Supervisory Control and Deadlock Avoidance Control Problem for Concurrent Discrete Event Systems

Benoit Gaudin, Hervé Marchand

Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Supervisory Control and Deadlock Avoidance Control Problem for Concurrent Discrete Event Systems

Benoit Gaudin and Hervé Marchand
IRISA, Campus Universitaire de Beaulieu, 35042 Rennes, France
{Benoit.Gaudin,Herve.Marchand}@irisa.fr

***Abstract*— We tackle the Non-Blocking Supervisory Control Problem control for Concurrent Discrete Event Systems that are defined by a collection of components that interact with each other. In this study, we first outline the method allowing to solve the state avoidance control problem on concurrent systems. We then present results offering an efficient method to detect deadlock states in the controlled system, which once combined with the previous method, allows us to solve a particular class of the non-blocking state avoidance control problem.**

I. Introduction

Supervisory control ([1], [2]) consists in modifying a *system* (plant) such that the modified (or *controlled*) system satisfies a given *specification* (e.g. a *safety property*). We focus on the control of Concurrent Discrete Event Systems defined by a collection of components that interact with each other. Several approaches have been considered to take into account the structure of concurrent systems ([3], [4], [5], [6]). 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. It may happen that the specification 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 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. [1]). Note that if one wants to use a language-based approach (as in e.g. [7]) to encode this problem, then the obtained specification does not fit with the structure of the system (it may be not separable), and may be of the size of the global system itself. This renders the use of the above works useless or at least intractable. These reasons lead us to develop techniques totally devoted to the state avoidance control problem. Following the methodology described in [8] and [9], we have developed in [10] a methodology that decompose the computation in two phases (an off-line and an on-line computation) leading to a supervisor than can be easily evaluated on the fly.

In [10], the non-blocking aspect was not taken into account, which means that, once controlled the system may be blocking. In this paper, we focus on the deadlock avoidance control problem which consists in ensuring that all the deadlock states (from which no action can be triggered) can not be reachable in the controlled system. It turns out that there exists two types of deadlock states: the one induced by composition and the one induced by control. For the first one, we adapt to our model the techniques described in [11] in order to efficiently extract deadlock states from a concurrent system without having to build the system itself. This deadlock states are then added to the set of forbidden states. Further the action of the supervisor may add some new deadlock states in the controlled system. However, we can not reuse as such the first method to detect them because the controlled system is no more a concurrent system. It is actually given by a concurrent system that is constrained by a supervisor. We thus provide two methods based on the structure of the system and of the supervisor that can be used in parallel to detect such deadlock states. The first one is based on the characterization of these states according to the status of the events that have been removed by control, whereas the second is based on an incremental detection of the set of deadlock states according to some projections of the system and of the supervisor that have the property to keep the deadlock status of the states.

II. Preliminaries

Each component of the system is modeled as Finite State Machines (FSM), defined by a 4-tuple $G = (\Sigma, Q, q_o, \delta)$, where Σ is a finite alphabet. 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 $\text{Pre}_A^G(E) = E \cup \{q \in Q | \ \exists \sigma \in A,\ \delta(\sigma, q) \in E\}$, which corresponds to states that can reach the set E by triggering at most one event belonging to A.

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. We 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*), which consists in computing a maximal supervisor ensuring that a set of

0-7803-9568-9/05/$20.00 ©2005 IEEE

forbidden states cannot be reached in the controlled system. In order to solve this problem, we first introduce the *weak forbidden set* $\mathcal{I}$:

$$\mathcal{I}(E) = CoReach^{G}_{\Sigma_{uc}}(E) = \bigcup_{n\geq 0} \mathrm{Pre}^{G^{(n)}}_{A}(E) \quad (1)$$

$\mathcal{I}(E)$ is the set of states from which it is possible to evolve into E triggering a sequence of uncontrollable events.

Proposition 1: Given an FSM G and a set of states $E \subseteq Q$, if $q_0 \notin \mathcal{I}(E)$, then the supervisor $\mathcal{S}_E$ of G, s.t. $\forall q \in Q$

$$\mathcal{S}_E(q) = \{\sigma \in \Sigma_c |\ \delta(\sigma, q)! \wedge \delta(\sigma, q) \in \mathcal{I}(E)\} \quad (2)$$

is one of the most permissive supervisors ensuring the avoidance of E in G.

III. SACP for Concurrent Systems

Let us 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 $G = G_1 \parallel \cdots \parallel G_n$ where the operation $\parallel$ is the classical *parallel composition* (i.e. $G_1 \parallel G_2$ represents the concurrent behavior of G_1 and G_2 with synchronization on the shared events). The resulting FSM will be noted $\langle \Sigma, Q, q_o, \delta \rangle$ with $\Sigma = \cup_i \Sigma_i$. 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 $\sigma \in \Sigma_i$. For each G_i, we have $\Sigma_i = \Sigma_{i,uc} \cup \Sigma_{i,c}$. The alphabet of the global plant G is given by $\Sigma = \cup_i \Sigma_i,\ \Sigma_c = \cup_i \Sigma_{i,c}$, and $\Sigma_{uc} = \Sigma \setminus \Sigma_c$. Moreover, we assume that the following relation holds between the control status of shared events, i.e. $\forall i, j,\ \Sigma_{i,uc} \cap \Sigma_{j,c} = \emptyset$, which simply means that the components that share an event agree on the control status of this event. Under this hypothesis, we have that $\Sigma_{uc} = \cup_i \Sigma_{i,uc}$.

In [10], the problem under investigation was the SACP for concurrent systems. We here briefly recall the main results given in this paper. It can be shown that a set of states E can be expressed by a union of product sets $E = \bigcup_{k\leq N} E^k$ with $E^k = E^k_1 \times \cdots \times E^k_n$ with $E^k_i \subseteq Q_i$. Given the concurrent structure of the system, this decomposition of sets in terms of product sets will be the basis for the expression of states that will have to be forbidden by control. As explained in Section II, 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 (2). However, due to the state space explosion, this computation may be unfeasible for concurrent systems. This is why specific methods, taking into account the structure of the system, have been developed in [10]. We here focus on concurrent systems $G = G_1 \parallel \cdots \parallel G_n$, for which the uncontrollable events are local to each component (i.e. $\Sigma_s \subseteq \Sigma_c$)[1]. Under this hypothesis, it is shown in [10] that

$$\mathcal{I}(E) = \bigcup_{k\leq N} \left(\times_i \mathcal{I}(E^k_i)\right) \quad (3)$$

[1]To simplify the paper, the general case (i.e. when $\Sigma_s \cap \Sigma_{uc} \neq \emptyset$) is not presented here (See [10] for the details).

Now, given $\mathcal{I}(E)$ as in (3), one can easily extract a supervisor as follows

$$\mathcal{S}_E(q) = \{\sigma \in \Sigma_c |\ \delta(\sigma, q)! \wedge \delta(\sigma, q) \in \bigcup_{k\leq N} \left(\times_i \mathcal{I}(E^k_i)\right)\} \quad (4)$$

The expression of the supervisor given by (4) is interesting for computational reasons. Indeed, whereas concurrent systems can be very complex, determining S_E only consists here in computing sets $(\mathcal{I}(E^k_i))_{1\leq i\leq n, 1\leq k\leq N}$. The counter part of this approach is that on-line computations are needed during the execution phase, which is not the case when the system is modeled by one FSM (See [10] for details).

IV. Deadlock Avoidance control problem

Next, we are interested in deadlock avoidance. Intuitively, given a system $G = G_1 \parallel \cdots \parallel G_n$, a deadlock state is a state from which no event can be triggered in the system (i.e. $q \in Q$ is a deadlock state of G if $\delta(q) = \emptyset$). An FSM (or a plant) is said to be *deadlock-free* if no state is in deadlock. The *Deadlock Avoidance Control Problem* (DACP) consists in computing a supervisor $\mathcal{S}$ ensuring that all the deadlock states are no more reachable in the controlled system.

The classical method that is used to solve this problem is given by the following iteration:

(a) Extraction of the set of deadlock states in G, say D_G.
(b) Computation of the possibly blocking supervisor $\mathcal{S}$ w.r.t. to G and D_G ensuring avoidance of D_G.
(c) Repeat points (a) and (b) with $G := S/G$ until S/G is deadlock-free.

Remark 1: Note that if one want to prevent, at the same time, a set of states E to be reachable in G, it is sufficient to control the system so that the set $E \cup D_G$ becomes not reachable (point (b)) and that the obtained system is deadlock-free, in order to obtain a maximal deadlock-free supervisor ensuring the avoidance of E.

This is the method we want to be applied on system G. From Section III, we already know how to efficiently solve point (b). Now all the problem consists in extracting the deadlock states without having to build the whole system (point (a)). One can note that initially G is a concurrent system whereas after an iteration, we need to extract deadlock states from a controlled system that is no more a concurrent system (i.e. the controlled system is given by G and a supervisor $\mathcal{S}$). We thus need specific methods to characterizes deadlock-states in order to reduce the complexity of this computation by reducing the state space in which the deadlock states can be searched. In Section IV-A, we focus on the characterization of deadlock states in a concurrent system, whereas in Section IV-B, we outline a methodology allowing to efficiently extract deadlock states from a controlled system $\mathcal{S}/G$.

A. Characterization of the deadlock states of G

When dealing with a concurrent system, the method that is used to characterize the deadlock states is quite classical (see e.g. [11]). We here recall the main aspects of this method. Let us consider $G = G_1 \parallel \cdots \parallel G_n$ a concurrent

system, such that $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$. The set of states of G is given by $Q = Q_1 \times \cdots \times Q_n$. Now, it is worthwhile noting that the fact that a state $q = (q_1, \ldots, q_n)$ is in deadlock in G is due to the synchronization between the different components of G. This entails that

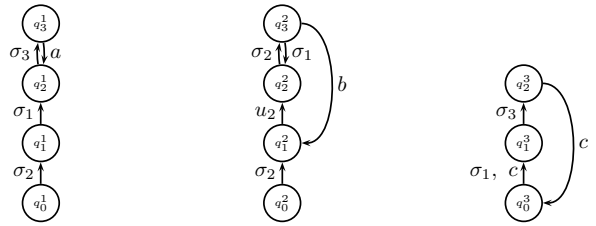
$$q \text{ is in deadlock in } G \Longrightarrow \forall i \leq n,\ \delta_i(q_i) \subseteq \Sigma_s \tag{5}$$

where Σ_s denotes the set of shared events of G. In other words, a state may be in deadlock if only shared events can be locally triggered from each component. $\forall i \leq n$, let us denote by $\text{PB}_i(G_i)$ the set $\{q_i \in Q_i |\ \delta_i(q_i) \subseteq \Sigma_s\}$. Now if D_G denotes the set of deadlock states in G, we have that

$$D_G \subseteq \text{PB}(G) \triangleq \text{PB}_1(G_1) \times \cdots \times \text{PB}_n(G_n) \tag{6}$$

(6) is interesting in the sense that it reduces the state space in which a deadlock state can be, since the size of set $\text{PB}_1(G_1) \times \cdots \times \text{PB}_n(G_n)$ can be expected to be much smaller than that of the whole system G. This is actually the case when the concurrent system is *loosely synchronized* (i.e. the shared events are triggered with a few frequency).

Example 1: Let us consider $G = G_1 \parallel G_2 \parallel G_3$ (Fig. 1).



(a) G_1 (b) G_2 (c) G_3

Fig. 1. A concurrent system containing deadlock states

The alphabet of each subsystem is given by $\Sigma_1 = \{a, \sigma_1, \sigma_2, \sigma_3\}$, $\Sigma_2 = \{b, u_2, \sigma_1, \sigma_2\}$ and $\Sigma_3 = \{c, \sigma_1, \sigma_3\}$. The only uncontrollable event is u_2 and is a local event of subsystem G_2. We also have that $\Sigma_s = \{\sigma_1, \sigma_2, \sigma_3\}$. The set PB of possible deadlock in G is given by

$$PB = \{q_0^1, q_1^1, q_2^1\} \times \{q_0^2, q_2^2\} \times \{q_1^3\}$$

and according to (6), $PB(G)$ contains the deadlock states of G, which means that, in this case, only 6 states have to be checked, whereas the size of Q is 48. Finally, checking for each of these states whether some events can be triggered from them, we obtain that only (q_1^1, q_0^2, q_1^3) and (q_1^1, q_2^2, q_1^3) are in deadlock in G. ◇

The characterization of deadlock states given by equation (5) can help to perform the point (a) of the method solving DACP. Nevertheless, this is only helpful at the first iteration of the method, because the method is based on the fact that G is a concurrent system, which is no more the case after the first iteration.

B. Characterization of deadlock states in S/G

In this section, we give useful results to characterize deadlock states of a controlled concurrent system. These characterizations are given using the structure of both the system under control (which is concurrent), and the supervisor (which is given by Equation (4)). Let G be a concurrent system and S a supervisor ensuring the avoidance of a set E given by $E = \bigcup_k E_1^k \times \cdots \times E_n^k$, with $\forall i, k\ E_i^k \subseteq Q_i$. Note that E can be seen as the set of deadlock states that have been found at the previous step of the computations. In the sequel, for theoretical reasons, we shall assume that the controlled system $\mathcal{S}/G$ can be modeled by a sub-FSM $(\Sigma, Q, q_o, \delta_{\mathcal{S}/G})$ of G, such that

$$\forall q \in Q, \delta_{\mathcal{S}/G}(q) = \delta_G(q) \setminus \mathcal{S}(q) \tag{7}$$

We say that q is a deadlock state of $\mathcal{S}/G$ if $\delta_{\mathcal{S}/G}(q) = \emptyset$.

Now, given this controlled system S/G, deadlock states of S/G are either deadlock states of G or states on which the supervisor is acting. For the latter, the state is in deadlock because the supervisor disables some events. There actually exists two different cases:

(1) either the supervisor only disables shared events,

(2) or the supervisor disables at least one local event.

Based on Equations (4) and (7), we now give two different results that will help to characterize deadlock states depending whether they are in deadlock because of (1) or (2).

Proposition 2: Let $G = G_1 \parallel \cdots \parallel G_n$ and S be a supervisor acting upon on G. If $q \in Q$ is in deadlock in S/G and $S(q) \subseteq \Sigma_s$, then $q \in PB(G)$ as defined by Equation (6).

Proof: Let $q = (q_1, \ldots, q_n)$ be a state of G, with $q_i \in Q_i$. Assume that q is a deadlock state of S/G, and $S(q) \subseteq \Sigma_s$ and $q \notin \text{PB}(G)$. Since $q \notin \text{PB}(G)$, $\exists i \leq n$ such that $\delta_i(q_i) \not\subseteq \Sigma_s$. Thus it exists $\sigma \in \Sigma_i \setminus \Sigma_s$, such that $\delta_i(q_i, \sigma)!$. Now, as σ is local we also have that $\delta(q, \sigma)!$. Now by assumption, $S(q) \subseteq \Sigma_s$, which means that $\sigma \notin S(q)$ and therefore q is not a deadlock state in S/G. ■

Proposition 2 entails that $\text{PB}(G)$ not only contains deadlock states of G, but also deadlock states of S/G from which only shared events are disabled by control.

Proposition 3 shows how to use the particular structure of the supervisor given by equation (4) to characterize the states that are in deadlock because at least one local event has been disabled by $\mathcal{S}$. But first we need to introduce the following notations:

Let $\mathcal{A}$ be a non empty subset of $\{1, \ldots, n\}$. When a set of states $E \subseteq Q$ of G is given as a union of product sets $E = \bigcup_{k \leq N} E^k$, with $E^k = E_1^k \times \cdots \times E_n^k$ and $\forall i, k$ $E_i^k \subseteq Q_i$, then the projection of E over $\mathcal{A}$ is given by

$$p_{\mathcal{A}}(E) = \bigcup_{k \leq N} \times_{i \in \mathcal{A}} E_i^k \tag{8}$$

Proposition 3: Let $G = G_1 \parallel \cdots \parallel G_n$ and S, defined by (4), be a supervisor acting upon G, ensuring the avoidance of $E = \bigcup_k E_1^k \times \cdots \times E_n^k$, with $\forall i, k\ E_i^k \subseteq Q_i$. If q is a deadlock state of S/G and $\sigma \in \Sigma_i \setminus \Sigma_s$ is such that $\sigma \in S(q) \cap \delta(q)$, then

$$p_{\{i\}^c}(q) \in p_{\{i\}^c}(E),$$

where $\{i\}^c$ is the complementary set of $\{i\}$ in $\{1, \ldots, n\}$.

Proof: Consider $q \in Q$ be such that $\exists \sigma \in \Sigma_i \setminus \Sigma_s$ such that $\delta(q, \sigma) = q' = (q'_1, \ldots, q'_n)$ and assume that q is a deadlock state in S/G, which means that $\delta_{S/G}(q) = \emptyset$.

Morevoer, we also have that $\forall j \neq i,\ q'_j = q_j$ and $q'_i = \delta_i(q_i, \sigma)$. Now, as $\delta_{S/G}(q) = \emptyset$, we have that $\delta(q) \subseteq S(q)$ and thus $\sigma \in S(q)$. We thus deduce from (4) that $q' \in \mathcal{I}(E)$. Moreover, as $E = \bigcup_{j \leq m} E^j$, $\exists j \leq m$ such that $q' \in \mathcal{I}(E^j_1) \times \cdots \times \mathcal{I}(E^j_n)$, from which we deduce that $p_{\text{IN}(\sigma)^c}(q') \in \times_{k \in \text{IN}(\sigma)^c} \mathcal{I}(E^j_k)$. Consequently, $\forall k \notin \text{IN}(\sigma)$, $q'_k \in \mathcal{I}(E^j_k)$. Now as $\text{IN}(\sigma) = \{i\}$, we have that $\forall k \notin \text{IN}(\sigma)$, $q'_k = q_k$ and we obtain that $\forall k \in \text{IN}(\sigma)^c$, $q_k \in \mathcal{I}(E^j_k)$.

Let us now show that $\forall k \in \text{IN}(\sigma)^c$, $q_k \in E^j_k$. To do so, let us assume that $q_k \notin E^j_k$. Let $k \in \text{IN}(\sigma)^c$. Since $q_k \in \mathcal{I}(E^j_k)$, if $q_k \notin E^j_k$, then $\exists \sigma' \in \Sigma_{uc,k} \setminus \Sigma_s$ such that $\delta_k(q_k, \sigma')!$.We thus obtain that $\delta(q, \sigma')!$. However, $\sigma' \in \Sigma_{uc}$ thus $\sigma' \notin S(q)$, which entails that $\sigma' \in \delta_{S/G}(q)$ (According to (7)). This contradicts the fact that q is a deadlock state in S/G and thus $\forall k \in \text{IN}(\sigma)^c$, $q_k \in E^j_k$, or in other words, $p_{\{i\}^c}(q) \in p_{\{i\}^c}(E)$. ■

According to the previous proposition, we know that if q is in deadlock in the controlled system because the supervisor ensuring the avoidance of E disables a local event of Σ_i, then $\forall j \neq i$, $q_j \in E^k_j$, or in other words,

$$q \in \bigcup_k \left(E^k_1 \times \cdots \times E^k_{i-1} \times Q_i \times E^{k+1}_i \cdots \times E^k_n\right) \qquad (9)$$

Hence, if a local event of Σ_i is disabled from state q, then it gives information about $n-1$ components of q. This characterization can be useful to reduce the state space in which looking for deadlock states of S/G.

Example 2: Let us consider the concurrent system of example 1 again. As shown in Example 1, states (q^1_1, q^2_0, q^3_1) and (q^1_1, q^2_2, q^3_1) are in deadlock in G. We thus make the use of the techniques presented in Section III to compute a supervisor $\mathcal{S}$ that ensures the avoidance of these two states. To this end, according to (4), we first compute the set of states $(\mathcal{I}(\{q^j_i\}))_{i,j}$ and we obtain that $\mathcal{I}(\{q^2_2\}) = \{q^2_1, q^2_2\}$ and $\mathcal{I}(\{q^j_i\}) = \{q^j_i\}$ for the others local states. From this sets we easily derive a supervisor $\mathcal{S}$ following (4). We can now use Proposition 2 and 3 to reduce the state space of S/G in which we have to identify deadlocks. First, using Proposition 2, we search for deadlock states of S/G that belongs to PB(G) (which only contains 2 tuples (see example 1))). The result gives us states (q^1_0, q^2_0, q^3_1) and (q^1_1, q^2_2, q^3_1) (which were already identified as deadlock states in G). Further, using Proposition 3, we can identify the states that are in deadlock because at least one local event have been disabled by control. In this example, only two states (q^1_1, q^2_0, q^3_1) and (q^1_1, q^2_2, q^3_1) are forbidden. Thus according to (9), we first identify deadlock states of the form $(q^1_1, q^2_2, -)$. We obtain that (q^1_1, q^2_2, q^3_0) is a deadlock state of S/G. Then looking for states of the form $(q^1_1, -, q^3_1)$, we obtain that (q^1_1, q^2_3, q^3_1) is a deadlock state of S/G ((q^1_1, q^2_2, q^3_1) is also detected but was already identified as a deadlock state of G). Morevoer, no deadlock states of the form $(-, q^2_2, q^3_1)$ exists. Applying the same technique with the forbidden state (q^1_1, q^2_0, q^3_1), we obtain that (q^1_1, q^2_0, q^3_1) and (q^1_1, q^2_3, q^3_1) also are deadlock states of S/G. ◇

In this section, we outline methods that allow to restrict the state space in which a state can be in deadlock. However, these methods somehow assume that the system is a loosely synchronous system and that the number of product sets that are forbidden is not too large.

V. Incremental Computation

To alleviate these problems we here provide a method based on the concurrent structure of the system that allows to incrementally detect the states that are in deadlock. For that purpose, we introduce the notion of projections of a supervisor and of a controlled system S/G.

First, we introduce several notions of projection, according to a concurrent system. Let $G = G_1 \parallel \cdots \parallel G_n$ with $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$. Let $\mathcal{A}$ be a non empty subset of $\{1, \ldots, n\}$. For simplicity, we say that a FSM G_i belongs to $\mathcal{A}$ if $i \in \mathcal{A}$. We denote $\Sigma_{s,\mathcal{A}}$ the set of events defined by

$$\Sigma_{s,\mathcal{A}} = \bigcup_{i \in \mathcal{A} \wedge j \notin \mathcal{A}} (\Sigma_i \cap \Sigma_j) \qquad (10)$$

$\Sigma_{s,\mathcal{A}}$ represents the set of shared events of G that are events of both one FSM belonging to $\mathcal{A}$ and one that does not.

Definition 1: The projection of system $G = G_1 \parallel \cdots \parallel G_n$ over $\mathcal{A}$, denoted $p_{\mathcal{A}}(G)$, is defined by

$$p_{\mathcal{A}}(G) = \parallel_{i \in \mathcal{A}} G^{\mathcal{A}}_i \qquad (11)$$

where $G^{\mathcal{A}}_i$ denotes the FSM $(\Sigma^{\mathcal{A}}_i, Q_i, q_{0_i}, \delta_i)$, with $\Sigma^{\mathcal{A}}_i = \Sigma_i \setminus \Sigma_{s,\mathcal{A}}$ and $\delta^{\mathcal{A}}_i$ is the restriction of δ_i from $Q_i \times \Sigma_i$ to $Q_i \times \Sigma^{\mathcal{A}}_i$. ◇

Now, if Σ' is a subset of Σ then $p_{\mathcal{A}}(\Sigma')$ represents the set of events of Σ' which belongs to $p_{\mathcal{A}}(G)$ (i.e belongs to $\bigcup_{i \in \mathcal{A}} \Sigma^{\mathcal{A}}_i$).

Definition 2: Let G be a concurrent system and E a set of states of G. Let S_E be the supervisor ensuring the avoidance of E in G as described by equation (4).

- *The projection of S_E w.r.t. $\mathcal{A}$, denoted $p_{\mathcal{A}}(S_E)$ is the supervisor defined for all state q of $p_{\mathcal{A}}(G)$ by*

$$p_{\mathcal{A}}(S_E)(q) = \{\sigma \in p_{\mathcal{A}}(\Sigma_c) | \\ \delta_{p_{\mathcal{A}}(G)}(q, \sigma)! \wedge \delta_{p_{\mathcal{A}}(G)}(q, \sigma) \in p_{\mathcal{A}}(\mathcal{I}(E))\}$$

- *The projection of the controlled system S_E/G w.r.t. $\mathcal{A}$, denoted $p_{\mathcal{A}}(S_E/G)$ is defined by*

$$p_{\mathcal{A}}(S_E/G) = p_{\mathcal{A}}(S_E)/p_{\mathcal{A}}(G) \qquad (12)$$

◇

$p_{\mathcal{A}}(S_E)$ can be seen as a supervisor ensuring the avoidance of $p_{\mathcal{A}}(\mathcal{I}(E))$ over $p_{\mathcal{A}}(G)$. $p_{\mathcal{A}}(S_E/G)$ is then the corresponding controlled system.

We now introduce lemma 1 which makes the link between the dynamic of the global system and the one of its projections.

Lemma 1: Let $G = \parallel_{1 \leq i \leq n} G_i$ be a concurrent system. We denote $G = (\Sigma, Q, q_0, \delta)$. Let $\mathcal{A}$ be a non empty subset of $\{1, \ldots, n\}$, $q \in Q$ and $\sigma \in \Sigma$, then

$$\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q), \sigma)! \\ \Longrightarrow \delta(q, \sigma)! \wedge [\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q), \sigma) = p_{\mathcal{A}}(\delta(q, \sigma))]$$

Proof: Assume that $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)!$. Therefore, $\text{IN}(\sigma) \subseteq \mathcal{A}$ and we can deduce that $\forall i \in \text{IN}(\sigma),\ \delta_i(q_i,\sigma)!$ which implies that $\delta(q,\sigma)!$. We now denote $q' = \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)$ and $q'' = \delta(q,\sigma)$. From (11)

$$(\forall i \in \text{IN}(\sigma),\ q'_i = \delta_i(q_i,\sigma)) \wedge (\forall i \notin \text{IN}(\sigma),\ q'_i = q_i)$$

Hence $\forall i \in \mathcal{A} \setminus \text{IN}(\sigma),\ q'_i = q_i$. Moreover, $\forall i \in \text{IN}(\sigma), \delta_i(q_i,\sigma) = q''_i$ and $\forall i \notin \text{IN}(\sigma),\ q''_i = q_i$. We can deduce that $\forall i \in \mathcal{A}, q'_i = q''_i$. Finally, we obtain $q' = p_{\mathcal{A}}(q'')$. Hence the result. ■

Lemma 1 shows that only events that can be triggered from a state q in G, can be triggered from $p_{\mathcal{A}}(q)$ in $p_{\mathcal{A}}(G)$. Moreover, the state reached triggering this event from $p_{\mathcal{A}}(q)$ in $p_{\mathcal{A}}(G)$ corresponds to the projection over $\mathcal{A}$ of the one reached from q in G.

The next lemma shows how the projection of a supervisor acts upon the projection of a system.

Lemma 2: Let $G = \|_{1\leq i\leq n}\ G_i$ be a concurrent system. We denote $G = (\Sigma, Q, q_0, \delta)$. Let $\mathcal{A}$ be a non empty subset of $\{1,\ldots,n\}$ and $q \in Q$. Consider a set of states $E \subseteq Q$ and S_E ensuring the avoidance of E, as described in (4). Then,

$$S_E(q) \cap p_{\mathcal{A}}(\Sigma) \subseteq p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$$

Proof: Let us consider $\sigma \in S_E(q) \cap p_{\mathcal{A}}(\Sigma)$. Since $\sigma \in p_{\mathcal{A}}(\Sigma)$, we obtain $\text{IN}(\sigma) \subseteq \mathcal{A}$ (according to Definition 1). Moreover, according to Equation 4,

$$\sigma \in S_E(q) \Longrightarrow \sigma \in \Sigma_c \wedge \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}(E)$$

$$\begin{aligned} \text{But } \delta(q,\sigma)! &\Longrightarrow \forall i \in \text{IN}(\sigma),\ \delta_i(q_i,\sigma)! \\ &\Longrightarrow \forall i \in \text{IN}(\sigma) \cap \mathcal{A},\ \delta_i(q_i,\sigma)! \\ &\Longrightarrow \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)! \end{aligned}$$

Therefore, from lemma 1, if $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)!$ then

$$\delta(q,\sigma)! \wedge [\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma) = p_{\mathcal{A}}(\delta(q,\sigma))] \qquad (\alpha)$$

But if $\delta(q,\sigma) \in \mathcal{I}(E)$, then $p_{\mathcal{A}}(\delta(q,\sigma)) \in p_{\mathcal{A}}(\mathcal{I}(E))$. Hence from (α), if $\delta(q,\sigma) \in \mathcal{I}(E)$, then $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma) \in p_{\mathcal{A}}(\mathcal{I}(E))$. Finally, we can deduce that

$$\sigma \in S_E(q) \Longrightarrow (\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)! \wedge \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)) \in p_{\mathcal{A}}(\mathcal{I}(E)))$$

which in turns implies that $\sigma \in p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$. ■

Based on this lemma, we can now prove the following property

Proposition 4: With the notations of lemma 2, if $q \in Q$ is a deadlock state of $\mathcal{S}_E/G$, then for all $\emptyset \subset \mathcal{A} \subseteq \{1,\ldots,n\}$, $p_{\mathcal{A}}(q)$ is a deadlock state of $p_{\mathcal{A}}(S_E/G)$.

Proof: Let $q \in Q$ be a deadlock state of S_E/G, i.e $\delta_{S_E/G}(q) = \emptyset$. According to definition of S_E/G, we obtain that $\delta(q) \setminus S_E(q) = \emptyset$ which means that

$$\delta(q) \subseteq S_E(q) \qquad (\alpha)$$

Therefore by definition again,

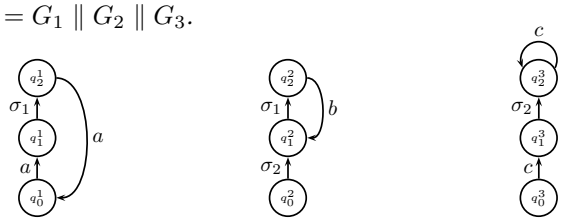
$$\delta_{p_{\mathcal{A}}(S_E/G)}(p_{\mathcal{A}}(q)) = \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q)) \setminus p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$$

If $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q)) = \emptyset$, then the result holds. otherwise, we consider $\sigma \in \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q))$. In this case, $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)!$ and from lemma 1 we obtain $\delta(q,\sigma)!$ and finally $\sigma \in \delta(q)$. Hence, from (α), $\sigma \in S_E(q)$. From lemma 2 (we have $p_{\mathcal{A}}(\Sigma) \cap \delta(q)$), we obtain that $\sigma \in p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$. Finally we can deduce that, $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q)) \subseteq p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$. Therefore, $\delta_{p_{\mathcal{A}}(S_E/G)}(p_{\mathcal{A}}(q)) = \emptyset$. ■

The intuitive meaning of Proposition 4 is the following: if you consider the projection of a deadlock state of $\mathcal{S}/G$ over a subset of the components, then this projected state is also in deadlock in the projected controlled system. This result allows to detect deadlock states of S/G in an incremental manner. At each step, previous computations can be used, reducing the state space in which the deadlocks are.

To begin the detection of deadlock states, let us first consider the set D_1 of states that are in deadlock in $p_{\mathcal{A}}(S_E/G)$, with $\mathcal{A} = \{1\}$ and that belong to $p_{\mathcal{A}}(PB(G))$. Further, we add a new component, say G_2 and we detect the deadlock states on $p_{\{1,2\}}(S_E/G)$ knowing that the deadlock states are in $D_1 \times Q_2$. We thus incrementally add the components until we obtain the whole controlled system. Note that, at each iteration of this procedure, we can make the use of propositions 2 and 3 to restrict the state space in which we have to find deadlock states.

Example 3: Let us now illustrate the proposed method in this paper, combining the various results previously given. To that aim, let us consider the simple concurrent system given by figure 2. This system G is composed of three subsystems: $G = G_1 \parallel G_2 \parallel G_3$.



(a) G_1 (b) G_2 (c) G_3

Fig. 2. A controlled concurrent system containing deadlock states

The alphabet of each subsystem is given by $\Sigma_1 = \{a, \sigma_1\}$, $\Sigma_2 = \{b, \sigma_1, \sigma_2\}$ and $\Sigma_3 = \{c, \sigma_2\}$. For simplicity, there is no uncontrollable event in the system. We also have that $\Sigma_s = \{\sigma_1, \sigma_2\}$. Moreover, we suppose here that the product sets $E^1 = \{q_2^1\} \times \{q_2^2\} \times \{q_1^3\}$ and $E^2 = \{q_1^1\} \times \{q_0^2\} \times \{q_0^3, q_1^3\}$ are forbidden. We denote $E = E^1 \cup E^2$ the set of forbidden states, and S_E denotes the supervisor ensuring the non reachability of E in G, as described by (4). Our goal is now to detect the deadlock states of this controlled system. To that aim, we consider the set PB(G) of possible deadlock in G

$$PB(G) = \{q_1^1\} \times \{q_0^2, q_1^2\} \times \{q_1^3\}$$

To apply incremental computation, we first consider the projection of S_E/G over $\{1\}$. $p_{\{1\}}(G)$ is given by figure 3 (a), and $p_{\{1\}}(S_E)$ ensures the avoidance of $p_{\{1\}}(E)$ over

$p_{\{1\}}(G)$. $p_{\{1\}}(E)$ corresponds to the union of $\{q_2^1\}$ and $\{q_1^1\}$. Applying propositions 2, 3 and 4, we look for states of $p_{\{1\}}(S_E/G)$ which are in deadlock and either are forbidden or belong to PB_1. Since $\delta_{p_{\{1\}}(G)}(q_0^1, a) = q_1^1$ which is a forbidden state, we actually have that $\delta_{p_{\{1\}}(S_E/G)}(q_0^1) = \emptyset$. As mentioned above, q_0^1 is in deadlock in $p_{\{1\}}(S_E/G)$, but it is not forbidden and does not belong to PB_1. Hence the only interesting state is q_1^1 which is in deadlock and forbidden (and in PB_1). We note $D_1 = \{q_1^1\}$.

Then, in a second part, we add one component and consider the system $p_{\{1,2\}}(S_E/G)$. $p_{\{1,2\}}(G)$ is given by figure 3 (b). $p_{\{1,2\}}(E)$ is now given by the union of $\{q_2^1\} \times \{q_2^2\}$ and $\{q_1^1\} \times \{q_0^2\}$, and $p_{\{1,2\}}(S_E)$ ensures the avoidance of $p_{\{1,2\}}(E)$ over $p_{\{1,2\}}(G)$. Applying results of propositions 2, 3 and 4 again, we now look for states of $p_{\{1,2\}}(S_E/G)$ which are in deadlock and either are forbidden or belong to $PB_1 \times PB_2$. Moreover, thanks to proposition 4, it is sufficient to restrict our search to the states for which the first component belongs to D_1. The considered system is quite simple here, but such an incremental method can help in significantly saving computations (in our example, D_1 only contains one state). At this step, two states are considered: (q_1^1, q_0^2) which is forbidden and in deadlock in $p_{\{1,2\}}(S_E/G)$, and (q_1^1, q_1^2) which is in deadlock and belongs to $PB_1 \times PB_2$. We denote $D_2 = \{(q_1^1, q_0^2), (q_1^1, q_1^2)\}$.

Finally, at the last step, we consider S_E/G, and the purpose is to detect deadlock states in this system. Likely, thanks to the previous computations, not all states of S_E/G need to be considered, but only the ones for which the projection over $\{1, 2\}$ belongs to the set D_2. Finally (q_1^1, q_0^2, q_1^3) and (q_1^1, q_1^2, q_1^3) are detected to be deadlock states in S_E/G.

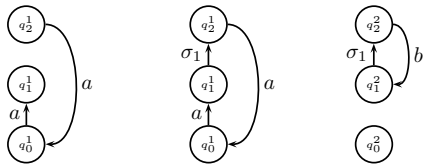
Actually, to ensure the fact that there is no more deadlock states in this system, some computations still need to be performed. Indeed, the computations made above give access to the deadlock states of S_E/G from which either only shared events are prevented from occurring, or at least one local event of Σ_3 is prevented from occurring. (q_1^1, q_0^2, q_1^3) is of the first type and (q_1^1, q_1^2, q_1^3) is of the second one.

To consider other cases, deadlock states of S_E/G from which at least one local event of Σ_1 is disabled, should also be detected too. For that purpose one can apply the same method: computation of the set D_1' of deadlock and forbidden states of $p_2(S_E/G)$. Then computation of the set D_2' of deadlock and forbidden states of $p_{2,3}(S_E/G)$ for which the component over G_2 belongs to D_1'. And Finally, computation of deadlock states of S_E/G for which the projection over $\{2, 3\}$ belongs to D_2'. Finally, the same method should be applied again to detect deadlock states of S_E/G from which at least one local event of Σ_2 is disabled.

Finally, in this example, performing all the computations mentioned above gives us the following set of deadlock states: $\{(q_1^1, q_0^2, q_1^3), (q_1^1, q_1^2, q_1^3)\}$.

VI. Conclusion

In this paper, the non-blocking State Avoidance Control Problem over concurrent discrete event systems is considered. We first outline a method allowing to efficiently compute a supervisor ensuring the avoidance of a set of states E (see [10] for details). Regarding the Deadlock Avoidance Control Problem, we first provide some characterizations of deadlock states in such systems, using the structure of both the system and the supervisor. In a second time, an incremental method, also based on the structure of both the system and the supervisor is given. This method uses notion of projection of concurrent systems, supervisors, and states. Moreover, it can be combined with the results that help to characterize deadlock states, in order to improve the efficiency of their detection. At this step, two ways seem relevant to complete this work. We first want to apply for the control of large scale systems, to underline the practical interest of our method. In particular, we would be interested in considering examples classified as "hard" in [9]. We also want to extend these kind of results to livelock detection. Indeed, since blocking state of a system is either a deadlock or a livelock state, this could allow to ensure the classical non blocking using in supervisory control.



(a) $p_{\{1\}}(G_1)$ (b) $p_{\{1,2\}}(G_1 \parallel G_2)$

Fig. 3. Projections of the system G

References

[1] W. M. Wonham, "Notes on control of discrete-event systems," Department of ECE, University of Toronto, Tech. Rep. ECE 1636F/1637S, July 2003.

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

[3] M. deQueiroz and J. Cury, "Modular supervisory control of large scale discrete-event systems," in *Discrete Event Systems: Analysis and Control. Proc. WODES'00*, pp. 103–110. 2000.

[4] K. Akesson, H. Flordal, and M. Fabian, "Exploiting modularity for synthesis and verification of supervisors," in *Proc. of the IFAC*, 2002.

[5] Y. Willner and M. Heymann, "Supervisory control of concurrent discrete-event systems," *International Journal of Control*, vol. 54, no. 5, pp. 1143–1169, 1991.

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

[7] 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 2004.

[8] R. Minhas, "Complexity reduction in discrete event systems," Ph.D. dissertation, Univeristy of Toronto, September 2002.

[9] A. Vahidi, B. Lennarston, and M. Fabian, "Efficient supervisory synthesis of large systems," in *Workshop on Discrete Event Systems, WODES'04*, September 2004.

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

[11] A. Abdelwahed and W. Wonham, "Blocking detection in discrete event systems," in *Proc. of 2003 American Control Conference*, Denvers, Colorado USA, June 2003.

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