Discrete Event Dynamic Systems manuscript No.
(will be inserted by the editor)

Decentralized Control of Infinite Systems

Gabriel Kalyon · Tristan Le Gall · Hervé Marchand · Thierry Massart

Received: date / Accepted: date

Abstract We propose algorithms for the synthesis of decentralized state-feedback controllers with partial observation of infinite state systems, which are modeled by Symbolic Transition Systems. We first consider the computation of safe controllers ensuring the avoidance of a set of forbidden states and then extend this result to the deadlock free case. The termination of the algorithms solving these problems is ensured by the use of abstract interpretation techniques, but at the price of overapproximations, in particular, in the computation of the states which must be avoided. We then extend our algorithms to the case where the system to be controlled is given by a collection of subsystems (modules). This structure is exploited to locally compute a controller for each module. Our tool SMACS gives an empirical evaluation of our methods by showing their feasibility, usability and efficiency.

Keywords Symbolic Transition Systems · Decentralized Controller Synthesis · Partial Observation · Abstract Interpretation · Controller Synthesis of Modular Systems.

G. Kalyon · T. Le Gall · T. Massart
Université Libre de Bruxelles (U.L.B.), Campus de la Plaine, Bruxelles, Belgique
Tel.: +3226505614
Fax: +3226505609
E-mail: {gkalyon,tlegall,tmassart}@ulb.ac.be

H. Marchand
INRIA, Centre Rennes - Bretagne Atlantique
Tel.: +33299847509
Fax: +33299847171
E-mail: herve.marchand@inria.fr

G. Kalyon is supported by the Belgian National Science Foundation (FNRS) under a FRIA grant.
This work has been done in the MoVES project (P6/39) which is part of the IAP-Phase VI Interuniversity Attraction Poles Programme funded by the Belgian State, Belgian Science Policy.

1 Introduction

We are interested in the *state avoidance control problem*, as defined in Kumar and Garg (2005), in the domain, introduced by Ramadge and Wonham (1989), of controller synthesis for Discrete Events Systems. The aim is to synthesize controllers which prevent the system from reaching any element of a specified set of forbidden states.

When modelling realistic systems, it is often convenient to manipulate state variables instead of simply atomic states, allowing a compact way to specify systems handling data. Within this framework, a state of the underlying state machine can be seen as a particular instantiation of vector of variables. If the domain of the variables is infinite, the semantics of such a system is therefore given by a potentially infinite state labeled transition system where the states are valuations of the variables. For example, one can consider timed automata, which extend finite transition with clocks and allow transitions upon thresholds and reset operations. Other examples of infinite systems with variables are the general Petri nets or the Vector Discrete Event Systems. In this paper, we model the system to be controlled by *Symbolic Transition Systems* (STS) which encompass the two previous classes of systems. STS is a model of (infinite) systems defined over a set of variables, whose domain can be infinite, and composed of a finite set of *symbolic transitions*. Each transition has a guard, which indicates in which condition it can be fired, and an *update* function which indicates the evolution of the variables when the transition is fired. Furthermore, transitions are labeled with *actions* taken from a finite alphabet[1].

From a control point of view, the controller interacts with the plant through sensors and actuators, and in general it has only a *partial observation* of the system, since these elements may work with some imprecision or some parts of the plant are not observed. Since control specifications are defined on the system states, it is more natural and more useful to consider a controller observing the system through its states as done in Wonham and Ramadge (1988). We also follow and extend the approach taken by Kumar et al. (1993), where the partial observation is modeled by a *mask* corresponding to a mapping from the state space to a possibly *infinite* observation space. Note that an important topic is the quality of the synthesized controller: it can be measured by *permissiveness* criteria, which state, for example in Takai and Kodama (1997), that the set of transitions allowed by the controller must be maximal.

In this paper, we consider a *decentralized framework* where n controllers interact with the system. The key elements of this approach are the following: (*i*) each controller has its own partial view of the system, (*ii*) each controller can control only a part of the system, and (*iii*) a *fusion rule*, depending on the control decisions of each controller, defines the global control, which must be applied to the system. The decentralized control aims at the supervision of systems where having a single centralized controller seems unrealistic, e.g.

[1] Note that in Jeannet et al. (2005), this alphabet is assumed to be infinite.

in the case of distributed systems. The decentralized control framework has been widely studied in the past years and has shown its usefulness in several application domains like manufacturing systems, communication networks protocol, etc (see e.g. Rudie and Wonham (1992a); Takai (1998); Yoo and Lafortune (2000); Rudie and Wonham (1992b)). Moreover, if the structure is known and can be decomposed into several subsystems acting in parallel, it is of interest to compute the set of controllers ensuring the expected properties without having to compute the whole system, which could lead to the classical state space explosion due to the parallel composition of the subsystems. This might also have an impact when performing some fixpoint computations that are necessary to calculate the set of states that has to be avoided by control. In the sequel, we shall refer to this particular control problem as a *modular control problem* as opposed to the decentralized control problem for which the structure of the system is not taken into account when performing the computation of the controller.

State of the art. The control synthesis of *finite systems with partial observation on the actions* has been widely studied in the literature (see Cassandras and Lafortune (2008) for an outline of the results). The synthesis of a *centralized controller* for finite systems with *partial observation on the states* has been introduced in Kumar et al. (1993) using the notion of *mask*, which provides a partition of the state space. Takai and Kodama (1998) define the concept of *M-controllability* and give a necessary and sufficient condition based on this notion to decide the existence of a controller whose resulting controlled system can reach exactly a set Q of allowed states. The synthesis of centralized controllers *for infinite state systems with perfect observation* has been studied by Kumar and Garg (2005). They prove that, in this case, the state avoidance control problem is undecidable. They also show that the problem can be solved for the particular case of Petri nets when the set of forbidden states is upward closed. Le Gall et al. (2005) use symbolic techniques to control, in a centralized framework, infinite systems modeled by STS. They also use abstract interpretation techniques (e.g. Cousot and Cousot (1977); Jeannet (2003)) to ensure that the computation of the controller always terminates. These techniques have been extended by the authors in Kalyon et al. (2009) where partial observation is also handled. The synthesis of *decentralized controllers for finite state systems with partial observation on the states* has been studied by Takai et al. (1994). In this work, each local controller has its own observation of the system and the aim is to satisfy a global specification Q of control, where Q is the set of allowed states. The authors define the notion of *n-observability* and present a necessary and sufficient condition to decide the existence of decentralized controllers such that the set of reachable states in the resulting controlled system is Q.

We extend here the results presented in Kalyon et al. (2009) to achieve synthesis of decentralized controllers for infinite state systems with partial observation on the states. To handle infinite state spaces, the algorithms provided are *symbolic* i.e., they do not enumerate the states, but use symbolic predicate

transformers on the variables of the system. We use abstract interpretation techniques to obtain effective algorithms (i.e., which always terminate). Of course, since the state avoidance control problem for infinite systems is undecidable, our algorithms may not be optimal. Note that the *abstract domain* that we have used in the abstract interpretation can be infinite. Our algorithms have been implemented in our tool SMACS[2], which allowed us to provide empirical evaluations of our approach and show its efficiency in processing time.

In section 2, we introduce our model of STS. In section 3, we define the control mechanisms used and the state avoidance decentralized control problem. In section 4, we first present procedures to solve our problem; these are not algorithms since they may not terminate. Then, we explain how to obtain effective algorithms by the use of abstract interpretation techniques and after, we compare the centralized control versus the decentralized one. In section 5 we consider the modular case where the system to be controlled is given by a collection of subsystems. In section 6 gives an empirical evaluation of our method.

2 Symbolic Transition Systems

The model of Symbolic Transition Systems (STS) is a transition system with variables, whose domain can be infinite, and is composed of *symbolic transitions*. Each transition has a *guard* on the variables of the system and an *update function* which indicates the evolution of the variables when the transition is fired. Furthermore, transitions are labeled with symbols taken from a finite alphabet. This model allows the representation of infinite systems whenever the variables take their values in an infinite domain. It has a finite structure and offers a compact way to specify systems handling data.

Variables, Predicates, Assignments. In the sequel, we assume to have a k-tuple $V = \langle v_1, \ldots, v_k \rangle$ (k is constant) of variables that belong to different domains; the (infinite) domain of a variable v is denoted by $\mathcal{D}_v$. $\mathcal{D}_V$ denotes $\prod_{i\in[1,k]} \mathcal{D}_{v_i}$. A *valuation* $\boldsymbol{\nu}$ of V is a k-tuple $\langle \boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_k \rangle \in \mathcal{D}_V$ and represents a possible assignment of values for the variables. Let V be a set of variables, a *predicate* $P : \mathcal{D}_V \mapsto \{\mathsf{true}, \mathsf{false}\}$ over the set $\mathcal{D}_V$ is a function, which associates to each element $\boldsymbol{\nu} \in \mathcal{D}_V$ a *truth value* $P(\boldsymbol{\nu})$. $P(\boldsymbol{\nu}) = \mathsf{true}$ and $P(\boldsymbol{\nu}) = \mathsf{false}$ are respectively denoted $P(\boldsymbol{\nu})$ and $\neg P(\boldsymbol{\nu})$. The predicate $P : \mathcal{D}_V \mapsto \{\mathsf{true}, \mathsf{false}\}$ can also be viewed as a subset $Y \subseteq \mathcal{D}_V$ defined by $Y \triangleq \{\boldsymbol{\nu} \in \mathcal{D}_V \mid P(\boldsymbol{\nu})\}$. The *complement* of a set $H \subseteq \mathcal{D}_V$ is denoted by $\overline{H}$. The *preimage function* of $f : D_1 \mapsto D_2$ is denoted by $f^{-1} : D_2 \mapsto 2^{D_1}$ and is defined, for all $d_2 \in D_2$, by $f^{-1}(d_2) = \{d_1 \in D_1 | f(d_1) = d_2\}$. Finally, we naturally extend a function $f : D_1 \mapsto D_2$ to sets $H \subseteq D_1$ as follows: $f(H) = \bigcup_{h\in H} f(h)$.

Model of the System. Our systems to be controlled are modeled by STS defined as follows:

[2] See *http://www.smacs.be*

Definition 1 (Symbolic Transition System) A *symbolic transition system* (STS) is a tuple $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ where:

- $V = \langle v_1, \ldots, v_k \rangle$ is a k-tuple of variables (k is constant)
- $\Theta \subseteq \mathcal{D}_V$ is a predicate on V, which defines the initial condition on the variables
- Σ is a finite alphabet of actions
- Δ is a finite set of symbolic transitions $\delta = \langle \sigma_\delta, G_\delta, A_\delta \rangle$ where (i) $\sigma_\delta \in \Sigma$ is the action of δ, (ii) $G_\delta \subseteq \mathcal{D}_V$ is a predicate on V, which defines the guard of δ, and (iii) $A_\delta : \mathcal{D}_V \mapsto \mathcal{D}_V$ is the update function of δ. •

The semantics of an STS is a possibly infinite Labeled Transition System (LTS) whose states are valuations of the variables:

Definition 2 (STS's Semantics) The semantics of an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ is an LTS $[\![\mathcal{T}]\!] = \langle X, X_0, \Sigma, \rightarrow \rangle$ where (i) $X = \mathcal{D}_V$ is the set of states, (ii) $X_0 = \Theta$ is the set of initial states, (iii) Σ is the set of labels, and (iv) the transition relation $\rightarrow \subseteq X \times \Sigma \times X$ is defined by $\{\langle \boldsymbol{\nu}, \sigma, \boldsymbol{\nu}' \rangle \mid \exists \langle \sigma, G, A \rangle \in \Delta : (\boldsymbol{\nu} \in G) \wedge (\boldsymbol{\nu}' = A(\boldsymbol{\nu}))\}$. •

Initially, an STS is in one of its initial states. In each state, a transition can be fired only if its guard is satisfied. When it is fired, the variables are updated according to the update function. A state $\boldsymbol{\nu} \in \mathcal{D}_V$ is in *deadlock* if no transition can be fired from this state i.e., $\forall \delta \in \Delta : \boldsymbol{\nu} \notin G_\delta$. Note that the LTS $[\![\mathcal{T}]\!]$ can be non-deterministic.

Remark 1 *The formalism presented in Definition 1 does not allow to define* explicitly *locations. But, this model is in fact equivalent to a model with explicit locations, because a variable $Loc = \{\ell_1, \cdots \ell_n\}$ of enumerated type can always be added to encode the locations. In our examples, the figures representing STSs will have, to be clearer, explicit locations that will be encoded by this mechanism.* ◇

Example 1 (Producer and consumer) *The STS of Fig. 1 is a system of stock management. This example will be used throughout this paper to illustrate the concepts and the methods presented. Two units produce and send (consume) two kinds of pieces X and X'. The notations* Id, $\top$ *and* $\bot$ *define respectively the identity function and the predicates true and false.*

The STS has explicit locations $\ell \in \{\mathsf{CX}, \mathsf{PX}, \mathsf{Choice}, \mathsf{CX}', \mathsf{PX}'\}$ and four variables in $\mathbb{N}$, the set of natural numbers: x (resp. x') gives the current number of pieces X (resp. X') and y (resp. y') gives the number of pieces X (resp. X') that can be produced. A state of the system corresponds to a 5-tuple $\langle \ell, x, x', y, y' \rangle$ and, in this example, the initial state is chosen to be $\langle \mathsf{Choice}, 50, 50, 0, 0 \rangle$. The choice of the kind of pieces to produce is performed in the location Choice*: the action Choice_X (resp. Choice_X') allows the system to choose the production of pieces of type X (resp. X')[3]. In the location*

[3] For convenience, in the guards and update functions of the transitions of the system, we omit the conditions and assignments related to the locations. For example, the transition δ_9

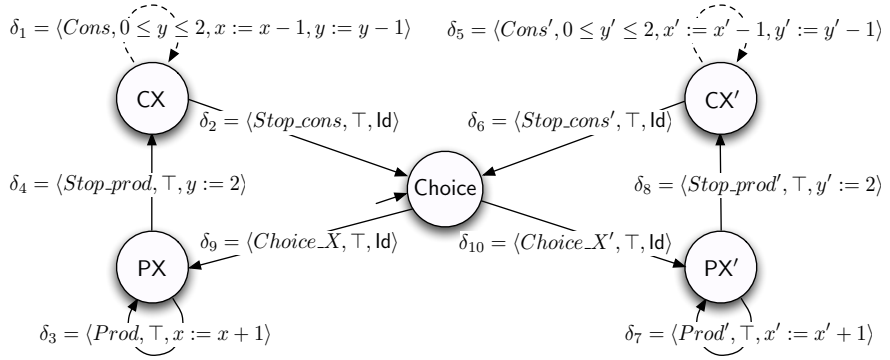


Fig. 1 Example of producer and consumer.

PX *(resp.* PX′*), the action Prod (resp. Prod′) produces a piece of kind X (resp. X′) whereas the action Stop_prod (resp. Stop_prod′) stops the process of production. In the location* CX *(resp.* CX′*), the action Cons (resp. Cons′) consumes a piece X (resp. X′) whereas the action Stop_cons (resp. Stop_cons′) stops the process of consumption. The variables y and y′ ensure that at most two pieces can be consumed in each cycle of consumption.* ◇

A Predicate Transformer is a function from $\mathcal{D}_V$ to $\mathcal{D}_V$ that given a set of states (or a predicate) computes a new set of states (or a new predicate). In the sequel, we shall use the following notations, for all STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta\rangle$, $\delta = \langle \sigma_\delta, G_\delta, A_\delta\rangle \in \Delta$, $\Delta' \subseteq \Delta$, $\Sigma' \subseteq \Sigma$ and set of states $B \subseteq \mathcal{D}_V$:

- $\mathrm{Trans}(\sigma) = \{\delta \in \Delta \mid \sigma_\delta = \sigma\}$ is the set of transitions labeled by the action σ,
- $\mathsf{Pre}^{\mathcal{T}}_{\delta}(B) = G_\delta \cap A_\delta^{-1}(B) = \{\boldsymbol{\nu} \in \mathcal{D}_V \mid \exists \boldsymbol{\nu}' \in B : (\boldsymbol{\nu} \in G_\delta) \wedge (\boldsymbol{\nu}' = A_\delta(\boldsymbol{\nu}))\}$ is the set of states leading to B through the transition δ,
- $\mathsf{Pre}^{\mathcal{T}}_{\Delta'}(B) = \bigcup_{\delta\in\Delta'} \mathsf{Pre}^{\mathcal{T}}_{\delta}(B)$ is the set of states leading to B through a transition in Δ',
- $\mathsf{Pre}^{\mathcal{T}}_{\Sigma'}(B) = \bigcup_{\sigma\in\Sigma'} \mathsf{Pre}^{\mathcal{T}}_{\mathrm{Trans}(\sigma)}(B)$ is the set of states leading to B through a transition labeled by an action in Σ',
- $\mathsf{Post}^{\mathcal{T}}_{\delta}(B) = A_\delta(G_\delta \cap B)$ is the set of states that are reachable from B through the transition δ,
- $\mathsf{Post}^{\mathcal{T}}_{\Delta'}(B) = \bigcup_{\delta\in\Delta'} \mathsf{Post}^{\mathcal{T}}_{\delta}(B)$ is the set of states that are reachable from B through a transition in Δ',
- $\mathsf{Post}^{\mathcal{T}}_{\Sigma'}(B) = \bigcup_{\sigma\in A} \mathsf{Post}^{\mathcal{T}}_{\mathrm{Trans}(\sigma)}(B)$ is the set of states that are reachable from B through a transition labeled by an action in Σ', and
- $\mathsf{Coreach}^{\mathcal{T}}_{\Sigma'}(B) = \mathrm{lfp}(\lambda B'.B \cup \mathsf{Pre}^{\mathcal{T}}_{\Sigma'}(B'))$ is the set of states leading to B by triggering only events in Σ' (lfp denotes the least fixpoint).

In the remainder of this paper, we work with sets of states and use operations on these sets. In our tool SMACS, the sets of states are symbolically repre-

is defined by $\langle Choice\_X, \mathsf{T}, \mathsf{Id}\rangle$, whereas it should be defined by $\langle Choice\_X, l = \mathsf{Choice}; l := PX\rangle$.

sented by predicates and each operation on sets corresponds to a predicate transformer. For example, $\mathsf{Pre}_\delta^{\mathcal{T}}(B)$ is given by the set of states $\boldsymbol{\nu}$ which satisfy the predicate transformer $\exists \boldsymbol{\nu}' \in B : (\boldsymbol{\nu} \in G_\delta) \wedge (\boldsymbol{\nu}' = A_\delta(\boldsymbol{\nu}))$. Further details can be found in Le Gall et al. (2005); Jeannet et al. (2005).

Parallel Composition of STS. Frequently, a system is initially given by a collection of (simpler) components modeled by STS that interact with each other by synchronizing on common events. The global behavior of this system is then obtained by composing these STS together using the parallel composition operator that represents the concurrent behavior of the STS with synchronization on the common events. It is defined as follows:

Definition 3 (Parallel Composition) Let $\mathcal{T}_1 = \langle V_1, \Theta_1, \Sigma_1, \Delta_1 \rangle$ and $\mathcal{T}_2 = \langle V_2, \Theta_2, \Sigma_2, \Delta_2 \rangle$ be STS such that $V_1 \cap V_2 = \emptyset$ and we defined the set of shared events $\Sigma_s = \Sigma_1 \cap \Sigma_2$. The parallel composition $\mathcal{T}_1 || \mathcal{T}_2$ of $\mathcal{T}_1$ and $\mathcal{T}_2$ is given by the STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ where (i) $V = V_1 \cup V_2$, (ii) $\Theta = \Theta_1 \times \Theta_2$, (iii) $\Sigma = \Sigma_1 \cup \Sigma_2$, and (iv) Δ is defined as follows for each $\sigma \in \Sigma$:

- if $\sigma \in \Sigma_1 \setminus \Sigma_s$, then for each $\delta_1 = \langle \sigma, G_1, A_1 \rangle \in \Delta_1$ the transition $\delta = \langle \sigma, G_1, A \rangle \in \Delta$, where the update function $A : \mathcal{D}_{V_1} \times \mathcal{D}_{V_2} \mapsto \mathcal{D}_{V_1} \times \mathcal{D}_{V_2}$ is defined, for each $\boldsymbol{\nu}_1 \in \mathcal{D}_{V_1}$ and $\boldsymbol{\nu}_2 \in \mathcal{D}_{V_2}$, by $A(\langle \boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \rangle) = \langle A_1(\boldsymbol{\nu}_1), \boldsymbol{\nu}_2 \rangle$.
- if $\sigma \in \Sigma_2 \setminus \Sigma_s$, then for each $\delta_2 = \langle \sigma, G_2, A_2 \rangle \in \Delta_2$ the transition $\delta = \langle \sigma, G_2, A \rangle \in \Delta$, where the update function $A : \mathcal{D}_{V_1} \times \mathcal{D}_{V_2} \mapsto \mathcal{D}_{V_1} \times \mathcal{D}_{V_2}$ is defined, for each $\boldsymbol{\nu}_1 \in \mathcal{D}_{V_1}$ and $\boldsymbol{\nu}_2 \in \mathcal{D}_{V_2}$, by $A(\langle \boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \rangle) = \langle \boldsymbol{\nu}_1, A_2(\boldsymbol{\nu}_2) \rangle$.
- if $\sigma \in \Sigma_s$, then for each $\delta_1 = \langle \sigma, G_1, A_1 \rangle \in \Delta_1$ and $\delta_2 = \langle \sigma, G_2, A_2 \rangle \in \Delta_2$ the transition $\delta = \langle \sigma, G_1 \cap G_2, A \rangle \in \Delta$, where the update function $A : \mathcal{D}_{V_1} \times \mathcal{D}_{V_2} \mapsto \mathcal{D}_{V_1} \times \mathcal{D}_{V_2}$ is defined, for each $\boldsymbol{\nu}_1 \in \mathcal{D}_{V_1}$ and $\boldsymbol{\nu}_2 \in \mathcal{D}_{V_2}$, by $A(\langle \boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \rangle) = \langle A_1(\boldsymbol{\nu}_1), A_2(\boldsymbol{\nu}_2) \rangle$. •

3 The State Avoidance Decentralized Control Problem

In this section, we define the state avoidance control problem w.r.t. the available information from the observation of the system and the available control mechanisms. The decentralized control of discrete event systems theory is a natural approach to decrease the computational complexity of synthesizing controllers for large scale systems: the overall task of the synthesis is divided into smaller tasks of synthesizing local controllers, each of them reacting according to a (different) partial observation of the system.

3.1 Means of observation and control

Definition 4 (Observer) An observer of the state space $\mathcal{D}_V$ is a pair $\langle Obs, M \rangle$, where (i) Obs is a variable, whose domain is the (possibly infi-

nite) observation space $\mathcal{D}_{Obs}$[4], and (*ii*) the mask $M : \mathcal{D}_V \mapsto \mathcal{D}_{Obs}$ is a total function which gives, for each state $\boldsymbol{\nu} \in \mathcal{D}_V$, the observation $M(\boldsymbol{\nu})$ that the controller has when the system enters this state. •

Example 2 *For the system of Fig. 1, partial observation can be given by the mask* $M : Loc \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \mapsto Loc \times \mathbb{N} \times \mathbb{N}$*, where for each state* $\langle \ell, x, x', y, y' \rangle \in \mathcal{D}_V$*, the value* $M(\langle \ell, x, x', y, y' \rangle) = \langle \ell, x, y \rangle$*. It means that the variables related to the pieces* X' *are not visible. In that case,* $\mathcal{D}_{Obs} = Loc \times \mathbb{N} \times \mathbb{N}$ *and the variable Obs is a 3-tuple* $\langle L, X, Y \rangle$ *where the domain of* L *is Loc and the domain of* X, Y *is* $\mathbb{N}$*.* ◇

For each observation $obs \in \mathcal{D}_{Obs}$, $M^{-1}(obs)$ gives the set of states, whose observation is obs. Note that the mask M is a *partition* of the state space, which means that $\forall obs, obs' \in \mathcal{D}_{Obs} : obs \neq obs' \Rightarrow M^{-1}(obs) \cap M^{-1}(obs') = \emptyset$.

In the decentralized framework, the control is performed by a set of controllers $\mathcal{C}_i$ ($\forall i \in [1, n]$) which interact with the system in a feedback manner. Each controller $\mathcal{C}_i$ has its own partial view of the state space of the system $\mathcal{T}$. Thus in the sequel, we assume that to each controller $\mathcal{C}_i$ is associated an observer $\langle Obs_i, M_i \rangle$. Given the current state $\boldsymbol{\nu}$ of the system, the control decision of each controller $\mathcal{C}_i$ (i.e., the set of actions that, from its observation, it cannot allow) is thus based on its own local observations of the system $M_i(\boldsymbol{\nu}) \in \mathcal{D}_{Obs_i}$.

The control is performed by means of *controllable events*: for each controller $\mathcal{C}_i$, the alphabet Σ classically is partitioned into the set of controllable events $\Sigma_{i,c}$, that the controller $\mathcal{C}_i$ can decide not to allow, and the set of uncontrollable events $\Sigma_{i,uc}$; this also induces a partitioning of the set of symbolic transitions Δ between the set of controllable transitions $\Delta_{i,c}$ and the set of uncontrollable ones $\Delta_{i,uc}$, where $\delta = (\sigma, G, A) \in \Delta_{i,c}$ (resp. $\Delta_{i,uc}$) if $\sigma \in \Sigma_{i,c}$ (resp. $\Sigma_{i,uc}$). Note that the subsets $\Sigma_{1,c}, \ldots, \Sigma_{n,c}$ are not necessarily disjoint. The set $\Sigma_c = \bigcup_{i=1}^{n} \Sigma_{i,c}$ denotes the set of actions that can be controlled by at least one controller and the set $\Sigma_{uc} = \Sigma \setminus \Sigma_c$ denotes the set of actions that cannot be controlled. We will see (Def. 6) that an action σ is globally forbidden if each controller, which can forbid it, decides so. Moreover, $\mathrm{In}(\sigma) = \{i \mid \sigma \in \Sigma_{i,c}\}$ denotes the set of indices of the controllers which can control σ.

3.2 Controllers and controlled system

The aim of the controllers is to restrict the behavior of the system in order to ensure a forbidden state invariance property (i.e., they must prevent the system from reaching forbidden states). First we recall the definition of a centralized controller with partial observation and then extend its definition to the decentralised case.

[4] To remain coherent with the formalization of the state space $\mathcal{D}_V$, we have chosen to define the observation space $\mathcal{D}_{Obs}$ by means of a variable *Obs* whose domain is $\mathcal{D}_{Obs}$. In particular, it allows us to use predicate transformers w.r.t. this variable.

Definition 5 (Controller) Given an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, a partition $\Sigma = \Sigma_{i,c} \cup \Sigma_{i,uc}$ and an observer $\langle Obs_i, M_i \rangle$, a controller for $\mathcal{T}$ is a pair $\mathcal{C}_i = \langle \mathcal{S}_i, E_i \rangle$, where:

1. $\mathcal{S}_i : \mathcal{D}_{Obs_i} \mapsto 2^{\Sigma_{i,c}}$ is a supervisory function which defines, for each observation $obs \in \mathcal{D}_{Obs_i}$, a set $\mathcal{S}_i(obs)$ of controllable actions to be forbidden when obs is observed by the controller.
2. $E_i \subseteq \mathcal{D}_V$ is a set of states to be forbidden, which restricts the set of initial states. •

To avoid repetitions, in the remaining part of the paper, we always work with a system $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ to be controlled, n observers $\langle Obs_i, M_i \rangle$ $(\forall i \in [1, n])$, and n controllers $\mathcal{C}_i = \langle \mathcal{S}_i, E_i \rangle$ $(\forall i \in [1, n])$.

The decentralized control mechanism is composed of n controllers $\mathcal{C}_i$ $(\forall i \in [1, n])$ interacting with the system and which can disable actions. The synchronization of these n controllers defines then the *global control* applied to the system. This control is in fact defined by fusion rules which give the actions and the initial states to be forbidden:

Definition 6 (Fusion rules) The fusion rules for the actions and the initial states to be forbidden are defined by :

1. Assume that the system is in state $\boldsymbol{\nu}$ and that each controller $\mathcal{C}_i$ observe $M_i(\boldsymbol{\nu}) \in \mathcal{D}_{Obs_i}$, then $B_i = \mathcal{S}_i(M_i(\boldsymbol{\nu})) \subseteq \Sigma_{i,c}$ $(\forall i \in [1, n])$ be the actions forbidden by the controller $\mathcal{C}_i$. The fusion rule $\mathcal{R}^{\mathcal{S}}$, which gives the actions to be forbidden *globally*, is defined by :

$$\mathcal{R}^{\mathcal{S}}(B_1, \ldots, B_n) = \{\sigma \,|\, \forall i \in \mathrm{In}(\sigma) : \sigma \in B_i\} \tag{1}$$

   An action σ is globally forbidden if each controller, which can forbid it, decides so.
2. Let $E_i \subseteq \mathcal{D}_V$ $(\forall i \in [1, n])$ be the states that the controller $\mathcal{C}_i$ decided to forbid at the beginning of the execution of the system. The fusion rule $\mathcal{R}^E$, which gives the initial states to be forbidden *globally*, is defined by :

$$\mathcal{R}^E(E_1, \ldots, E_n) = \bigcap_{i=1}^{n} E_i \tag{2}$$

•

Based on Definitions 5 and 6, a decentralized controller under a conjunctive architecture is defined as follows:

Definition 7 (Decentralized Controller) Given a system $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ to be controlled, n observers $\langle Obs_i, M_i \rangle$ $(\forall i \in [1, n])$, a *decentralized controller under a conjunctive architecture* is given by a tuple $\langle (\mathcal{C}_i)_{i=[1\ldots n]}, \mathcal{R}^{\mathcal{S}}, \mathcal{R}^E \rangle$, where

– $\mathcal{C}_i = \langle \mathcal{S}_i, E_i \rangle$ $(\forall i \in [1, n])$, is a controller defined according to Definition 5 w.r.t. the observer $\langle Obs_i, M_i \rangle$.

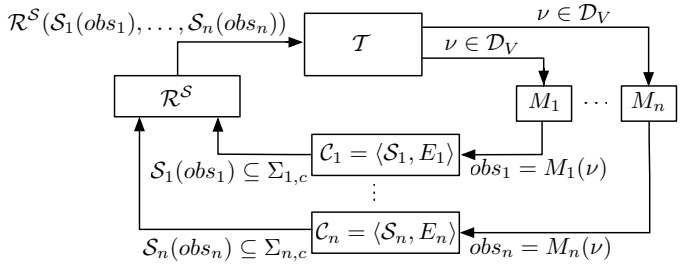


Fig. 2 Decentralized control under partial observation.

– $\mathcal{R}^{\mathcal{S}}, \mathcal{R}^{E}$ are the fusion rules defined according to Definition 6. •

The feedback interaction between the system $\mathcal{T}$ and a decentralized controller $\langle (\mathcal{C}_i)_{i=[1\ldots n]}, \mathcal{R}^{\mathcal{S}}, \mathcal{R}^{E} \rangle$ is depicted in Fig. 2 and can be summarized as follows. When the system is in a state $\boldsymbol{\nu}$, each controller $\mathcal{C}_i = \langle \mathcal{S}_i, E_i \rangle$ receives the observation $obs_i = M_i(\boldsymbol{\nu})$ and computes the actions $\mathcal{S}_i(obs_i)$ that it would be prudent to forbid, because a transition labeled by one of these actions could lead to *Bad* (the computation of $\mathcal{S}_i$ is defined in section 4). Then, the fusion rule $\mathcal{R}^{\mathcal{S}}$ (see (1)) gives the actions that the system cannot execute: an action σ is globally forbidden, if each controller, which can control σ, decides to forbid it. The controlled system resulting from this interaction is then defined as follows.

Definition 8 (Controlled System) The system $\mathcal{T}$ controlled by a decentralized controller under a conjunctive architecture $\langle (\mathcal{C}_i)_{i=[1\ldots n]}, \mathcal{R}^{\mathcal{S}}, \mathcal{R}^{E} \rangle$ is an STS $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}} = \langle V, \Theta_{/(\mathcal{C}_i)_{i=[1\ldots n]}}, \Sigma, \Delta_{/(\mathcal{C}_i)_{i=[1\ldots n]}} \rangle$, where:

– $\Theta_{/(\mathcal{C}_i)_{i=[1\ldots n]}} = \Theta \setminus \mathcal{R}^{E}(E_1,\ldots,E_n)$
– $\Delta_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$ is defined by the following rule: if $\langle \sigma, G, A \rangle \in \Delta$ then $\langle \sigma, G_{/(\mathcal{C}_i)_{i=[1\ldots n]}}, A \rangle \in \Delta_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$ with $G_{/(\mathcal{C}_i)_{i=[1\ldots n]}} = \{\boldsymbol{\nu} | \boldsymbol{\nu} \in G \wedge \sigma \notin \mathcal{R}^{\mathcal{S}}(\mathcal{S}_1(M_1(\boldsymbol{\nu})),\ldots,\mathcal{S}_n(M_n(\boldsymbol{\nu})))\}$. •

The supervisory functions $\mathcal{S}_i$ allow to restrict the guards of the controlled system. Indeed, a transition δ (labeled by σ) can no longer be fired from a state $\boldsymbol{\nu}$ in $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$, if the action σ is forbidden in this state by the global control.

For convenience, in the remainder of Section 3, a decentralized controller under a conjunctive architecture $\langle (\mathcal{C}_i)_{i=[1\ldots n]}, \mathcal{R}^{\mathcal{S}}, \mathcal{R}^{E} \rangle$ will be simply called a decentralized controller and will be denoted $(\mathcal{C}_i)_{i=[1\ldots n]}$ as the fusion rules of the conjunctive architecture are independent of the system and the controllers and simply reflect the chosen control architecture.

3.3 Discussions on our choices of observation and control means

We discuss here the choices of observation and control means we made in this paper, which are slightly different from a more classical approach of the decentralized control problem. The state-based observation approach is rather classical as taken by Takai and Kodama (1997, 1998); Kumar et al. (1993), but our definition of the fusion rules is not. Usually, one uses a disjunctive or conjunctive architecture as explained in Cassandras and Lafortune (2008) i.e., an event is globally enabled when all local supervisors enable it (conjunctive architecture) or at least one local supervisor enables it (disjunctive architecture). In our framework, an event is enabled whenever at least one supervisor, that can control this event, enables it.

In Takai et al. (1994), which is one of the few papers on decentralized state-feedback control of discrete event systems, the authors chose a conjunctive architecture. However, they want to obtain *balanced* controllers i.e., for each pair of states $\boldsymbol{\nu}, \boldsymbol{\nu}'$ reachable in the controlled system, if there is a transition σ which can be fired from $\boldsymbol{\nu}$ to $\boldsymbol{\nu}'$, then it must be enabled by each local controller. In other words, all local controllers make the same decision on shared events. The controllers we obtain are not balanced, and that is why a purely conjunctive architecture may not work in our framework. We will prove that, with our architecture, our computation of local controllers solves the basic and deadlock free state avoidance decentralized control problems.

Finally, from an implementation point of view, the reason why we prefer to do intersections rather than unions in the definition of the fusion rule is that to approximate sets of states, we make the use of convex polyhedra (see Section 4.3). When working with such elements, intersection is exact while "union" (convex hull) is not. Further, if we represent explicit union of convex polyhedra (instead of convex hull), the effective computation of the fixpoint becomes a lot more complex, since we will not have a clear widening operator nor a canonical representation of our sets of states.

3.4 Definition of the state avoidance decentralized control problems

We are interested in two distinct versions of the state avoidance decentralized control problem consisting of avoiding the system to reach some particular states, either because some properties are not satisfied in these states (e.g. the states where two states variables are equal or more generally the states in which some particular state predicates are satisfied) or because they are deadlocking states:

Problem 1 (Basic State Avoidance Decentralized Control Problem) *Given an STS* $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$*, n observers* $\langle Obs_i, M_i \rangle$ $(\forall i \in [1, n])$ *and a predicate Bad representing the set of forbidden states, the Basic State Avoidance Decentralized Control problem (BDP for short), consists in computing a decentralized controller* $(\mathcal{C}_i)_{i=[1\ldots n]}$ *such that* $\mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}) \cap Bad = \emptyset$.

In the sequel, *a valid decentralized controller* denote a decentralized controller $(\mathcal{C}_i)_{i=[1...n]}$ such that $\mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}_i)_{i=[1...n]}}) \cap Bad = \emptyset$. A solution to BDP does not ensure that the controlled system is deadlock free i.e., it does not ensure that the controlled system has always the possibility to make a move. To ensure this important property, we define a second problem:

Problem 2 (Deadlock Free State Avoidance Decentralized Control Problem) *Given an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, n observers $\langle Obs_i, M_i \rangle$ $(\forall i \in [1, n])$ and a predicate Bad, the Deadlock Free State Avoidance Decentralized Control Problem (DfDP for short) consists in computing a decentralized controller $(\mathcal{C}_i)_{i=[1...n]}$ such that (i) $\mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}_i)_{i=[1...n]}}) \cap Bad = \emptyset$ and (ii) $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1...n]}}$ does not contain reachable deadlocking states.*

We can immediately notice that a trivial correct decentralized controller $(\mathcal{C}_i)_{i=[1...n]}$ is the one where $E_i \supseteq \Theta$ for all $i \in [1, n]$ (i.e., where no state remains). Therefore, the notion of permissiveness has been introduced to compare the quality of different decentralized controllers for a given STS.

Definition 9 (Permissiveness) Given two valid decentralized controllers $(\mathcal{C}_i)_{i=[1...n]}$ and $(\mathcal{C}'_i)_{i=[1...n]}$ solving Problem 1 (resp. 2), $(\mathcal{C}_i)_{i=[1...n]}$ is more permissive than $(\mathcal{C}'_i)_{i=[1...n]}$ iff $\mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}_i)_{i=[1...n]}}) \supseteq \mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}'_i)_{i=[1...n]}})$. When the inclusion is strict, we say that the first one is strictly more permissive than the second one. •

In our settings, since the observations and the control of the system are based on (masked) states it is more appropriate to define the permissiveness w.r.t the states that are reachable in the controlled system, rather than w.r.t. the language of the actions that can be fired. Note that two controlled systems with the same reachable state space can have different enabled transitions[5]. We have proven in Kalyon et al. (2009) that a most permissive controller solving BDP or DfDP does not always exist; this result would have been the same for other kinds of permissiveness like language or execution inclusions. Consequently, we are looking for a maximal solution to BDP (resp. DfDP) i.e., an n-tuple of controllers $(\mathcal{C}_i)_{i=[1...n]}$ solving BDP (resp. DFDP) and such that there does not exist an n-tuple of controllers $(\mathcal{C}'_i)_{i=[1...n]}$ strictly more permissive, which also solves this problem. Unfortunately, computing a maximal solution to BDP or DfDP is undecidable as shown in Kalyon et al. (2009). In consequence, our aim is to find solutions of *good practical value* which are correct and as close as possible to a maximal solution to be of good practical value. Our experiments will empirically validate our solutions.

4 Symbolic Computation of the Controllers

In this section, we first present algorithms to synthesize decentralized controllers solving BDP and DfDP. Since the systems to be controlled are infinite,

[5] We could have used an extended definition of permissiveness where if two controlled systems have equal reachable state space, inclusion of the transitions that can be fired from reachable states is also taken into account.

these algorithms, where no approximation is done, do not ensure the termination of the computations. In section 4.3, we will explain how to modify them to obtain algorithms which always terminate. The idea of the control is the following. Given the set of forbidden states Bad, it might be the case that this set of states can be reached from a given state by triggering sequences of uncontrollable events. So, to prevent the set Bad to be reached, one has to remove these states from the reachable state space of the controlled system. So the general idea of the control is to compute, using a fixpoint computation, the set of states $I(Bad)$ that can lead to Bad or to deadlocking states in the controlled system (for the deadlock free case) triggering only uncontrollable transitions[6]. Then, based on this set of states, we compute the decentralized controller where each local component forbids, for each observation, the controllable transitions that could lead to $I(Bad)$.

4.1 Basic state avoidance decentralized control problem

We formalize the two steps which allow us to compute the decentralized controller $(\mathcal{C}_i)_{i=[1\ldots n]}$, whose synchronization by the fusion rules solves BDP.

Computation of $I(Bad)$. This set of states (and more generally the function $I(.)$) is given by the function $\mathsf{Coreach}^{\mathcal{T}}_{uc} : 2^{\mathcal{D}_V} \mapsto 2^{\mathcal{D}_V}$ defined in (3). This set corresponds to the set of states which lead to Bad firing only uncontrollable transitions (i.e., transitions which belong to Δ_{uc}). Classically, it is obtained by the following fixpoint equation:

$$\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad) = \mathrm{lfp}(\lambda B.Bad \cup \mathsf{Pre}^{\mathcal{T}}_{\Delta_{uc}}(B)) \tag{3}$$

where lfp denotes the least fixpoint and $\mathsf{Pre}^{\mathcal{T}}_{\Delta_{uc}}(B)$ is the function, which computes the set of states from which a state of B is reachable by triggering exactly one uncontrollable transition (see notations in section 2). By the Tarski's theorem given in Tarski (1955), since the function $\mathsf{Coreach}^{\mathcal{T}}_{uc}$ is monotonic, the limit of the fixpoint $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$ actually exists. But it may be uncomputable, because the coreachability is undecidable in the model of STS. Note that this function is used by all the n local controllers $\mathcal{C}_i$. In subsection 4.3, we explain how to compute an overapproximation of this fixpoint, while ensuring the termination of the computations.

Computation of the local controllers $\mathcal{C}_i$ $(\forall i \in [1, n])$ *and of the controlled system* $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$. We first define the function $\mathcal{F}^{\mathcal{T}}_i : \Sigma \times 2^{\mathcal{D}_V} \mapsto 2^{\mathcal{D}_{Obs_i}}$ where, for an action $\sigma \in \Sigma$ and a set $B \subseteq \mathcal{D}_V$ of states to be forbidden, $\mathcal{F}^{\mathcal{T}}_i(\sigma, B)$ specifies the set of observation states, for which the action σ must be forbidden by the

[6] Making a parallel with the classical language-based approach, the language L_{Bad} generated by the system from which the set of states Bad has been removed is not controllable w.r.t. the language L of the system, whereas the one generated by the system to which $I(Bad)$ has been removed is actually the largest controllable sub-language of L_{Bad} w.r.t. L. Note that none of these languages is regular.

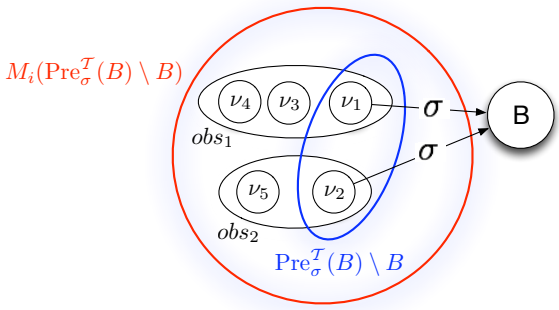


Fig. 3 Computation of $M_i^{\mathcal{T}}(\mathsf{Pre}_\sigma(B) \setminus B)$.

controller $\mathcal{C}_i$ i.e., the smallest set $\mathcal{O}_i$ of observations such that there exists a state $\boldsymbol{\nu} \notin B$ with $M_i(\boldsymbol{\nu}) \in \mathcal{O}_i$, from which a transition labeled by σ leads to B:

$$\mathcal{F}_i^{\mathcal{T}}(\sigma, B) = \begin{cases} M_i(\mathsf{Pre}_\sigma^{\mathcal{T}}(B) \setminus B) & \text{if } \sigma \in \Sigma_{i,c} \\ \emptyset & \text{otherwise} \end{cases} \tag{4}$$

Figure 3 illustrates this computation. Suppose $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ allow to access B in one controllable transition labeled by σ (i.e., $\mathsf{Pre}_\sigma^{\mathcal{T}}(B) \setminus B) = \{\boldsymbol{\nu}_1, \boldsymbol{\nu}_2\}$). Since, $M(\boldsymbol{\nu}_1) = obs_1$ and $M(\boldsymbol{\nu}_2) = obs_2$, these two observations belong to $\mathcal{F}_i^{\mathcal{T}}(\sigma, B) = M_i(\mathsf{Pre}_\sigma^{\mathcal{T}}(B) \setminus B)$.

Finally, the decentralized controller $(\mathcal{C}_i)_{i=[1\ldots n]}$ is such that $\forall i \in [1 \cdots n]$,

$$\mathcal{C}_i = \langle \mathcal{S}_i, E_i \rangle \tag{5}$$

where (i) the supervisory function $\mathcal{S}_i$ is defined, for each $obs \in \mathcal{D}_{Obs_i}$, by

$$\mathcal{S}_i(obs) = \{\sigma_c \in \Sigma \,|\, obs \in \mathcal{F}_i^{\mathcal{T}}(\sigma, I(Bad))\}$$

and (ii) the set $E_i = I(Bad)$.

Remark 2 *For each $i \in [1, n]$, the function $\mathcal{F}_i^{\mathcal{T}}$ can be computed offline. Given a state $\boldsymbol{\nu} \in \mathcal{D}_V$, each controller $\mathcal{C}_i$ $(\forall i \in [1, n])$ computes online the set $\mathcal{S}_i(M_i(\boldsymbol{\nu}))$ (which uses the function $\mathcal{F}_i^{\mathcal{T}}$). Since Σ is finite, $\mathcal{S}_i(M_i(\boldsymbol{\nu}))$ is computable when $I(Bad)$ is computed. Finally, the actions given by the fusion rule $\mathcal{R}^{\mathcal{S}}$ parameterized by the sets $\mathcal{S}_i(M_i(\boldsymbol{\nu}))$ $(\forall i \in [1, n])$ are forbidden.* ◇

The controlled system is computed according to Def. 8 using the decentralized controller $(\mathcal{C}_i)_{i=[1\ldots n]}$. Note that the restricted guards $G_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$ of the controlled system can be computed by $G \setminus \{\bigcap_{i \in \mathrm{In}(\sigma)} M_i^{-1}(\mathcal{F}_i^{\mathcal{T}}(\sigma, I(Bad)))\}$, since the fusion rule, giving the actions to be globally forbidden, corresponds to the set of actions σ which are forbidden by all the controllers involved in the control of this action.

Proposition 1 *The decentralized controller $(\mathcal{C}_i)_{i=[1\ldots n]}$, where each $\mathcal{C}_i$ is defined by (5), solves BDP.*

Proof: We prove by induction on the length ℓ of the execution that $\mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}) \cap I(Bad) = \emptyset$. This would imply that $\mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}) \cap Bad = \emptyset$, because $Bad \subseteq I(Bad)$:

- *Base case* ($\ell = 0$): the set of initial states of the controlled system $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$ is defined by $\Theta_{/(\mathcal{C}_i)_{i=[1\ldots n]}} = \Theta \setminus I(Bad)$. Thus, the execution of $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$ starts in a state that does not belong to $I(Bad)$.
- *Induction case*: suppose the proposition holds for paths of transitions of length less or equal to ℓ. We prove that this property remains true for paths of transitions of length $\ell + 1$. By induction hypothesis, each state $\boldsymbol{\nu}$ reachable with a path of length ℓ does not belong to $I(Bad)$. We show that no transition $\delta = \langle \sigma_\delta, G_\delta, A_\delta \rangle \in \Delta$ can be fired from this state $\boldsymbol{\nu} \notin I(Bad)$ to a state $\boldsymbol{\nu}' \in I(Bad)$. Indeed, either (i) $\delta \in \Delta_c$, then this transition cannot be fired since $\forall i \in \mathrm{In}(\sigma_\delta) : \sigma_\delta \in \mathcal{S}_i(M_i(\boldsymbol{\nu}))$ by the construction of the controller $\mathcal{C}_i$ defined in (5), or (ii) $\delta \in \Delta_{uc}$, then $\boldsymbol{\nu} \in I(Bad)$ (by(3)), which is impossible by hypothesis. □

Example 3 *Back to Example 1 of Figure 1, we assume that the system can be observed through two different observers $\langle Obs_i, M_i \rangle$ (for $i = 1, 2$), where $\langle Obs_1, M_1 \rangle$ is defined as in Example 2 and $\langle Obs_2, M_2 \rangle$ is such that $\mathcal{D}_{Obs_2} = Loc \times \mathbb{N} \times \mathbb{N}$ and $M_2 : Loc \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \mapsto Loc \times \mathbb{N} \times \mathbb{N}$ is defined, for each state $\langle \ell, x, x', y, y' \rangle \in \mathcal{D}_V$, by $M(\langle \ell, x, x', y, y' \rangle) = \langle \ell, x', y' \rangle$. The two local controllers $\mathcal{C}_i$ (for $i = 1, 2$), observing the system through the observer $\langle Obs_i, M_i \rangle$ (for $i = 1, 2$), must ensure that there are at least 11 pieces of each kind: $Bad = \{\langle CX, x, x', y, y' \rangle | (x \leq 10) \wedge (y \in [0, 2]) \wedge (x', y' \in \mathbb{N})\} \cup \{\langle CX', x, x', y, y' \rangle | (x' \leq 10) \wedge (y' \in [0, 2]) \wedge (x, y \in \mathbb{N})\}$. The controllable (resp. uncontrollable) transitions for both local controllers are the ones drawn in plain (resp. dashed) lines in Figure 1. This implies that $I(Bad) = Bad \cup \{\langle CX, x, x', y, y' \rangle | [(x \leq 11) \wedge (y \in [1, 2]) \wedge (x', y' \in \mathbb{N})] \vee [(x \leq 12) \wedge (y = 2) \wedge (x', y' \in \mathbb{N})]\} \cup \{\langle CX', x, x', y, y' \rangle | [(x' \leq 11) \wedge (y' \in [1, 2]) \wedge (x, y \in \mathbb{N})] \vee [(x' \leq 12) \wedge (y' = 2) \wedge (x, y \in \mathbb{N})]\}$. The computation of $\mathcal{F}_1^{\mathcal{T}}$ gives:*

$$\mathcal{F}_1^{\mathcal{T}}(\sigma, I(Bad)) = \begin{cases} M_1(\{\langle PX, x, x', y, y' \rangle | (x \leq 12) \wedge (x', y, y' \in \mathbb{N})\}) & \\ \quad = \{\langle PX, x, y \rangle | (x \leq 12) \wedge (y \in \mathbb{N})\} & \text{if } \sigma = stop\_prod \\ M_1(\{\langle PX', x, x', y, y' \rangle | (x' \leq 12) \wedge (x, y, y' \in \mathbb{N})\}) & \\ \quad = \{\langle PX', x, y \rangle | x, y \in \mathbb{N}\} & \text{if } \sigma = stop\_prod' \\ \emptyset & \text{otherwise} \end{cases}$$

Thus, the controller $\mathcal{C}_1$ always forbids $stop\_prod'$, because it does not observe the variables x' and y'. Similarly, $\mathcal{C}_2$ always forbids $stop\_prod$ and it forbids $stop\_prod'$ when $x' \leq 12$. The controlled system is obtained by restricting the guard of δ_4 (which can no longer be fired when $x \leq 12$) and the guard of δ_8 (which can no longer be fired when $x' \leq 12$). ◇

4.2 Deadlock free state avoidance decentralized control problem

Computation of $I(Bad)$. This set of states (and more generally the function $I(.)$) is defined by the function $\mathsf{Coreach}^{\mathcal{T}}_{uc,bl} : 2^{\mathcal{D}_V} \mapsto 2^{\mathcal{D}_V}$ defined below. This set corresponds to the set of states that can lead to Bad or to deadlocking states in the controlled system triggering only uncontrollable transitions. To compute $\mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad)$, we first compute $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$ (defined by (3)). Then, if we make unreachable the forbidden states by cutting all the controllable transitions that lead to a bad state, the corresponding controlled system could have new deadlocking states. We must add these deadlocking states to the set of forbidden states. The function $\mathsf{Pre}^{\mathcal{T}}_{bl}(B)$ computes, for a set $B \subseteq \mathcal{D}_V$ of states to be forbidden, the set of states, that would be in deadlock in the controlled system, if the states of B were no longer reachable. The computation of the deadlocking states is based on the function $\mathcal{F}^{\mathcal{T}}_i$ $(\forall i \in [1, n])$ defined in (4). To ensure the convergence in the computation of $\mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad)$, $\mathsf{Pre}^{\mathcal{T}}_{bl}$, and therefore $\mathcal{F}^{\mathcal{T}}_i$, must be monotonic. Thus, we use the monotonic function $\widehat{\mathcal{F}}^{\mathcal{T}}_i$ instead of $\mathcal{F}^{\mathcal{T}}_i$ in the computation of the controllers for the deadlock free case:

$$\widehat{\mathcal{F}}^{\mathcal{T}}_i(\sigma, B) = \begin{cases} M_i(\mathsf{Pre}^{\mathcal{T}}_{\sigma}(B)) & \text{if } \sigma \in \Sigma_{i,c} \\ \emptyset & \text{otherwise} \end{cases} \tag{6}$$

We now explain how to compute the deadlocking states in the controlled system $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$. A state $\boldsymbol{\nu} \in \mathcal{D}_V$ is in deadlock in $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$, if the following conditions are satisfied in the system $\mathcal{T}$:

1. the state $\boldsymbol{\nu}$ has no outgoing uncontrollable transitions.
2. for each controllable transition δ, this transition cannot be fired from $\boldsymbol{\nu}$ (i.e., $\boldsymbol{\nu} \notin G_\delta$) or the action σ_δ (which labels δ) is forbidden by the global control for the observations $\langle M_1(\boldsymbol{\nu}), \ldots, M_n(\boldsymbol{\nu})\rangle$ (i.e., $\sigma_\delta \in \mathcal{R}^{\mathcal{S}}(\mathcal{S}_1(M_1(\boldsymbol{\nu})), \ldots, \mathcal{S}_n(M_n(\boldsymbol{\nu})))$). This second condition is equivalent to $\forall i \in \mathrm{In}(\sigma_\delta) : M_i(\boldsymbol{\nu}) \in \widehat{\mathcal{F}}^{\mathcal{T}}_i(\sigma_\delta, B)$, since the fusion rule, which provides the actions to be globally forbidden, corresponds to the set of actions σ which are forbidden by all the local controllers involved in the control of this action.

Definition 10 Formally, for a set of states $B \subseteq \mathcal{D}_V$ to be forbidden, a state $\boldsymbol{\nu}$ is in deadlock if:

1. $\forall \delta \in \Delta_{uc} : \boldsymbol{\nu} \notin G_\delta$, and
2. $\forall \delta \in \Delta_c : (\boldsymbol{\nu} \notin G_\delta) \vee (\forall i \in \mathrm{In}(\sigma_\delta) : M_i(\boldsymbol{\nu}) \in \widehat{\mathcal{F}}^{\mathcal{T}}_i(\sigma_\delta, B))$. •

Since $\widehat{\mathcal{F}}^{\mathcal{T}}_i(\sigma, B) = \emptyset$ $(\forall \sigma \in \Sigma_{uc})$, the function $\mathsf{Pre}^{\mathcal{T}}_{bl}$, which computes the states that would be in deadlock in the controlled system, can be defined as follows:

$$\mathsf{Pre}^{\mathcal{T}}_{bl}(B) = B \cup \left[\bigcap_{\delta \in \Delta} \left(\overline{G_\delta} \cup \bigcap_{i \in \mathrm{In}(\sigma_\delta)} (M_i^{-1}(\widehat{\mathcal{F}}^{\mathcal{T}}_i(\sigma_\delta, B))) \right) \right]$$

Adding the deadlocking states to the forbidden states can provide new states leading uncontrollably to a forbidden state. Consequently, the set $\mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad)$ is computed by the following fixpoint equation:

$$\mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad) = \mathrm{lfp}(\lambda B.Bad \cup \mathsf{Pre}^{\mathcal{T}}_{bl}(\mathsf{Coreach}^{\mathcal{T}}_{uc}(B))) \tag{7}$$

Computation of the local controllers $\mathcal{C}_i$ $(\forall i \in [1,n])$ *and of the controlled system* $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$. The controllers $\mathcal{C}_i$ and the controlled system are computed as in section 4.1.

Proposition 2 *The decentralized controller* $(\mathcal{C}_i)_{i=[1\ldots n]}$*, computed above, solve DFDP.*

Proof: Since $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad) \subseteq \mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad)$, it can be proved as in the proof of Prop. 1 that Bad is not reachable in this more restrictive controlled system.

Let us suppose that the controlled system does not satisfy the deadlock free property. Then, there exists at least one deadlocking state $\boldsymbol{\nu} \in \mathcal{D}_V$, which is reachable in the controlled system. By definition of the fixpoint (7), $\boldsymbol{\nu}$ belongs to $\mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad)$, and so is any state $\boldsymbol{\nu'} \in \mathcal{D}_V$ such that there is a sequence of uncontrollable transitions from $\boldsymbol{\nu'}$ to $\boldsymbol{\nu}$. According to (5), $\boldsymbol{\nu}$ and $\boldsymbol{\nu'}$ are made unreachable by the decentralized controllers and are thus not reachable in $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$. □

4.3 Effective computation by means of abstract interpretation

The actual computation of the controller and in particular the computation of $I(Bad)$, which is based on a fixpoint, is generally not possible for undecidability (or complexity) reasons. To overcome this problem, we use *abstract interpretation* techniques (see e.g. Cousot and Cousot (1977); Halbwachs et al. (1997); Jeannet (2003)). These techniques allow us to compute an overapproximation of the fixpoint $I(Bad)$. The controller obtained with this overapproximation satisfies the control requirements (in particular it prevents from reaching Bad), but it may forbid more states than needed (it is then less permissive).

Outline of the abstract interpretation techniques. In our case, abstract interpretation gives a theoretical framework to the approximate solving of fixpoint equations of the form $x = F(x)$, where $x \in 2^{\mathcal{D}_V}$ and F is a monotonic function. We need to compute the least fixpoint (lfp) of a monotonic function $F : 2^{\mathcal{D}_V} \mapsto 2^{\mathcal{D}_V}$, and since $2^{\mathcal{D}_V}$ is a complete lattice, we know by the Tarski's theorem (Tarski (1955)) that $\mathrm{lfp}(F) = \bigcap\{x \in 2^{\mathcal{D}_V} \mid x \supseteq F(x)\}$. So, any post fixpoint x (with $x \supseteq F(x)$) is an overapproximation of $\mathrm{lfp}(F)$.

We use the following approach to compute a post fixpoint of F: (i) the concrete domain (*i.e.*, the sets of states $2^{\mathcal{D}_V}$) is substituted by a simpler (*possibly infinite*) abstract domain Λ (both domains have a lattice structure); moreover, the concrete lattice $\langle 2^{\mathcal{D}_V}, \subseteq, \cup, \cap, \emptyset, \mathcal{D}_V\rangle$ and the abstract lattice $\langle \Lambda, \sqsubseteq,$

$\sqcup,\sqcap,\bot,\top\rangle$ are linked by a Galois connection $2^{\mathcal{D}_V} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \Lambda$, which ensures the correctness of the method, (*ii*) the fixpoint equation is transposed into the abstract domain; so, the equation to solve has the form: $\ell = F^\sharp(\ell)$ with $\ell \in \Lambda$ and $F^\sharp \sqsupseteq \alpha \circ F \circ \gamma$, (*iii*) a widening operator ∇ ensures that the fixpoint computation converges after a finite number of steps to some upper-approximation ℓ_∞[7], and (*iv*) the concretization $\gamma(\ell_\infty)$ is an overapproximation of the least fixpoint of the function F.

Lattice of convex polyhedra. For our experiments, we use the abstract lattice of *convex polyhedra* of Cousot and Halbwachs (1978). A convex polyhedron on the n-tuple of variables $\langle v_1, \ldots, v_n\rangle$ is defined as a conjunction of k linear constraints. For example, $v_1 \geq 0 \wedge v_2 \geq 0 \wedge v_1 + v_2 \leq 1$ defines a right angle triangle. In this lattice,

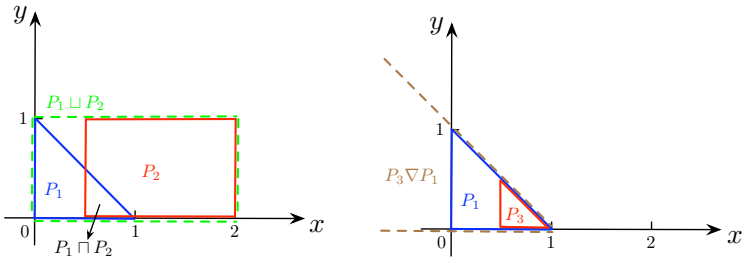
(*i*) $\sqcap$ is the classical intersection, $\sqcup$ is the convex hull and $\sqsubseteq$ is the inclusion.
(*ii*) The *concretization function* $\gamma : \Lambda \mapsto 2^{\mathcal{D}_V}$ is defined by the identity function.
(*iii*) The *abstraction function* $\alpha : 2^{\mathcal{D}_V} \mapsto \Lambda$ is defined as follows: for each set $B \in 2^{\mathcal{D}_V}$, if this set corresponds to a polyhedron, then $\alpha(B)$ is defined by the least convex polyhedron which contains B, otherwise $\alpha(B)$ is defined by $\top$.
(*iv*) The widening operator Cousot and Halbwachs (1978) $P_1 \nabla P_2$ roughly consists in removing from P_1 all the constraints not satisfied by P_2. In other words, if between P_1 and P_2 the value of a variable or a linear expression grows between two steps of the fixpoint computation, then the widening operator considers that it grows indefinitely and thus it removes it.

Example 4 *Let us consider a state space with two integer variables x and y. The convex polyhedron $P_1 = (x \geq 0) \wedge (y \geq 0) \wedge (x+y \leq 1)$ defines a right-angle triangle and the convex polyhedron $P_2 = (x \geq 0.5) \wedge (x \leq 2) \wedge (y \geq 0) \wedge (y \leq 1)$ defines a rectangle (see Figure 4(a)). The intersection of P_1 and P_2 gives the convex polyhedron $P_3 = (x \geq 0.5) \wedge (y \geq 0) \wedge (x+y \leq 1)$ (see Figure 4(a)) and the convex hull of P_1 and P_2 gives the convex polyhedron $P_4 = (x \geq 0) \wedge (x \leq 2) \wedge (y \geq 0) \wedge (y \leq 1)$ (see Figure 4(a)). The widening operation $P_3 \nabla P_1$ gives the convex polyhedron $P_3 \nabla P_1 = (y \geq 0) \wedge (x + y \leq 1)$, because P_1 does not satisfy the linear constraint $x \geq 0.5$ (see Figure 4(b)).* $\diamond$

We assume in the sequel that the abstract lattice $\langle \Lambda, \sqsubseteq, \sqcup, \sqcap, \top, \bot\rangle$, the functions $\alpha : 2^{\mathcal{D}_V} \mapsto \Lambda$, $\gamma : \Lambda \mapsto 2^{\mathcal{D}_V}$, and the widening operator $\nabla : \Lambda \mapsto \Lambda$ are defined, with $2^{\mathcal{D}_V} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \Lambda$.

Computation of the controllers using abstract interpretation. The transposition of the function $\mathsf{Pre}^{\mathcal{T}}_{\Delta_{uc}} : 2^{\mathcal{D}_V} \mapsto 2^{\mathcal{D}_V}$ into the abstract domain gives the

[7] Roughly, a widening operator tries to *guess* the limit of an ascending sequence of elements of the abstract domain in a finite number of steps (see Cousot and Cousot (1977)).



(a) Example of $\sqcap$ and $\sqcup$ operations (b) Example of widening operation

Fig. 4 Examples with abstract lattice of polyhedra

function $\mathsf{Pre}^{\mathcal{T},\sharp}_{\Delta_{uc}} : \Lambda \mapsto \Lambda$ which is defined, for each $\ell \in \Lambda$, as follows:

$$\mathsf{Pre}^{\mathcal{T},\sharp}_{\Delta_{uc}}(\ell) = \bigsqcup_{\delta \in \Delta_{uc}} \mathsf{Pre}^{\mathcal{T},\sharp}_{\delta}(\ell) \text{ , where} \tag{8}$$

$$\mathsf{Pre}^{\mathcal{T},\sharp}_{\delta}(\ell) = \alpha(G_\delta \cap A_\delta^{-1}(\gamma(\ell))) \tag{9}$$

$\mathsf{Coreach}^{\mathcal{T},\sharp}_{uc}(Bad)$ is the least fixpoint of the function $\lambda\ell.\alpha(Bad) \sqcup \mathsf{Pre}^{\mathcal{T},\sharp}_{uc}(\ell)$ and we compute ℓ_∞, defined as the limit of the following sequence:

$$\ell_i = \begin{cases} \alpha(Bad) & \text{if } i = 1 \\ \ell_{i-1} \nabla \mathsf{Pre}^{\mathcal{T},\sharp}_{uc}(\ell_i) & \text{if } i > 1 \end{cases} \tag{10}$$

The abstract interpretation theory ensures that this sequence stabilizes after a finite number of steps, and that $\gamma(\ell_\infty)$ is an overapproximation of $I(Bad)$ (recall that the fixpoint $I(Bad)$ always exists, but may not be computable). Thus, we obtain $I'(Bad) = \gamma(\ell_\infty) \supseteq I(Bad)$. Finally, we define the controller $(\mathcal{C}_i)_{i=[1\ldots n]}$ as in section 4.1 using $I'(Bad)$ instead of $I(Bad)$ and this controller is valid, since $I'(Bad) \supseteq I(Bad)$.

Remark 3 *With this approach only the computation of $I(Bad)$ requires over-approximations. Note that for the deadlock free problem, the same kind of transformations are involved in the effective computation of $I(Bad)$.* ◇

4.4 Comparison between the centralized and decentralized controls

Let us now compare the permissiveness of the n local controllers $\mathcal{C}_i$ ($\forall i \in [1, n]$) computed by the algorithms of section 4 with a centralized controller $\mathcal{C}$ also computed by these algorithms (with one local controller). This controller $\mathcal{C}$ can act on the actions in Σ_c (i.e., $\mathcal{C}$ can control an action σ, if this action can be controlled by at least one of the controllers $\mathcal{C}_i$) and observes the system through the mask $M = \prod_{i\in[1,n]} M_i$ (i.e., when the system arrives in a state $\boldsymbol{\nu}$, the controller receives $\langle obs_1, \ldots, obs_n \rangle$ as information). Therefore, $\mathcal{C}$ cannot distinguish a state $\boldsymbol{\nu} \in \mathcal{D}_V$ from the states in $\bigcap_{i=1}^{n} M_i^{-1}(M_i(\boldsymbol{\nu}))$. The following property shows that $\mathcal{C}$ gives a more permissive solution for BDP.

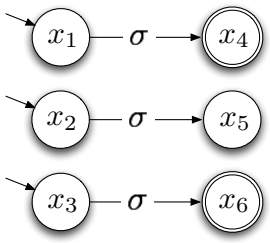


Fig. 5 Centralized controller and decentralized controllers.

Proposition 3 *The centralized controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$ computed with the algorithm in section 4.1 (using the mask $M = \prod_{i \in [1,n]} M_i$ and assuming that $n = 1$) is more permissive than the decentralized controller computed using the same method.*

Proof: Note first that the set $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$ computed for the centralized case is the same than the one computed for the decentralized case, since the masks are not used in these computations.

Then, we show that the centralized controller is as permissive as the decentralized ones by proving that $\forall \boldsymbol{\nu} \in \mathcal{D}_V, \forall \sigma \in \Sigma_c : \sigma \in \mathcal{S}(M(\boldsymbol{\nu})) \Rightarrow \sigma \in \mathcal{R}^{\mathcal{S}}(\mathcal{S}_1(M_1(\boldsymbol{\nu})), \ldots, \mathcal{S}_n(M_n(\boldsymbol{\nu})))$. Indeed, if $\sigma \in \mathcal{S}(M(\boldsymbol{\nu}))$, then there exists a state $\boldsymbol{\nu}' \notin \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$, which is undistinguishable from $\boldsymbol{\nu}$ (in fact $\boldsymbol{\nu}' \in \bigcap_{i=1}^{n} M_i^{-1}(M_i(\boldsymbol{\nu}))$ by definition of the mask M) and from which a transition labeled by σ leads to $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$. The state $\boldsymbol{\nu}' \in M_i^{-1}(M_i(\boldsymbol{\nu}))$ for each $i \in [1, n]$, which implies that each controller $\mathcal{C}_i$, which can control σ, forbids this action in $M_i(\boldsymbol{\nu})$. In consequence, $\sigma \in \mathcal{R}^{\mathcal{S}}(\mathcal{S}_1(M_1(\boldsymbol{\nu})), \ldots, \mathcal{S}_n(M_n(\boldsymbol{\nu})))$.

Finally, the LTS of Fig. 5 shows that the centralized controller can be strictly more permissive than the decentralized controllers. The set of initial states is $X_0 = \{x_1, x_2, x_3\}$, the set $Bad = \{x_4, x_6\}$ and all the controllers can control the unique action σ. The decentralized control is performed by two controllers $\mathcal{C}_1$ and $\mathcal{C}_2$: the first one does not distinguish x_1 and x_2, and the second one does not distinguish x_2 and x_3. The centralized controller $\mathcal{C}$ distinguishes all the states (by definition of its mask M). Hence in the decentralized control, the action σ is forbidden in the states x_1, x_2 and x_3, whereas $\mathcal{C}$ forbids σ only in the states x_1 and x_3. □

We can also extend this result to the deadlock free case.

The following proposition gives a sufficient condition (based on the M-controllability condition in Takai et al. (1994)) to have the same permissiveness between the centralized controller and the decentralized controller. Intuitively, this condition says that if a controllable action is not forbidden by the centralized controller then one of the local controllers of the decentralized controller, which controls it, will not forbid it.

Proposition 4 *The centralized controller and the decentralized controllers, both computed by the algorithm in section 4.1, have the same permissiveness*

if $\forall \boldsymbol{\nu} \notin \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad), \forall \sigma \in \Sigma_c$:

$[\mathsf{Post}^{\mathcal{T}}_{\sigma}(M^{-1}(M(\boldsymbol{\nu})) \setminus \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)) \cap \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad) = \emptyset]$
$\Rightarrow [\exists i \in \mathrm{In}(\sigma) : \mathsf{Post}^{\mathcal{T}}_{\sigma}(M_i^{-1}(M_i(\boldsymbol{\nu})) \setminus \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)) \cap \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad) = \emptyset]$

Proof: In the sequel, we use the terms *equivalence condition* to denote the above condition. By Prop. 3, we must only prove that the decentralized controller has the same permissiveness as the centralized controller. For that, we show that if the centralized controller allows an action σ ($\forall \sigma \in \Sigma$) in a state $\boldsymbol{\nu}$ ($\forall \boldsymbol{\nu} \notin \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$), then it is also allowed in the decentralized control. Two cases must be considered:

- either $\sigma \in \Sigma_{uc}$: in this case, no controller $\mathcal{C}_i$ ($\forall i \in [1, n]$) can control σ. The action σ can thus be fired from $\boldsymbol{\nu}$ in the decentralized control.
- or $\sigma \in \Sigma_c$: by (4) an action is forbidden in $\boldsymbol{\nu}$ iff there exists a state $\boldsymbol{\nu}' \notin \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$ such that (i) $M(\boldsymbol{\nu}) = M(\boldsymbol{\nu}')$ and (ii) $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$ is reachable from $\boldsymbol{\nu}'$ through this action. Therefore, since $\mathcal{C}$ allows σ in the state $\boldsymbol{\nu}$, we have that $\mathsf{Post}^{\mathcal{T}}_{\sigma}(M^{-1}(M(\boldsymbol{\nu})) \setminus \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)) \cap \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad) = \emptyset$. By the equivalence condition, at least one local controller $\mathcal{C}_i$, which can control σ, knows that, from any state of $M_i^{-1}(M_i(\boldsymbol{\nu})) \setminus \mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$, this action does not lead to $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$. In consequence, $\mathcal{C}_i$ allows σ in the state $\boldsymbol{\nu}$, which implies that it will also be allowed in $\boldsymbol{\nu}$ by the global control (after application of the fusion rule). □

A condition similar to the one of Prop. 4 can be given for the deadlock free case by replacing $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad)$ by the sets obtained during the iterations of the fixpoint.

5 State Avoidance Modular Control Problem

In many applications, systems are often composed of several components acting concurrently. With the method of the revious section, the computation of

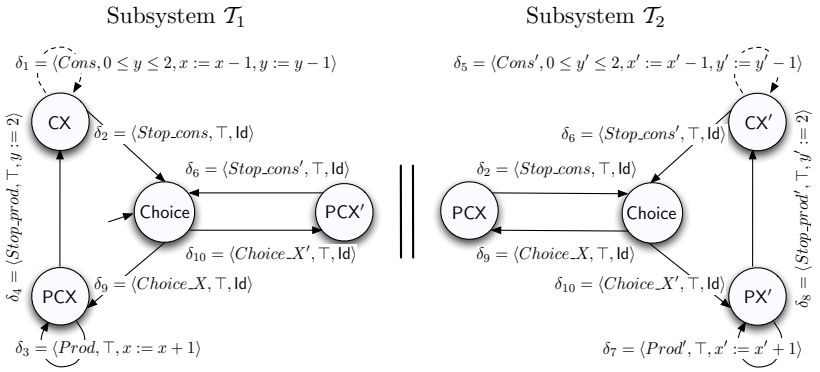


Fig. 6 Modular example of producer and consumer.

the controllers is performed on the whole system. We define here a *modular* method, in which we exploit th concurrent structure of the system to perform locally the computations of a solution, thus avoiding the computation of the global system. To illustrate this point, let us consider again the system $\mathcal{T}$ of Fig. 1. This system has actually the same behavior as the parallel composition of the subsystems $\mathcal{T}_1$ and $\mathcal{T}_2$ depicted in Fig. 6. The corresponding controlled system could have been obtained by performing only local computations on each subsystem. As in the previous section, we focus on the state avoidance control problem, but now we assume that the system is given by a collection of subsystems acting concurrently. A solution of this control problem, exploiting the modularity of the system, has already been given in Gaudin and Marchand (2007). However, this approach only holds for finite systems. We extend it to handle the case of infinite systems and exploit the structure of the system to solve the state avoidance control problem more efficiently. Compared to Gaudin and Marchand (2007) which compute a global controller, we will keep the decentralized architecture and a decentralized controller such that each controller has a partial observation of the system and has its own set of controllable events.

5.1 Model and Statement of the control problem

Model of the system. We assume that the system $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ is defined by the parallel composition of n subsystems modeled by STS $\mathcal{T}_i = \langle V_i, \Theta_i, \Sigma_i, \Delta_i \rangle$ $(\forall i \in [1, n])$: $\mathcal{T} = \mathcal{T}_1 || \ldots || \mathcal{T}_n$. A state $\boldsymbol{\nu} \in \mathcal{D}_V$ is given by an n-tuple $\langle \boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_n \rangle \in \mathcal{D}_{V_1} \times \ldots \times \mathcal{D}_{V_n}$ (where each $\boldsymbol{\nu}_i$ gives the value of the variables of the system $\mathcal{T}_i$). There is no shared variables, meaning that for each $\delta = \langle \sigma, G, A \rangle \in \Delta_i$ the guard G and the assignment A of δ use only the variables of the module $\mathcal{T}_i$. The synchronization between subsystems is achieved through shared events The global set of events is given by $\Sigma = \Sigma_1 \cup \cdots \cup \Sigma_n$. We use the notation Σ_s to represent the set of shared events i.e., $\Sigma_s = \bigcup_{i \neq j \in [1,n]} (\Sigma_i \cap \Sigma_j)$. Given the set of subsystems $\mathcal{T}_i$ of $\mathcal{T}$, Belongs(.) is a function which, for each $\sigma \in \Sigma$, gives the set of indices $i \in \{1, \ldots, n\}$ such that $\sigma \in \Sigma_i$. Finally, we call *product set* a product $\prod_{i=1}^{n} B_i$ of local sets $B_i \subseteq \mathcal{D}_{V_i}$.

Control problem statement. For a control point of view, the alphabet of $\mathcal{T}_i$ is partitioned into the controllable event set $\Sigma_{i,c}$ and the uncontrollable event set $\Sigma_{i,uc}$ i.e., $\Sigma_i = \Sigma_{i,uc} \uplus \Sigma_{i,c}$. We also assume that the shared events are controllable, namely $\forall i : \Sigma_s \cap \Sigma_{i,uc} = \emptyset$. We have $\Sigma_c = \bigcup_{i=1}^{n} \Sigma_{i,c}$ and with the above assumption we have $\Sigma_{uc} = \Sigma \setminus \Sigma_c = \bigcup_{i=1}^{n} \Sigma_{i,uc}$.

In the decentralized framework, each controller has a partial view of the global system. Here, we assume that there is one controller per module. We thus consider a special case of partial observation where each controller has only a local view of the module it controls and a perfect observation of all its local variables (partial local observation could be considered, but gives an

even more elaborate solution). Therefore, we associate to each controller $\mathcal{C}_i$ an observer $\langle Obs_i, M_i \rangle$ defined as follows:

$$\begin{cases} Obs_i = V_i, \\ M_i : \mathcal{D}_V \rightarrow \mathcal{D}_{V_i} \text{ s.t. } M_i(\langle \boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_i, \ldots, \boldsymbol{\nu}_n \rangle) = \boldsymbol{\nu}_i \end{cases} \tag{11}$$

In fact, we do not have here a partial observation, but a *shared* observation, where a state $\boldsymbol{\nu} \in \mathcal{D}_V$ is equal to the tuple of its observations $\boldsymbol{\nu} = \langle M_1(\boldsymbol{\nu}), \ldots, M_n(\boldsymbol{\nu}) \rangle$.

We still want to solve a state avoidance control problem, but a modular one. Therefore, knowing the modular structure of the system, we assume that the set Bad is decomposed according to the structure of the system and is thus defined by a finite union of product sets:

$$Bad = \bigcup_{j=1}^{m} \prod_{i=1}^{n} B_i^j \text{ (with } B_i^j \subseteq \mathcal{D}_{V_i}) \tag{12}$$

Bad can therefore be seen as disjunction of conjunction of local predicates (each conjuction defines a product set). This definition of Bad covers a lot of practical cases.

Problem 3 *Given an STS $\mathcal{T}$ defined by the parallel composition of n STS $\mathcal{T}_i = \langle V_i, \Theta_i, \Sigma_i, \Delta_i \rangle$ $(\forall i \in [1, n])$, n set of observations $\langle Obs_i, M_i \rangle$ $(\forall i \in [1, n])$ (defined as in (11)) and a set of states Bad (defined as in (12)), The Basic State Avoidance Modular Control Problem (BMP for short) consists in computing a decentralized controller $\langle (\mathcal{C}_i)_{i=[1\ldots n]}, \mathcal{R}^{\mathcal{S}}, \mathcal{R}^E \rangle$ such that* $\mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}) \cap Bad = \emptyset$.

In some sense, it is a particular case of State Avoidance Decentralized Control Problem that can be solved locally. Note that to fit the modular structure of the system the fusion rules $\mathcal{R}^{\mathcal{S}}$ and $\mathcal{R}^E$ will be different from the one defined in Definition 2.

5.2 Resolution of the state avoidance modular control problem

As in the previous section, in order to compute controllers $\mathcal{C}_i$ ensuring the avoidance of a set of bad states Bad, we first need compute the set of forbidden states $I(Bad)$. Within the modular framework, we want to find a way to compute this set of states and these controllers without having to compute the entire system.

The Pre *Operator.* The fixpoint, which allows us to compute $I(Bad)$, is based on the $\mathsf{Pre}^{\mathcal{T}}_{\Delta_{uc}}$ operator. Let us first present some properties of this operator which allow us to compute it locally.

Given a concurrent system $\mathcal{T} = \mathcal{T}_1 || \ldots || \mathcal{T}_n$, and a set of states $B_1 \times \ldots \times B_n$, we have:

$$\mathsf{Pre}_A^{\mathcal{T}}(B_1 \times \ldots \times B_n) = \bigcup_{\sigma \in A} \widetilde{\mathsf{Pre}}_\sigma^{\mathcal{T}_1}(B_1) \times \ldots \times \widetilde{\mathsf{Pre}}_\sigma^{\mathcal{T}_n}(B_n) \tag{13}$$

where, $\forall i \in [1, n]$:

$$\widetilde{\mathsf{Pre}}_\sigma^{\mathcal{T}_i}(B_i) = \begin{cases} \mathsf{Pre}_\sigma^{\mathcal{T}_i}(B_i) & \text{if } \sigma \in \Sigma_i \\ B_i & \text{otherwise} \end{cases} \tag{14}$$

The distributivity of the $\mathsf{Pre}_A^{\mathcal{T}}$ is a direct consequence of the assumptions we made on the parallel composition of the modules (no shared state variable). It means that the $\mathsf{Pre}_A^{\mathcal{T}}$ operator on the product set $B_1 \times \ldots \times B_n$ can be computed locally on each set B_i. Using this result, one can easily show the following proposition (see Gaudin and Marchand (2007)).

Proposition 5 *Given a concurrent system $\mathcal{T} = \mathcal{T}_1 || \ldots || \mathcal{T}_n$ and a set of states $B_1 \times \ldots \times B_n$, if $A \subseteq \Sigma \setminus \Sigma_s$, then*

$$\mathsf{Coreach}_A^{\mathcal{T}}(B_1 \times \ldots \times B_n) = \mathsf{Coreach}_{A \cap \Sigma_1}^{\mathcal{T}_1}(B_1) \times \ldots \times \mathsf{Coreach}_{A \cap \Sigma_n}^{\mathcal{T}_n}(B_n) \tag{15}$$

□

Local Computation of the set $I(Bad)$. The following proposition allows us to compute locally $I(Bad)$, the set of states leading uncontrollably to Bad:

Proposition 6 *Given a concurent system $\mathcal{T} = \mathcal{T}_1 || \ldots || \mathcal{T}_n$ with the assumptions as defined in section 5.1 and a set of states $Bad = \bigcup_{j=1}^{m} B_1^j \times \ldots \times B_n^j$, with $B_i^j \subseteq \mathcal{D}_{V_i}$, we have:*

$$I(Bad) = \bigcup_{j=1}^{m} \left(\prod_{i=1}^{n} I_i(B_i^j) \right) \tag{16}$$

where $I_i(B_i^j) = \mathsf{Coreach}_{\Sigma_{i,uc}}^{\mathcal{T}_i}(B_i^j)$.

This proposition means that the computation of $I(Bad)$ can be achieved by the computations of $I_i(B_i^j)$ (for any $j \in [1, m]$ and $i \in [1, n]$). The proof of this proposition is given in Gaudin and Marchand (2007) and is sketched below:

Proof: The computation of the function $I(.)$ can be distributed on each of the m product sets, because $I(B_1 \cup B_2) = I(B_1) \cup I(B_2)$. Then, the computation of $I(.)$ for each product set can be done locally thanks to Prop. 5. □

Synthesis of the local controllers $\mathcal{C}_i = \langle \mathcal{S}_i, E_i \rangle$. Recall that each controller $\mathcal{C}_i$ is only able to observe the variables V_i of the subsystem $\mathcal{T}_i$. Following the construction of the controllers in Subsections 4.1 and 4.2, we first investigate how the function $\widehat{\mathcal{F}}_i^{\mathcal{T}}$ defined as in (6) can be locally computed[8].

For a set of states $B = \bigcup_{j=1}^{m} B^j$ (where $B^j = B_1^j \times \ldots \times B_n^j$ is a product set), the following proposition explains how to locally compute (i.e., on $\mathcal{T}_i$) the function $\widehat{\mathcal{F}}_i^{\mathcal{T}}$ for each product set B^j of B:

Proposition 7 *Given a product set* $B^j = B_1^j \times \ldots \times B_n^j$ *and an event* $\sigma \in \Sigma_i$,

$$\widehat{\mathcal{F}}_i^{\mathcal{T}}(\sigma, B^j) = \begin{cases} \mathsf{Pre}_\sigma^{\mathcal{T}_i}(B_i^j) & \text{if } \sigma \in \Sigma_{i,c} \\ \emptyset & \text{otherwise} \end{cases} \tag{17}$$

Proof: If $\sigma \notin \Sigma_{i,c}$, then $\widehat{\mathcal{F}}_i^{\mathcal{T}}(\sigma, B^j) = \emptyset$ by (6). If $\sigma \in \Sigma_{i,c}$, then:

$$\begin{aligned} \widehat{\mathcal{F}}_i^{\mathcal{T}}(\sigma, B^j) &= M_i(\mathsf{Pre}_\sigma^{\mathcal{T}}(B^j)), \text{ by (6)} \\ &= M_i\left(\widetilde{\mathsf{Pre}}_\sigma^{\mathcal{T}_1}(B_1^j) \times \ldots \times \widetilde{\mathsf{Pre}}_\sigma^{\mathcal{T}_i}(B_i^j) \times \ldots \times \widetilde{\mathsf{Pre}}_\sigma^{\mathcal{T}_n}(B_n^j)\right), \text{ by (13)} \\ &= \widetilde{\mathsf{Pre}}_\sigma^{\mathcal{T}_i}(B_i^j), \text{ by definition of } M_i \\ &= \mathsf{Pre}_\sigma^{\mathcal{T}_i}(B_i^j), \text{ by (14)} \end{aligned}$$

□

We now explain how each local controller $\mathcal{C}_i = \langle \mathcal{S}_i, E_i \rangle$ is computed. According to Prop. 6, we write $I(Bad) = \bigcup_{j=1}^{m} I^j$ where $I^j = I_1^j \times \cdots \times I_n^j$. To define $\mathcal{S}_i$ and E_i, we take into account the decomposition of the system into subsystems and the decomposition of $I(Bad)$ into a union of product sets. This leads us to a new characterization of these two elements. Indeed, let us consider the product set I^j of $I(Bad)$. Then, an action $\sigma \in \Sigma_c$ should be disabled in a state $\boldsymbol{\nu} = \langle \boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_n \rangle$ due to this product set whenever:

1. $\forall i \leq n$ such that $\sigma \in \Sigma_i : \mathsf{Post}_\sigma^{\mathcal{T}_i}(\boldsymbol{\nu}_i) \in I_i^j$ and
2. $\forall i \leq n$ such that $\sigma \notin \Sigma_i : \boldsymbol{\nu}_i \in I_i^j$.

i.e., after σ, each subsystem is in a bad state substate or can uncontrollably lead to a bad substate. Thus, the supervisory function $\mathcal{S}_i$ ($\forall i \in [1, n]$) should indicate whether $\boldsymbol{\nu}_i \in I_i^j$ whenever $\sigma \notin \Sigma_i$ in order to determine the global control function. In consequence, we formally define $\mathcal{S}_i$ and E_i as follows:

1. For each state $\boldsymbol{\nu} = \langle \boldsymbol{\nu}_1, \cdots, \boldsymbol{\nu}_i, \cdots, \boldsymbol{\nu}_n \rangle \in \mathcal{D}_V$, the supervisory function $\mathcal{S}_i$ observes only $\boldsymbol{\nu}_i$ and is given by a pair $\mathcal{S}_i(\boldsymbol{\nu}_i) = \langle \mathcal{P}_i(\boldsymbol{\nu}_i), \mathcal{B}_i(\boldsymbol{\nu}_i) \rangle$ where:
   (a) The function $\mathcal{P}_i(\boldsymbol{\nu}_i)$ gives the set of pairs $\langle j, \sigma \rangle$ such that the action σ must be forbidden in the state $\boldsymbol{\nu}_i$ because of the product set I^j; compared to the previous approach, we memorize the index of the product set for which σ must be forbidden in the state $\boldsymbol{\nu}_i$:

$$\mathcal{P}_i(\boldsymbol{\nu}_i) = \{\langle j, \sigma \rangle \mid (\sigma \in \Sigma_{i,c}) \wedge (j \in [1, m]) \wedge (\boldsymbol{\nu}_i \in \widehat{\mathcal{F}}_i^{\mathcal{T}}(\sigma, I^j))\} \tag{18}$$

[8] Recall that $\widehat{\mathcal{F}}_i^{\mathcal{T}}(\sigma, B)$ gives the set of states for which σ must be forbidden by the controller $\mathcal{C}_i$ to prevent B to be reached. We use $\widehat{\mathcal{F}}_i^{\mathcal{T}}$ instead of $\mathcal{F}_i^{\mathcal{T}}$, because the first function can be computed locally unlike the second one.

(b) The function $\mathcal{B}_i(\boldsymbol{\nu}_i)$ gives the set of indices j such that $\boldsymbol{\nu}_i$ belongs to the forbidden set I_i^j:

$$\mathcal{B}_i(\boldsymbol{\nu}_i) = \{j \mid (j \in [1,m]) \wedge (\boldsymbol{\nu}_i \in I_i^j)\} \tag{19}$$

As explained above, this information will be used by the fusion rule.

2. The set $E_i = \{\langle j, E_i^j\rangle \mid (j \in [1,m]) \wedge (E_i^j = I_i^j)\}$.

We also need to redefine the fusion rules $\mathcal{R}^{\mathcal{S}}$ and $\mathcal{R}^E$ (called $\mathcal{R}_M^{\mathcal{S}}$ and $\mathcal{R}_M^E$ hereafter to stress the modularity architecture) to take into account the decomposition of $I(Bad)$ into product sets as well as the new definition of the supervisory function. For any $\boldsymbol{\nu} = \langle \boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_n\rangle \in \mathcal{D}_V$, we define:

– the fusion rule $\mathcal{R}_M^{\mathcal{S}}$ by:

$$\begin{aligned}\mathcal{R}_M^{\mathcal{S}}(\mathcal{S}_1(\boldsymbol{\nu}_1), \ldots, \mathcal{S}_n(\boldsymbol{\nu}_n)) = \{\sigma \mid \exists j \in [1,m] : [(\forall i \in \mathrm{In}(\sigma) :\\ \langle j, \sigma\rangle \in \mathcal{P}_i(\boldsymbol{\nu}_i)) \wedge (\forall i \notin \mathrm{Belongs}(\sigma) : j \in \mathcal{B}_i(\boldsymbol{\nu}_i))]\}\end{aligned} \tag{20}$$

It means that an action σ is forbidden in the state $\boldsymbol{\nu}$ if there exists a product set I^j such that (i) each controller $\mathcal{C}_i$, which can control σ, decides to forbid it because of this product set, and (ii) each subsystem $\mathcal{T}_i$, which has not σ in its alphabet, is in a forbidden state of I^j.

– the fusion rule $\mathcal{R}_M^E$ as follows: $\boldsymbol{\nu} \in \mathcal{R}_M^E(E_1, \ldots, E_n)$ iff $\exists j \in [1,m] : \boldsymbol{\nu} \in E_1^j \times \ldots \times E_n^j$ (with $\langle j, E_i^j\rangle \in E_i$, $\forall i \in [1,n]$). We can remark that, as expected, $\mathcal{R}_M^E$ forbids the states in $I(Bad)$.

Proposition 8 *The decentralized controller $\langle (\mathcal{C}_i)_{i\in[1,n]}, \mathcal{R}_M^{\mathcal{S}}, \mathcal{R}_M^E\rangle$, defined in this section, solves the BMP.*

Proof: As before, we prove by induction on the length ℓ of the execution that $\mathsf{reachable}(\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}) \cap I(Bad) = \emptyset$:

– *Base case* ($\ell = 0$): The controlled system $\mathcal{T}_{/(\mathcal{C}_i)_{i=[1\ldots n]}}$ starts its execution in a state $\boldsymbol{\nu}$, which belongs to Θ and which is not in $\mathcal{R}_M^E(E_1, \ldots, E_n)$. Thus, this state does not belong to $I(Bad)$.

– *Induction case*: suppose the proposition holds for paths of transitions of length less or equal to ℓ. We prove that this property remains true for paths of transitions of length $\ell + 1$. By induction hypothesis, each state $\boldsymbol{\nu} = \langle \boldsymbol{\nu}_1, \ldots, \boldsymbol{\nu}_n\rangle$ reachable with a path of length ℓ does not belong to $I(Bad)$. We show that no transition $\delta = \langle \sigma_\delta, G_\delta, A_\delta\rangle \in \Delta$ can be fired from this state $\boldsymbol{\nu} \notin I(Bad)$ to a state $\boldsymbol{\nu}' = \langle \boldsymbol{\nu}'_1, \ldots, \boldsymbol{\nu}'_n\rangle \in I(Bad)$ (let $I^j = I_1^j \times \ldots I_n^j$ be the product set of $I(Bad)$ to which $\boldsymbol{\nu}'$ belongs). Indeed:

1. either $\delta \in \Delta_{uc}$, then $\boldsymbol{\nu} \in I(Bad)$ (by(3)), which is impossible by hypothesis.
2. or $\delta \in \Delta_c$: we show that if $\boldsymbol{\nu}'$ is reachable from $\boldsymbol{\nu}$ through the action σ_δ, then this action is forbidden by the global control. We can first remark that, for any $i \in [1,n]$, if $\sigma_\delta \in \Sigma_i$, then $\sigma_\delta \in \Sigma_{i,c}$. Since the implication in the other sense is trivial, we have that $\mathrm{Belongs}(\sigma_\delta) = \mathrm{In}(\sigma_\delta)$. Then, if $\boldsymbol{\nu}'$ is reachable from $\boldsymbol{\nu}$, we have that $\boldsymbol{\nu} \in \mathsf{Pre}_{\sigma_\delta}^{\mathcal{T}}(\boldsymbol{\nu}')$. Next, we prove the desired properties as follows:

(*i*) $\forall i \notin \text{Belongs}(\sigma_\delta) : \boldsymbol{\nu}_i = \boldsymbol{\nu}'_i$ (by definition of the parallel composition of STS), which implies that $\boldsymbol{\nu}_i \in I_i^j$, because $\boldsymbol{\nu}' \in I^j$. Therefore, by 20 we have that $\forall i \notin \text{Belongs}(\sigma_\delta) : j \in \mathcal{B}_i(\boldsymbol{\nu}_i)$.

(*ii*) $\forall i \in \text{In}(\sigma_\delta)(= \text{Belongs}(\sigma_\delta)) : \boldsymbol{\nu}_i \in \mathsf{Pre}_{\sigma_\delta}^{\mathcal{T}_i}(\boldsymbol{\nu}'_i)$ (by definition of the parallel composition of STS), which implies that $\boldsymbol{\nu}_i \in \mathsf{Pre}_{\sigma_\delta}^{\mathcal{T}_i}(I_i^j)$, because $\boldsymbol{\nu}' \in I^j$. Therefore, by (18) we have that $\forall i \in \text{In}(\sigma_\delta) : \langle j, \sigma_\delta \rangle \in \mathcal{P}_i(\boldsymbol{\nu}_i)$.

Finally, $\sigma_\delta \in \mathcal{R}_M^{\mathcal{S}}(\mathcal{S}_1(\boldsymbol{\nu}_1), \ldots, \mathcal{S}_n(\boldsymbol{\nu}_n))$ by (*i*), (*ii*) and (20), which implies that σ_δ cannot be fired from $\boldsymbol{\nu}$. □

Effective Computation by the Means of Abstract Interpretation. The effective computation is here straightforward. We simply compute an overapproximation of each $I_i(B_i^j)$ for all $i \in [1, n]$ and $j \in [1, m]$, and this can be done locally. Then, the computation of each controller can be done as previously, because the number of product sets is finite. We pinpoint that no more approximations are made after we obtained the overapproximation of each $I_i(B_i^j)$, because the fusion of the controllers is done with explicit union (we do not use the convex hull at this time).

Some ideas regarding the control of modular systems with shared uncontrollable events. If we consider the general case, where some uncontrollable events can be shared by the subsystems, the local computation of $I(Bad)$ (where $Bad = \bigcup_{j=1}^m B^j$ is a finite union of product sets) in Prop. 6 is no more valid, because $\Sigma_{uc} \subseteq \Sigma \backslash \Sigma_s$ does not always hold. To compute locally $I(Bad)$, we can proceed as follows, for each product set B^j:

1. we compute locally a fixpoint, considering only unshared uncontrollable events; we obtain a new product set $I_{uc}^{loc}(B^j)$;
2. we compute the Pre operator, considering shared uncontrollable events; we obtain at most as many new product sets as the number of shared uncontrollable events;
3. we repeat this process for all product sets, including the new ones, until stabilization.

With this method, we increase the number of product sets each time we compute the Pre operator of a product set considering shared uncontrollable events. When considering finite systems, like in Gaudin and Marchand (2007), the number of product sets cannot increase infinitely. When considering infinite systems, this property does no longer hold and the fixpoint computation may not terminate.

However, it is still possible to fix an arbitrary limit on the number of product sets and merge some product sets when many of them are present. The result of the merging of two product sets $B^1 = \prod_{i=1}^n B_i^1$ and $B^2 = \prod_{i=1}^n B_i^2$ is the product set $\prod_{i=1}^n (B_i^1 \cup B_i^2)$. When we merge two product sets, we obtain an overapproximation of the set of states they represent.

The result of this merging operation may be very rough, especially when we merge uncomparable product sets. But if we have two comparable product

sets $B^1 = \prod_{i=1}^{n} B_i^1$ and $B^2 = \prod_{i=1}^{n} B_i^2$ (i.e., $\forall i \in [1,n] : \quad B_i^1 \subseteq B_i^2$), we can merge them using the widening operator and obtain:

$$B^1 \nabla B^2 \quad = \quad \prod_{i=1}^{n} (B_i^1 \nabla B_i^2)$$

If we merge only comparable product sets, with the help of the widening operator, we obtain an overapproximation of $I(Bad)$ after a finite number of computation steps. However, the number of uncomparable product sets is exponential w.r.t. the number of variables of the system. As future works, we plan to implement and evaluate with experimental results this approach.

6 Implementation and Empirical Evaluation

6.1 Description of SMACS

We have implemented the previous algorithms in our tool SMACS (Symbolic MAsked Controller Synthesis) which is written in Objective CAML(OCaml (2009)). SMACS uses the APRON library (APRON (2009)) and a generic fixpoint solver(FixPoint (2009)). The input of SMACS is a description of the STS to be controlled with explicit locations and the variables of this system can be either (unbounded) integer or real (float). The guards of the transitions are boolean combinations of linear constraints and the assignments are given by linear expressions. Indeed, the APRON library implements several numerical abstract lattices as intervals(Cousot and Cousot (1977)), octagons(Miné (2001)) and convex polyhedra(Cousot and Halbwachs (1978)) which work well when the guards are linear constraints and the assignments are also linear. In the input, the user can also specify a combination of linear constraints to define the forbidden states and a mask which can be of three kinds:

1. Undistinguishable locations: the controller cannot distinguish some specified locations.
2. Hidden variables: the controller cannot determine the value of these variables
3. Partially hidden variables: the value of a numerical variable is unknown if this value belongs to a specified interval (the user specifies an interval for each variable).

If no mask is specified, the analysis is performed on a system under full observation. The output of SMACS is a description of the function $\mathcal{F}$: it displays, for each action σ, the set of observation states for which σ is forbidden. SMACS can also generate a postscript file to display graphically the controlled system.

SMACS implements the algorithms of Section 4. A modular version of SMACS implements the algorithms of Section 5. Both versions can be downloaded from the SMACS webpage SMACS (2010). In what follows, SMACS refers to the main version of our tool.

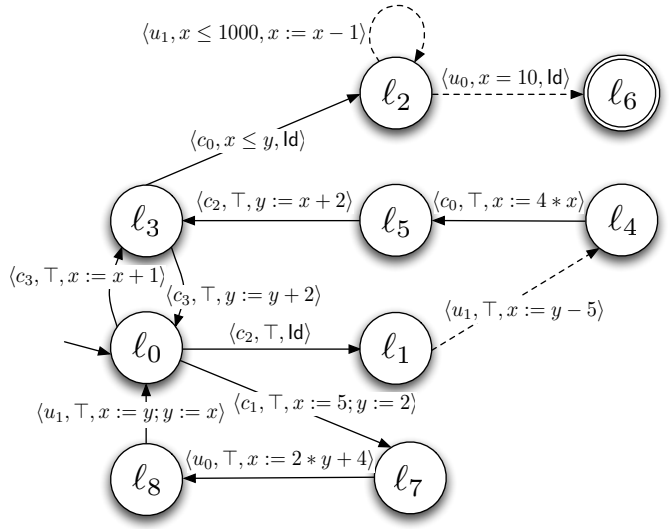


Fig. 7 Toy example.

6.2 Experiments

We define several examples and present the solutions that SMACS has computed for these systems in order to evaluate experimentally our method:

Toy example. The STS of Fig. 7 has explicit locations $\ell \in \{\ell_i \,|\, i \in [0, 8]\}$ and two integers variables x and y. A state of the system is a triple $\langle \ell, x, y\rangle$. Actions c_i (for $i \in [0, 3]$) are controllable and actions u_i (for $i = 0, 1$) are uncontrollable. The initial condition is given by the state $\langle \ell_0, 0, 0\rangle$. The set Bad is defined by $\{\langle \ell_6, k_1, k_2\rangle | k_1, k_2 \in \mathbb{Z}\}$. We consider several cases:

- The controller $\mathcal{C}_1$ has a full observation of the system and the controller $\mathcal{C}_2$ does not distinguish the locations ℓ_3 and ℓ_4 (i.e., $\forall x, y \in \mathbb{Z} : M_2(\langle \ell_3, x, y\rangle) = M_2(\langle \ell_4, x, y\rangle)$). We first suppose that the deadlock free property must not be ensured. SMACS computes a set $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad) = Bad \cup \{\langle \ell_2, x, y\rangle \,|\, 10 \leq x \leq 1000\}$. To prevent the system from reaching this set, it defines a controller $\mathcal{C}_1$ that disables the action c_0 in the location ℓ_3 when $(10 \leq x \leq 1000) \wedge (y \geq x)$ and a controller $\mathcal{C}_2$ that disables the action c_0 in the location ℓ_3 and ℓ_4 when $(10 \leq x \leq 1000) \wedge (y \geq x)$. The global control is similar to the control policy of $\mathcal{C}_1$. The graphical output generated by SMACS for this example is depicted in Fig. 8. When the deadlock free property must be ensured, SMACS computes a set $\mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad) = Bad \cup \{\langle \ell_2, x, y\rangle \,|\, x \geq 10\}$. The states in the set $\{\langle \ell_2, x, y\rangle \,|\, 10 \leq x \geq 1000\}$ are forbidden, because they lead uncontrollably to Bad and the states in the set $\{\langle \ell_2, x, y\rangle \,|\, x \geq 1000\}$ are forbidden, because they are in deadlock. SMACS defines a controller $\mathcal{C}_1$ which forbids

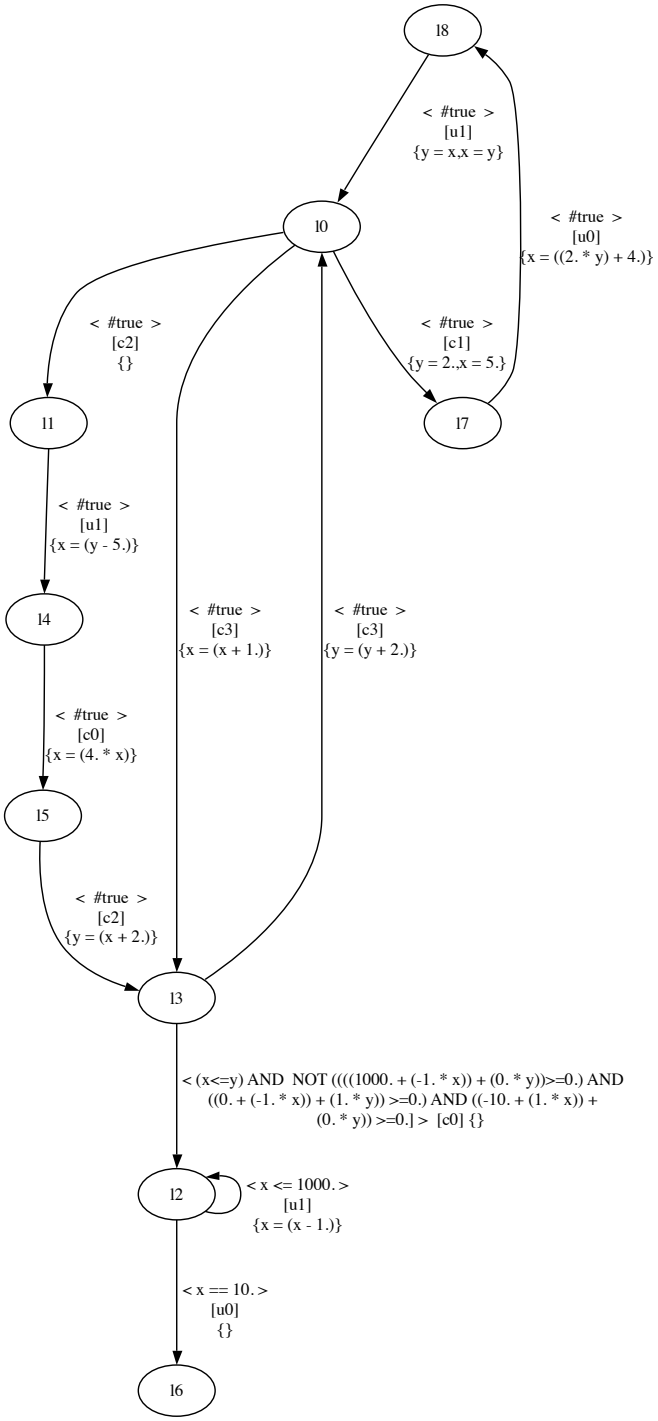


Fig. 8 Output generated by SMACS for the toy example when $\mathcal{C}_1$ has a full observation of the system, $\mathcal{C}_2$ does not distinguish the locations ℓ_3 and ℓ_4, and the deadlock free property is not ensured.

the action c_0 in the location ℓ_3 when $(x \geq 10) \wedge (y \geq x)$ and a controller $\mathcal{C}_2$ which forbids the action c_0 in the location ℓ_3 and ℓ_4 when $(x \geq 10) \wedge (y \geq x)$. The global control is similar to the control policy of $\mathcal{C}_1$.

- The controller $\mathcal{C}_1$ has a full observation of the system and the controller $\mathcal{C}_2$ does not observe the variable x (i.e., $\forall \boldsymbol{\nu} = \langle \ell, x, y \rangle \in \mathcal{D}_V$, the set of states that are undistinguishable from $\boldsymbol{\nu}$ is given by $\{\langle \ell_2, x', y \rangle \,|\, x' \in \mathbb{Z}\}$). We first suppose that the deadlock free property must not be ensured. SMACS computes a set $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad) = Bad \cup \{\langle \ell_2, x, y \rangle \,|\, 10 \leq x \leq 1000\}$. Next, it defines a controller $\mathcal{C}_1$ which forbids the action c_0 in the location ℓ_3 when $(10 \leq x \leq 1000) \wedge (y \geq x)$ and a controller $\mathcal{C}_2$ which forbids the action c_0 in the location ℓ_3 when $y \geq 10$. The global control is similar to the control policy of $\mathcal{C}_1$. When the deadlock free property must be ensured, SMACS computes a set $\mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad) = Bad \cup \{\langle \ell_2, x, y \rangle \,|\, x \geq 10\}$. SMACS defines a controller $\mathcal{C}_1$ which forbids the action c_0 in the location ℓ_3 when $(x \geq 10) \wedge (y \geq x)$ and a controller $\mathcal{C}_2$ which forbids the action c_0 in the location ℓ_3 when $(y \geq 10)$. The global control is similar to the control policy of $\mathcal{C}_1$.
- The controller $\mathcal{C}_1$ does not distinguish the locations ℓ_3 and ℓ_4, and the controller $\mathcal{C}_2$ does not observe the variable x. We first suppose that the deadlock free property must not be ensured. SMACS computes a set $\mathsf{Coreach}^{\mathcal{T}}_{uc}(Bad) = Bad \cup \{\langle \ell_2, x, y \rangle \,|\, 10 \leq x \leq 1000\}$. Next, it defines a controller $\mathcal{C}_1$ which forbids the action c_0 in the location ℓ_3 and ℓ_4 when $(10 \leq x \leq 1000) \wedge (y \geq x)$ and a controller $\mathcal{C}_2$ which forbids the action c_0 in the location ℓ_3 when $y \geq 10$. The global control consists in forbidding the action c_0 in the location ℓ_3 when $(10 \leq x \leq 1000) \wedge (y \geq x)$. When the deadlock free property must be ensured, SMACS computes a set $\mathsf{Coreach}^{\mathcal{T}}_{uc,bl}(Bad) = Bad \cup \{\langle \ell_2, x, y \rangle \,|\, x \geq 10\}$. SMACS defines a controller $\mathcal{C}_1$ which forbids the action c_0 in the location ℓ_3 and ℓ_4 when $(x \geq 10) \wedge (y \geq x)$ and $\mathcal{C}_2$ which forbids the action c_0 in the location ℓ_3 when $(y \geq 10)$. The global control consists in forbidding the action c_0 in the location ℓ_3 when $(x \geq 10) \wedge (y \geq x)$.

All these computations are done in few ms. Next, we consider modified versions of the toy example where we add variables to make the systems to be controlled more complex. The set of forbidden states is defined over these new variables and SMACS must compute a controller $\mathcal{C}_1$ (this controller does not distinguish the locations ℓ_3 and ℓ_4) and a controller $\mathcal{C}_2$ (this controller does not observe the variable x) which satisfy the specification. The results are reportes in Table 1. For example, if we consider the case with 170 variables, SMACS computes controllers $\mathcal{C}_1$ and $\mathcal{C}_2$ which fulfills the requirement in 21,1 seconds and uses for that 84 MB of RAM memory.

Producer and consumer. We consider a modified version of the system defined in Example 3 (Fig. 1), where the transitions δ_1, δ_4, δ_5 and δ_8 are respectively replaced by $\langle Cons; 0 \leq x \leq 500; x := x - 1, y := y - 1 \rangle$, $\langle Stop\_prod; \mathsf{TT}_{\mathcal{D}_V}; y := 500 \rangle$, $\langle Cons'; 0 \leq x' \leq 500; x' := x' - 1, y' := y' - 1 \rangle$, and $\langle Stop\_prod'; \mathsf{TT}_{\mathcal{D}_V}; y' := 500 \rangle$, which makes the system *harder* to be con-

System	Number of Variables	Time (seconds)	Maximal Memory (MB)
Toy	130	9.4	72
	170	21.1	84
	210	35.9	96
	250	58.9	108
	290	93.3	144

Table 1 Time (in seconds) and memory (in MB) used by SMACS to control modified versions of the toy example.

trolled. The set Bad of forbidden states is defined by $\{\langle \mathsf{CX}, x, x', y, y' \rangle \,|\, (x \leq 10) \wedge (y \in [0, 500])\} \cup \{\langle \mathsf{CX}', x, x', y, y' \rangle \,|\, (x' \leq 10) \wedge (y' \in [0, 500])\}$. The aim is to synthesize two controllers $\mathcal{C}_1$ and $\mathcal{C}_2$ which satisfy the control specification. $\mathcal{C}_1$ does not control the actions $Cons$, $Cons'$ and $Stop\_prod'$ and $\mathcal{C}_1$ does not control the actions $Cons$, $Cons'$ and $Stop\_prod$. We consider several cases:

- The controller $\mathcal{C}_1$ does not observe the variables x and y (i.e., $\forall \boldsymbol{\nu} = \langle \ell, x, y, x', y' \rangle \in \mathcal{D}_V$, the set of states that are undistinguishable from $\boldsymbol{\nu}$ is given by $\{\langle \ell, x_1, y_1, x', y' \rangle \,|\, x_1, y_1 \in \mathbb{Z}\}$), the controller $\mathcal{C}_2$ has a full observation of the system, and the deadlock free property is not ensured. SMACS defines a controller $\mathcal{C}_1$ which always forbids the action $Stop\_prod$ in the location PX and a controller $\mathcal{C}_2$ which forbids the action $Stop\_prod'$ in the location PX' when $x' \geq 510$. The global control consists in forbidding the action $Stop\_prod$ in the location PX and the action $Stop\_prod'$ in the location PX' when $x' \geq 510$.
- The controller $\mathcal{C}_1$ has a full observation of the system, the controller $\mathcal{C}_2$ does not observe the variables x' in the interval $[500, 600]$ (i.e., for each state $\boldsymbol{\nu} = \langle \ell, x, y, x', y' \rangle \in \mathcal{D}_V$ such that $x' \in [500, 600]$, the set of states that are undistinguishable from $\boldsymbol{\nu}$ is given by $\{\langle \ell, x, y, x'_1, y' \rangle \,|\, x'_1 \in [500, 600]\}$), and the deadlock free property is not ensured. SMACS defines a controller $\mathcal{C}_1$ which forbids the action $Stop\_prod$ in the location PX' when $x \geq 510$ and a controller $\mathcal{C}_2$ which forbids the action $Stop\_prod'$ in the location PX' when $x' \geq 600$. The global control consists in forbidding the action $Stop\_prod$ in the location PX' when $x \geq 510$ and the action $Stop\_prod'$ in the location PX' when $x' \geq 600$.
- The controller $\mathcal{C}_1$ does not observe the variables x and y, the controller $\mathcal{C}_2$ does not observe the variables x' in the interval $[500, 600]$, and the deadlock free property is not ensured. SMACS defines a controller $\mathcal{C}_1$ which always forbids the action $Stop\_prod$ in the location PX and a controller $\mathcal{C}_2$ which forbids the action $Stop\_prod'$ in the location PX' when $x' \geq 600$. The global control consists in forbidding the action $Stop\_prod$ in the location PX and the action $Stop\_prod'$ in the location PX' when $x' \geq 600$.

All these computations are done in few ms. Now, we consider modified versions of the producer and consumer (See Figure 2). The system is made more complex by producing n (where $n > 0$) kinds of pieces. The production of each kind of pieces requires the definition of two variables x_i and y_i, and the

System	Number of Variables	Time (seconds)	Maximal Memory (MB)
Producer and Consumer	40	4.4	84
	80	50.2	168
	120	231.6	348
	140	404.4	456
	160	686.9	660

Table 2 Time (in seconds) and memory (in MB) used by SMACS to control modified versions of the producer and consumer example.

control requirements consist in setting, for each kind of pieces, a bound on the number of produced pieces. SMACS must compute n controllers $\mathcal{C}_i$ (this controller does not observe the value of the variable x_i when it belongs to the interval $[400, 600]$). For example, if we consider the case with 80 variables (i.e., there 40 kinds of pieces), SMACS computes a controller which fulfills the requirements in 50.2 seconds and uses 168 MB of RAM memory.

Comparison between modular control and decentralized control We have also implemented our modular controller to compare it with our decentralized controller. The experimentations have been done on the producer and consumer example and are reported in Table 3. For the decentralized case, we consider the producer and consumer system described above and producing n kinds of pieces X_i ($\forall i \in [1, n]$). The decentralized controller must ensure that they are at least 10 pieces of each kind and is composed of n local controllers $\mathcal{C}_i$. Each local controller $\mathcal{C}_i$ observes the variables ℓ, x_i, y_i involved in the production of the pieces of kind X_i. Note that the mask is not of the same kind as in the experiments presented in Table 2. This explains why the time performance are not indentical in Tables Table 2 and 3.

For modular controller, the system to be controlled is composed of n subsystems $\mathcal{T}_i$. Each subsystem $\mathcal{T}_i$ produces the pieces of kind X_i and the modular controller must ensure that they are at least 10 pieces of each kind. Our experimentations show that the modular controller gives better results than the decentralized controller (see Table 3). For example, for the case where 100 kinds of pieces are produced (i.e., the case with 200 variables), the modular controller computes the solution 50 times faster and uses 100 times less memory.

7 Conclusion

We investigated the state avoidance decentralized control problem (basic and deadlock free) for infinite discrete event systems modeled by Symbolic Transition Systems (STS). Even if these problems are undecidable, our algorithms terminate (using overapproximations) and return valid controllers. We implemented these algorithms in our tool SMACS and tested them on several examples. Moreover, we proposed an algorithm to solve the basic state avoidance

System	Number of Variables	Decentralized		Modular	
		Time (seconds)	Maximal Memory (MB)	Time (seconds)	Maximal Memory (MB)
Producer and Consumer	40	1.3	84	0.2	< 12
	60	5	108	0.5	< 12
	80	12.7	168	0.7	< 12
	100	28	228	1.2	< 12
	120	53.5	300	1.5	< 12
	140	93.7	444	2.2	< 12
	160	153.8	612	3.2	< 12
	180	240.8	852	6.2	< 12
	200	362.7	1128	7.1	< 12

Table 3 Time (in seconds) and memory (in MB) used by SMACS to control modified versions of the producer and consumer example.

modular control problem, at least when there are no shared uncontrollable events, and gave some ideas to solve it in the most general case.

We consider this work as a first step to solve a more complex issue: the distributed control problem of infinite state systems where the system is modeled by a collection of sub-systems communicating through asynchronous channels[9]. Future work also include the improvement of our tool SMACS in order to handle industrial-size examples.

References

APRON, 2009. The APRON library. Http://apron.cri.ensmp.fr/.

Cassandras, C., Lafortune, S., 2008. Introduction to Discrete Event Systems (2nd edition). Springer.

Cousot, P., Cousot, R., 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL'77. pp. 238–252.

Cousot, P., Halbwachs, N., 1978. Automatic discovery of linear restraints among variables of a program. In: POPL '78. pp. 84–96.

FixPoint, 2009. Fixpoint: an OCaml library implementing a generic fix-point engine. Http://pop-art.inrialpes.fr/people/bjeannet/bjeannet-forge/fixpoint/.

Gaudin, B., Marchand, H., 2007. An efficient modular method for the control of concurrent discrete event systems: A language-based approach. Discrete Event Dynamic Systems 17 (2), 179–209.

Halbwachs, N., Proy, Y., Roumanoff, P., August 1997. Verification of real-time systems using linear relation analysis. Formal Methods in System Design 11 (2), 157–185.

[9] Note that in this situation, the control architecture (i.e the fusion rule) that we considered in this paper is no longer valid.

Jeannet, B., 2003. Dynamic partitioning in linear relation analysis. Application to the verification of reactive systems. Formal Meth. in Syst. Design 23 (1), 5–37.

Jeannet, B., Jéron, T., Rusu, V., Zinovieva, E., April 2005. Symbolic test selection based on approximate analysis. In: TACAS'05, Volume 3440 of LNCS. Edinburgh, pp. 349–364.

Kalyon, G., Le Gall, T., Marchand, H., Massart, T., August 2009. Control of infinite symbolic transition systems under partial observation. In: European Control Conference. Hungary, pp. 1456–1462.

Kumar, R., Garg, V., 2005. On computation of state avoidance control for infinite state systems in assignment program model. IEEE Trans. on Autom. Science and Engineering 2 (2), 87–91.

Kumar, R., Garg, V., Marcus, S., 1993. Predicates and predicate transformers for supervisory control of discrete event dynamical systems. IEEE Trans. Autom. Control 38 (2), 232–247.

Le Gall, T., Jeannet, B., Marchand, H., December 2005. Supervisory control of infinite symbolic systems using abstract interpretation. In: CDC/ECC'05. pp. 31–35.

Miné, A., October 2001. The octagon abstract domain. In: Proc. of the Workshop on Analysis, Slicing, and Transformation (AST'01). IEEE. IEEE CS Press, Stuttgart, Germany, pp. 310–319.

OCaml, 2009. The programming language Objective CAML. Http://caml.inria.fr/.

Ramadge, P., Wonham, W., 1989. The control of discrete event systems. Proceedings of the IEEE; Special issue on Dynamics of Discrete Event Systems 77 (1), 81–98.

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

Rudie, K., Wonham, W. M., 1992b. An automata-theoretic approach to automatic program verification. In: Proceedings of the IEEE Conference on Decision and Control (CDC). Tucson, Arizona, pp. 3770–3777.

SMACS, 2010. The SMACS tool. `http://www.smacs.be/`.

Takai, S., 1998. On the languages generated under fully decentralized supervision. IEEE Transactions on Automatic Control 43 (9), 1253–1256.

Takai, S., Kodama, S., 1997. M-controllable subpredicates arising in state feedback control of discrete event systems. International Journal of Control 67 (4), 553–566.

Takai, S., Kodama, S., July 1998. Characterization of all m-controllable subpredicates of a given predicate. International Journal of Control 70 (9), 541–549.

Takai, S., Kodama, S., Ushio, T., 1994. Decentralized state feedback control of discrete event systems. Syst. Control Lett. 22 (5), 369–375.

Tarski, A., 1955. A lattice-theoretical fixpoint theorem and its applications. Pacific Journal of Mathematics 5, 285–309.

Wonham, W., Ramadge, P., 1988. Modular supervisory control of discrete-event systems. Mathematics of Control, Signals, and Systems 1 (1), 13–30.

Yoo, T., Lafortune, S., August 2000. A general architecture for decentralized supervisory control of discrete-event systems. In: Proc of 5th Workshop on Discrete Event Systems, WODES 2000. Ghent, Belgium, pp. 335–377.