithic supervisor acting upon the whole system, this hypothesis seems to be relevant.

3.2 Control Problem formulation

In the remainder of this section, our aim is to solve the SACP for a set of forbidden states E of a concurrent system. In a first step, we assume that this set is decomposed according to the structure of G. In fact, it can be shown that *any set of states E can be*

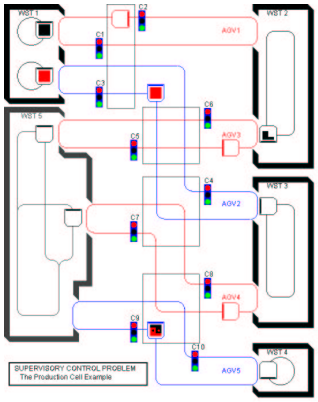
represented as a union of Cartesian products of sets, i.e.

$$E = \bigcup_{1 \leq j \leq m} E^j, \text{ where} \tag{7}$$

$$\forall 1 \leq j \leq m, \;\; E^j = E^j_1 \times \cdots \times E^j_n, \text{ and} \quad \forall 1 \leq i \leq n, \; E^j_i \subseteq Q_i \tag{8}$$

Given the concurrent structure of the system, this decomposition of sets in terms of product sets happens to be very natural and will be the basis for the expression of states that will have to be forbidden by control.

Example 1. To illustrate this aspect, we consider the classical example of the flexible manufacturing cell control [Krogh (1993)].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 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

$$\begin{aligned} Zone_1 &= E^1_1 \times E^2_1 \times Q_{AGV_3} \times Q_{AGV_4} \times \\ &\qquad Q_{AGV_5} \times \Pi_{1 \leq i \leq j} Q_{WST_i} \\ Zone_2 &= Q_{AGV_1} \times E^2_2 \times E^3_2 \times \\ &\qquad Q_{AGV_4} \times Q_{AGV_5} \times \Pi_{1 \leq i \leq j} Q_{WST_i} \end{aligned}$$

where E^i_j is the set of states of AGV_i modeling the fact the i^{th} AGV 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 (The conflict zone 3 and 4 are similarly defined). ◇

As explained in Section 2, this problem can be reduced to the computation of the set of weak forbidden states $\mathcal{I}(E) = CoReach^G_{\Sigma_{uc}}(E)$ and an optimal supervisor ensuring the avoidance of E is then simply given by the formula (4). It is theoretically possible to compute $\mathcal{I}(E)$ on G as far as G can be efficiently represented by a single FSM. However, due to the state space explosion, this may be not feasible for concurrent systems. Moreover, the expression of the supervisor requires an on-line evaluation in order to determine the set of events that has to be disabled by control (given an event, one have to test whether the state, reached by triggering this event belong to $\mathcal{I}(E)$ or not). This evaluation is even better when the set $\mathcal{I}(E)$ is itself well structured.

3.3 The State avoidance Control Problem

In this section, 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 3.3.2, we focus on systems composed of components not sharing uncontrollable events, whereas Section 3.3.3 is devoted to the general case. But first, we present some properties related to the concurrent systems and the predecessor operator.

3.3.1. Properties of the Pre operator. The computation of the weak forbidden set of states $\mathcal{I}(E)$ is the results of the fix-point computation using the $Pre^G_{\Sigma_{uc}}$ operator. Hence, the fact that the plant we want to control is given by a concurrent system $G = G_1 \parallel \ldots \parallel G_n$, lead us to see how this operator can be locally computed.

Proposition 2. Let $G = G_1 \parallel \ldots \parallel G_n$ be a concurrent system and $E = E_1 \times \cdots \times E_n$ a set of states, then

$$Pre^G_A(E_1 \times \cdots \times E_n) = \bigcup_{\sigma \in A} \left(\prod_{i \in \text{IN}(\sigma)} Pre^{G_i}_{\{\sigma\}}(E_i) \right) \times \left(\prod_{k \notin \text{IN}(\sigma)} E_k \right) \tag{9}$$

Moreover, if $A \subseteq \Sigma \setminus \Sigma_s$, then

$$\left(Pre^G_A\right)^*(E_1 \times \cdots \times E_n) = (Pre^{G_1}_{\Sigma_1 \cap A})^*(E_1) \times \cdots \times (Pre^{G_n}_{\Sigma_n \cap A})^*(E_n) \tag{10}$$

3.3.2. The case $\Sigma_s \subseteq \Sigma_c$. Let us consider a plant $G = G_1 \parallel \cdots \parallel G_n$, such that the uncontrollable events are local to each component of the plant (i.e. $\Sigma_s \subseteq \Sigma_c$).

Efficient computation of $\mathcal{I}(E)$. First we assume that E is reduced to a product set $E = E_1 \times \cdots \times E_n$ (with $E_i \subseteq Q_i$). The next proposition shows that

the $\mathcal{I}(\cdot)$ operator can be expressed using only local computations performed on each component.

Proposition 3. Let us consider $G = G_1 \parallel \ldots \parallel G_n$, such that $\Sigma_s \subseteq \Sigma_c$. Let $E = E_1 \times \cdots \times E_n$, with $E_i \subseteq Q_i$. Then,

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

where $\mathcal{I}_i(E_i)$ represents the weak forbidden set of states w.r.t. E_i and G_i.

Proof : By definition,

$$\begin{aligned}\mathcal{I}(E) &= CoReach^{G}_{\Sigma_{uc}} = \left(Pre^{G}_{\Sigma_{uc}}\right)^{*}(E)\\ &= \left(Pre^{G}_{\Sigma_{uc}}\right)^{*}(E_1 \times \cdots \times E_n)\end{aligned}$$

Moreover $\Sigma_{uc} \subseteq \Sigma \setminus \Sigma_s$ and $\Sigma_{i,uc} = \Sigma_i \cap \Sigma_{uc}$. Hence according to Propositions 2 (formula (10)):

$$\begin{aligned}\mathcal{I}(E) &= {Pre^{G_1}_{\Sigma_{1,uc}}}^{*}(E_1) \times \cdots \times {Pre^{G_n}_{\Sigma_{n,uc}}}^{*}(E_n)\\ &= CoReach^{G_1}_{\Sigma_{1,uc}} \times \cdots \times CoReach^{G_n}_{\Sigma_{n,uc}}\\ &= \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n)\end{aligned}$$ ◇

The above proposition gives us an efficient way to compute $\mathcal{I}(E)$, in the sense that the computation can be locally performed on each sub-system. We now give a modular description of $\mathcal{I}(E)$ for the general case (i.e. when the set of states is given by a union of product sets.

Proposition 4. Consider $G = G_1 \parallel \ldots \parallel G_n$, such that $\Sigma_s \subseteq \Sigma_c$. Let $E = \bigcup_{1\leq j\leq m} E^j_1 \times \cdots \times E^j_n$, with $E^j_i \subseteq Q_i$. Then,

$$\mathcal{I}(E) = \bigcup_{1\leq j\leq m} \left(\prod_{1\leq i\leq n} \mathcal{I}_i(E^j_i) \right) \qquad (11)$$

The proof of this property is simply based on the fact that $\mathcal{I}(E \cup E') = \mathcal{I}(E) \cup \mathcal{I}(E')$. Finally, in order to compute $\mathcal{I}(E)$, it is sufficient to compute on each component the sets of states $(\mathcal{I}_i(E^j_i))_{j\leq m, i\leq n}$. Based on (11), the overall complexity is in $\mathcal{O}(m.n.k.N)$, where n is the number of components, $N = max_i(|Q_i|)$ the number of states of each component and $k = max_i(|\Sigma_i|)$. It is worthwhile noting that if $\mathcal{I}(E)$ is expanded then it may be of the size of the global system and then unfeasible to compute. In order to avoid to store in memory this large set of states (i.e. $\mathcal{I}(E)$), we prefer to store in memory local set of states (i.e. the $\mathcal{I}_i(E^j_i)$) and to perform some on-line computations. This is the aim of the Corollary 1.

Example 2. Let us consider the well known Cat & Mouse example. 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.

The set of forbidden global states can be decomposed in 5 product sets

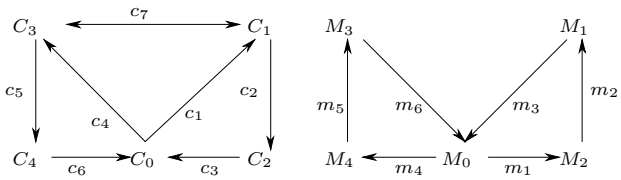


Fig. 1. The cat an mouse example

$$E = \bigcup_{1\leq j\leq 5} E^j \text{ avec } \forall 1 \leq j \leq 5,\ E^j = \{C_j\}\times\{M_j\}$$

Now, according to Proposition 4, $\mathcal{I}(E) = \bigcup_{1\leq j\leq 5} \mathcal{I}(E^j)$, with

$\mathcal{I}(E^1) = \{C_1, C_3\} \times \{M_1\}$, $\mathcal{I}(E^2) = \{C_2\} \times \{M_2\}$
$\mathcal{I}(E^3) = \{C_3, C_1\} \times \{M_3\}$, $\mathcal{I}(E^4) = \{C_4\} \times \{M_4\}$
$\mathcal{I}(E^5) = \{C_5\} \times \{M_5\}$

◇

On-line supervision. Finally, given $\mathcal{I}(E)$ as in (11), one can easily extract a supervisor as follows

Corollary 1. Let $G = G_1 \parallel \ldots \parallel G_n$, such that $\Sigma_s \subseteq \Sigma_c$, and E a set of states of G. With the notations of Corollary 4, the supervisor S defined for all $q \in Q$ by

$$\begin{aligned}S_E(q) = \{&\sigma \in \Sigma_c |\ \delta(q,\sigma)! \wedge\\ &\delta(q,\sigma) \in \bigcup_{1\leq j\leq m} (\mathcal{I}_1(E^j_1) \times \cdots \times \mathcal{I}_n(E^j_n))\}\end{aligned}$$

ensures the avoidance of E and is maximal. ◇

The proof follows directly from the expression of the supervisor (4) and Proposition 4.

Let us now see how the expression given in Corollary 1 allows en efficient on-line evaluation of $S_E(q)$. To do so, it is sufficient, for each $\sigma \in \Sigma_c$ to determine $\delta(\sigma, q) = \langle q'_1, \ldots, q'_n\rangle$ and to test whether $\langle q'_1, \ldots, q'_n\rangle \in \mathcal{I}(E)$ or not. According to (11) we just have to test that there exists $j \in \{1, \ldots, m\}$ such that $\forall i \in \{1, \ldots, n\}$, $q'_i \in \mathcal{I}_i(E^j_i)$. This can be done in $\mathcal{O}(|\Sigma_c|m.n.log(N))$, which is an acceptable complexity as far as the objectives are well structured. Note that only the $(\mathcal{I}_i(E^j_i)), i, j$ need to be stored in memory. The supervisor is only evaluated on the fly along the execution of the system.

3.3.3. General case. In the previous section, we have shown an efficient method allowing the computation of a supervisor ensuring the avoidance of a set of forbidden set of states for concurrent discrete event systems for which the shared events were assumed to be controllable. Even though there exist numerous systems that respect this hypothesis, it is also of interest to consider the case where some shared events are uncontrollable.

We thus consider a concurrent system $G = G_1 \parallel \cdots \parallel G_n$ an a set of states E. As the shared event are not supposed to be controllable anymore, the proposition 3 is no more valide. We thus need to refine the computation of the local weak forbidden set of states $\mathcal{I}_i(\cdot)$ in order to take into account that some shared event are uncontrollable.

Forbidden states and local events. Consider a Concurrent system $G = G_1 \parallel \ldots \parallel G_n$ and $E \subseteq Q$, we define

$$\mathcal{I}_{loc}(E) = CoReach^{G}_{\Sigma_{uc}\setminus\Sigma_s}(E)$$

and for a set of states E_i of G_i,

$$\mathcal{I}_{i,loc}(E_i) = CoReach^{G_i}_{\Sigma_{i,uc}\setminus\Sigma_s}(E_i)$$

$\mathcal{I}_{loc}(E)$ (resp. $\mathcal{I}_{i,loc}(E_i)$) represents the set of states of G (resp. G_i) from which it is possible to evolve into E (resp. E_i) by triggering a sequence of events which are uncontrollable and only local (i.e. with no shared event). Based on these sets, we obtain the following result:

Proposition 5. Let $G = G_1 \parallel \cdots \parallel G_n$ be a concurrent system and E a set of states of G such that $E = E_1 \times \cdots \times E_n$, with $E_i \subseteq Q_i$. Then,

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

◇

The proof of Proposition 5 is similar to the one of Proposition 3. Moreover, according to the monotonic property of the operator $\mathcal{I}_{loc}(\cdot)$, we also have that

$$\forall E, E',\ \mathcal{I}_{loc}(E \cup E') = \mathcal{I}_{loc}(E) \cup \mathcal{I}_{loc}(E')$$

The result of Proposition 5 can thus be easily generalized to any set of states:

Corollary 2. With the notations of Proposition 5, if $E = \bigcup_j E^j$ with $\forall j,\ E^j = E^j_1 \times \cdots \times E^j_n$ and $\forall i, j,\ E^j_i \subseteq Q_i$, then

$$\mathcal{I}_{loc}(E) = \bigcup_{j \leq m} \left(\prod_{i \leq n} \mathcal{I}_{i,loc}(E^j_i) \right)$$

One can easily note that $\mathcal{I}_{loc}(E) \subseteq \mathcal{I}(E)$, where $\mathcal{I}(E)$ is computed as in (3) on the whole system G. Equality is not met since the states that are traversed by an uncontrollable trajectory leading to E, and having at least one shared event are not taken into account.

We are now interested in finding conditions under which $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$ (one can remark that according to the previous section, $\Sigma_s \subseteq \Sigma_c$ is a sufficient condition. In this case, $\mathcal{I}_{i,loc} = \mathcal{I}_i$). Now, in order to characterize a necessary condition, we introduce the following operator $\mathcal{F}^{\sigma}_{G}(\cdot)$.

Définition 2. Let G be a concurrent system s.t. $\mathcal{L}(G) \subseteq \Sigma^*$, $\sigma \in \Sigma$, and E a set of states of G,

$$\mathcal{F}^{\sigma}_{G}(E) = Pre^{G}_{\{\sigma\}}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) \qquad (12)$$

$\mathcal{F}^{\sigma}_{G}(E)$ represents the set of states of G from which it is possible to reach $\mathcal{I}_{loc}(E)$ by triggering σ, but that do not belong to $\mathcal{I}_{loc}(E)$ [1]. Based on these sets, it is now possible to define a condition under which $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$.

Proposition 6. Consider $G = G_1 \parallel \cdots \parallel G_n$ and E a set of states of G. Then,

$$\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}^{\sigma}_{G}(E) = \emptyset \iff \mathcal{I}(E) = \mathcal{I}_{loc}(E)$$

Proof : ($\Leftarrow$) By definition,

$$\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}^{\sigma}_{G}(E) = Pre^{G}_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$$

We also have that $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$ (by hypothesis), thus

$$\mathcal{F}^{\sigma}_{G}(E) = Pre^{G}_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}(E)) \setminus \mathcal{I}(E)$$

Moreover, for $\sigma \in \Sigma_{uc}$, $Pre^{G}_{\sigma}(\mathcal{I}(E)) = \mathcal{I}(E)$. Thus, $\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}^{\sigma}_{G}(E) = \emptyset$.

($\Rightarrow$) By definition, $E \subseteq \mathcal{I}_{loc}(E) \subseteq \mathcal{I}(E)$ (α)

Now as $\mathcal{I}$ is monotonic, $\mathcal{I}(E) \subseteq \mathcal{I}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}(\mathcal{I}(E))$. Moreover $\mathcal{I}(\mathcal{I}(E)) = \mathcal{I}(E)$, from which we can deduce that

$$\mathcal{I}(\mathcal{I}_{loc}(E)) = \mathcal{I}(E) \qquad (\gamma)$$

Moreover, $\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}^{\sigma}_{G}(E) = \emptyset$ (by hypothesis), which entails that

$$Pre^{G}_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) = \emptyset$$

We can then deduce $Pre^{G}_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$. Moreover $Pre^{G}_{\Sigma_{uc}\setminus\Sigma_s}(\mathcal{I}_{loc}(E)) = \mathcal{I}_{loc}(E)$ (by definition). Overall, we obtain

$$Pre^{G}_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}_{loc}(E)) \cup Pre^{G}_{\Sigma_{uc}\setminus\Sigma_s}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E),$$

from which we can deduce that $Pre^{G}_{\Sigma_{uc}}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$. Now as the operator $Pre^{G}_{\Sigma_{uc}}$ is monotonic, we thus have that

$$\forall n \geq 1,\ (Pre^{G}_{\Sigma_{uc}})^{(n)}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$$

which entails that $\mathcal{I}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$ and using (γ) we can deduce that $\mathcal{I}(E) \subseteq \mathcal{I}_{loc}(E)$, which once combined with (α) gives the result. ◇

Proposition 6 gives a necessary and sufficient condition under which $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$. In particular, if $\cup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}^{\sigma}_{G}(E) = \emptyset$, the maximal supervisor ensuring the avoidance of E is given for all $q \in Q$ by :

$$S_E(q) = \{\sigma \in \Sigma_c | \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}_{loc}(E)\} \qquad (13)$$

Complexity discussion. Proposition 6 is interesting from an algorithm point of view in the sense that $\mathcal{I}(E)$ is given by a union of product sets. Indeed with the notations of Proposition 6, $\mathcal{I}(E)$ can be computed using only the sets $(\mathcal{I}_i(E^j_i))_{i,j}$, which reduces the off-line computation to the one of these sets. The overall complexity is in $\mathcal{O}(m.n.k.N)$. By definition of $\mathcal{F}^{\sigma}_{G}(E)$, in

[1] Note that, according to Proposition 2, $Pre^{G}_{\{\sigma\}}(\mathcal{I}_{loc}(E))$ can be locally computed w.r.t. the local components.

order to check the condition of Proposition 6 one has to check that for all $\sigma \in \Sigma_{uc} \cap \Sigma_s$,

$$Pre^G_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E) \qquad (14)$$

It is worthwhile noting that the number of states of $Pre^G_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}_{loc}(E))$ may be of the size of the global system. However, for loosely synchronous system for which most of the behavior of the components is local, checking the inclusion (14) is hoped to be tractable.

Further, one has also to take into account the computations that have to be done on-line when controlling the plant (i.e when computing the supervisor on the fly). Indeed, deciding whether the next state belong to the set of forbidden states is done at execution time. According to the previous section, the evaluation is in $\mathcal{O}(|\Sigma_c|m.n.log(N))$.

Example 3. Coming back to the AGV example, it is easy to see that the above condition is checked. Using our tool SYNTOOL, based on enumerative methods, the computation of $\mathcal{I}(E)$ is performed in less than 2 second, whereas the on-line evaluation of the supervisor, during a simulation, is immediate. ◇

If $\bigcup_{\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(E) \neq \emptyset$, then according to Proposition 6, $\mathcal{I}(E) \neq \mathcal{I}_{loc}(E)$. In this case, one can remark that the weak forbidden states that are not taken into account are the ones that lead to E via an uncontrollable sequence having shared events. The idea is then to consider this states are forbidden and to iterate until the previous condition is satisfied [2]. Obviously, the price to be paid is the increase of the number of product sets inducing a more important on(off)-line complexity [3].

In order to capture these states, we introduce the following function Φ:

Définition 3. Consider $G = G_1 \parallel \cdots \parallel G_n$ and $E \subseteq Q$. Then the function $\Phi : 2^Q \to 2^Q$ is defined by:

$$\Phi(E) = E \bigcup_{\sigma\in\Sigma_s\cap\Sigma_{uc}} \mathcal{F}^\sigma_G(E)$$

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 shared and uncontrollable. 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 $(\Phi^n(E))_{n\geq o}$. This limit always exists and is reached in a finite number of iterations, since the number of states of the system is finite.

Based on $\Phi(\cdot)$ and $\Phi^*(\cdot)$, we can show the two following lemmas

Lemma 1. Consider $G = G_1 \parallel \cdots \parallel G_n$ and $E \subseteq G$. Then,

$$\bigcup_{\sigma\in\Sigma_s\cap\Sigma_{uc}} \mathcal{F}^\sigma(\Phi^*(E)) = \emptyset$$

Proof : Let $E \subseteq Q$. According to the definition of $\Phi^*(\cdot)$, $\Phi^*(E) = \Phi(\Phi^*(E))$. Moreover, by definition of $\Phi(\cdot)$,we have

$$\Phi(\Phi^*(E)) = \Phi^*(E) \bigcup \left(\bigcup_{\sigma\in\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(\Phi^*(E)) \right)$$

Hence, we can deduce that

$$\Phi^*(E) = \Phi^*(E) \bigcup \left(\bigcup_{\sigma\in\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(\Phi^*(E)) \right)$$

and thus $\left(\bigcup_{\sigma\in\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(\Phi^*(E))\right) \subseteq \Phi^*(E)$. Moreover, by definition of $\mathcal{F}^\sigma_G(E)$ [4],

$$\left(\bigcup_{\sigma\in\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(\Phi^*(E)) \right) \cap \mathcal{I}_{loc}(\Phi^*(E)) = \emptyset$$

and as $\Phi^*(E) \subseteq \mathcal{I}_{loc}(\Phi^*(E))$, we can deduce that

$$\left(\bigcup_{\sigma\in\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(\Phi^*(E)) \right) \cap \Phi^*(E) = \emptyset$$

Moreover, since by definition $\left(\bigcup_{\sigma\in\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(\Phi^*(E))\right) \subseteq \Phi^*(E)$, we have that

$$\left(\bigcup_{\sigma\in\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(\Phi^*(E)) \right) = \emptyset$$

◇

Let us now compare $\Phi^*(E)$ with $\mathcal{I}(E)$.

Lemma 2. Consider $G = G_1 \parallel \cdots \parallel G_n$ and $E \subseteq Q$. $\mathcal{I}(\Phi^*(E)) = \mathcal{I}(E)$

Proof : ($\subseteq$) 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 assume that $\Phi^n(E) \subseteq \mathcal{I}(E)$. As Φ is monotonic,

$$\Phi^{n+1}(E) \subseteq \Phi(\mathcal{I}(E)) \qquad (\alpha)$$

Moreover, by definition $\Phi(\cdot)$,

$$\Phi(\mathcal{I}(E)) = \mathcal{I}(E) \bigcup \left(\bigcup_{\sigma\in\Sigma_{uc}\cap\Sigma_s} \mathcal{F}^\sigma_G(\mathcal{I}(E))\right) \qquad (\beta)$$

Now, for all $\sigma \in \Sigma_{uc} \cap \Sigma_s$,

$$\mathcal{F}^\sigma_G(\mathcal{I}(E)) = Pre^G_{\{\sigma\}}(\mathcal{I}_{loc}(\mathcal{I}(E))) \setminus \mathcal{I}_{loc}(\mathcal{I}(E)),$$

[2] Note that we still want $\mathcal{I}(E)$ to be expressed by means of a union of product sets in order to facilitate the on-line evaluation of the supervisor.

[3] The number of new product sets depends on the system and thus is not possible to characterize.

[4] C.f. formula (12)

by definition of $\mathcal{I}(\cdot)$ and $\mathcal{I}_{loc}(\cdot)$, we obtain $\mathcal{I}_{loc}(\mathcal{I}(E)) = \mathcal{I}(E)$. Since by definition of $\mathcal{I}(E)$,

$$\forall \sigma \in \Sigma_{uc},\ Pre^{G}_{\{\sigma\}}(\mathcal{I}(E)) \subseteq \mathcal{I}(E).$$

Hence, $\forall \sigma \in \Sigma_{uc} \cap \Sigma_s$, $\mathcal{F}^{\sigma}_{G}(\mathcal{I}(E)) = \emptyset$. We then deduce from (β) that $\Phi(\mathcal{I}(E)) = \mathcal{I}(E)$ and from (α) that $\Phi^{n+1}(E) \subseteq \mathcal{I}(E)$. Consequently, $\forall n \geq 0$, $\Phi^n(E) \subseteq \mathcal{I}(E)$. We thus obtain that $\Phi^*(E) \subseteq \mathcal{I}(E)$. Finally, as $\mathcal{I}$ is monotonic, $\mathcal{I}(\Phi^*(E)) \subseteq \mathcal{I}(\mathcal{I}(E)) = \mathcal{I}(E)$

$(\supseteq)$ As $E \subseteq \Phi^*(E)$, we easily obtain that $\mathcal{I}(E) \subseteq \mathcal{I}(\Phi^*(E))$. Hence the result. ◇

Proposition 7. Consider $G = G_1 \parallel \cdots \parallel G_n$ and E a set of states of G. Then, $\mathcal{I}(E) = \mathcal{I}_{loc}(\Phi^*(E))$

Proof : The proof can be directly obtained from Lemmas 1 and 2 and Proposition 6. ◇

The previous proposition gives us a method to locally compute $\mathcal{I}(E)$ without having to build the global system 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 computation of $\mathcal{I}(E)$ heavily depends on the number of product sets that is used to express E as well as the number of iterations needed to compute Φ^*. Indeed, at each iteration, new product sets need to be forbidden (i.e. the one that belongs to $\mathcal{F}^{\sigma}_{G}(\Phi^i(E))$ 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).

Corollary 3. Let $G = (\Sigma, Q, q_0, Q_m, \delta)$ be a concurrent system and $E \subseteq Q$. With the notations of Proposition 7, the maximal supervisor E ensuring the avoidance of E is defined for all $q \in Q$ by :
$S_E(q) = \{\sigma \in \Sigma_c |\ \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}_{loc}(\Phi^*(E))\}$
◇

The corollary 3 can be directly deduced from the expression (4) and from the proposition 7. From a computational point, we know from Proposition 5 that $\mathcal{I}_{loc}(\Phi^*(E))$ is indeed computed and expressed using the local sets $\mathcal{I}_{i,loc}()$, thus reducing the complexity of the on-line computation/evaluation phase (according to the above complexity discussion, it is in $\mathcal{O}(|\Sigma_c|.M.n.log(N))$ [5] . Note that as previously mentioned M may be important.

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 states 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 on each component leading to a global supervisor ensuring the avoidance of E, that has to be computed on the fly. Let us now emphasize some points that we did not mention so far:

- One can note that, in order to increase the efficiency of the methods, all the algorithms given in this paper can be easily implemented using BDD.
- Based on the results, it is easy to extend the methodology presented in [Gaudin and Marchand (2004b)] and to solve the SACP for a synchronous hierarchical finite state machines with two levels.

[5] where M is the number of product sets that are necessary to express $\Phi^*(E)$

REFERENCES

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

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

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.

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

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

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

R.S. Minhas. *Complexity eduction in Discrete Event Systems*. PhD thesis, Univeristy of Toronto, September 2002.

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

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.

A. Vahidi, B. Lennarston, and M. Fabian. Efficient supervisory synthesis of large systems. In *Workshop on Discrete Event Systems, WODES'04*, 2004.

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

W. M. Wonham. Notes on control of discrete-event systems. Technical Report ECE 1636F/1637S, Department of ECE, University of Toronto, July 2003.