Control of Infinite Symbolic Transition Systems under Partial Observation

Gabriel Kalyon, Tristan Le Gall, Hervé Marchand and Thierry Massart

Abstract— **We propose algorithms for the synthesis of state-feedback controllers through partial observation of infinite state systems modelled by Symbolic Transition Systems. We provide models of safe controllers both for potentially blocking and non blocking controlled systems. To obtain algorithms for these problems, we use abstract interpretation techniques which provide over-approximations of the transitions set to be disabled. To our knowledge, with the hypotheses taken, the improved version of our algorithm provides a better solution than what was previously proposed in the literature. Our tool SMACS allowed us to make an empirical validation of our methods to show their feasibility and usability.**

Keywords: Symbolic Transition Systems, Control Synthesis, Partial Observation, Abstract Interpretation.

I. Introduction

Discrete event systems control theory provides synthesis methods for a controller that usually has a full observation of the plant, modelled by a finite state system and can disable controllable actions. This simple and optimistic view of the problem is not always satisfactory. Indeed, in practice, the controller interacts with the plant through sensors and actuators, and an extended model with variables may be better suited to specify the plant. In that case, to provide an homogeneous treatment of these models, it is convenient to consider infinite variables domains. Moreover, the hypothesis of full observation can generally not be made either because the sensors only have finite precision or because some parts of the plant are not observed by the controller.

In this paper, we address the controller synthesis of partially observed infinite state systems to solve the *state avoidance problem*, where the controller's goal consists in preventing the system from reaching a specified set of states Bad. We use Symbolic Transition Systems (STS) [9] to model the plant, where an STS is a transition system defined over a set of variables whose domain can be infinite; each transition is *guarded* on the system variables, and has an *update* function which indicates the variables changes when the transition is fired. Furthermore, transitions are labelled with symbols taken from a finite alphabet. The semantics of an STS is therefore given by a potentially infinite state labelled transition system where the states are valuations of the variables.

When 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 [21]. Moreover, the controller gets in general only *partial observation*, because of the imprecision of the observing material. So, we follow the approach taken by [14], where the partial observation is modelled by a mask, corresponding to a mapping from the state space to an (infinite) observation space.

Related works: The *controller synthesis of finite state systems with partial observation* of the actions has been widely studied in various works. The problem with partial observation on the states (mask) has been introduced by Kumar et al. in [14]. In [19] properties of *M-controllability* give sufficient conditions to ensure controllability. To synthesize the controlled system, they use a *forward* approach with a *post* operator. Hill et al. extend this work in [10] and provide a method which synthesizes more permissive controllers, but with a different hypothesis on the masks. Since we take infinite state systems and use abstract interpretation techniques, we have preferred a *backward* approach. In game theory, the controller synthesis problem can be stated as the synthesis of a winning strategy in a two players game between the plant and the controller. The cases of *imperfect and incomplete information games* have been studied for finite state systems (see e.g. [4]).

Controller synthesis of infinite state systems modelled by STS in the case of full observation has been examined in a previous work [16]. We used abstract interpretation techniques to ensure that the controlled system can be effectively computed. We showed that, since these abstract interpretation techniques induce an over-approximation of the computations, this implies that the computed controlled system in not always the most permissive. In [15], Kumar and Garg extend their previous work [14] to consider infinite systems. They prove that, in that case, the state avoidance control problem is undecidable. They also show that the problem can be solved in the case of Petri nets, when the set Bad is upward-closed. The controller synthesis of *infinite state systems* modelled by Petri nets has also been considered in [11].

In order to deal with the infiniteness of state space, the algorithms presented in this paper are symbolic: they do not enumerate individual states, but deal with the system variables by means of symbolic computations and the use

G. Kalyon, T. Le Gall and T. Massart are with the Université Libre de Bruxelles (U.L.B.), First.Last@ulb.ac.be

H. Marchand is with the IRISA/INRIA, Campus de Beaulieu, Rennes, France, First.Last@irisa.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.

ISBN 978-963-311-369-1

of predicate transformers. Moreover, since the problem is undecidable, we use abstract interpretation techniques (see e.g. [5], [8], [12]) to get effective algorithms (i.e. which always terminate). It is worth noticing that *both concrete and abstract domains can be infinite*. Those algorithms were implemented in a tool named SMACS.

In section II, we introduce our model for infinite systems to be controlled. In section III, we define the control mechanisms we can use and we define the state avoidance control problem. In section IV, we present an algorithm, which solves our problem, but which does not always terminate. In section V, we explain how to obtain an effective algorithm using abstract interpretation techniques. In section VI, we experimentally validate our method on various examples.

II. Symbolic Transition Systems

The (infinite) domain of a variable v is denoted $\mathcal{D}_v$. If $V = \langle v_1, \ldots, v_n \rangle$ is a tuple of variables, we note $\mathcal{D}_V = \prod_{i\in[1,n]} \mathcal{D}_{v_i}$. A *valuation* $\vec{\nu}$ of V is a tuple $\langle \vec{\nu}_1, \ldots, \vec{\nu}_n \rangle \in \mathcal{D}_V$. A *predicate* over a tuple V is defined as a subset $P \subseteq \mathcal{D}_V$ (a state set for which the predicate holds). The complement of a set $H \subseteq \mathcal{D}_V$ is denoted by $\overline{H}$. The preimage function of $f : D_1 \to D_2$ is denoted by $f^{-1} : D_2 \to 2^{D_1}$.

Definition 1 (Symbolic Transition System): A *symbolic transition system* (STS) is a tuple $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ where $V = \langle v_1, \ldots, v_n \rangle$ is a tuple of variables, $\Theta \subseteq \mathcal{D}_V$ is a predicate on V defining the initial condition on the variables, Σ is a finite alphabet of actions and Δ is a finite set of symbolic transitions $\delta = \langle \sigma_\delta, G_\delta, A_\delta \rangle$ where:

- $\sigma_\delta \in \Sigma$ is the action of δ,
- $G_\delta \subseteq \mathcal{D}_V$ is a predicate on V, which guards δ,
- $A_\delta : \mathcal{D}_V \mapsto \mathcal{D}_V$ is the update function of δ.

Given an action $\sigma \in \Sigma$, we define the set of transitions labelled by σ as $\mathrm{Trans}(\sigma) = \{\delta \in \Delta \mid \sigma_\delta = \sigma\}$. The semantics of an STS is a possibly infinite Labelled Transition System (LTS) where states are valuations of its variables:

Definition 2 (STS's Semantics): The semantics of an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ is an LTS $[\![\mathcal{T}]\!] = \langle Q, Q_0, \Sigma, \to \rangle$ where $Q = \mathcal{D}_V$ is the set of states, $Q_0 = \Theta$ is the set of initial states, Σ is the set of labels and $\to \subseteq Q \times \Sigma \times Q$ is the transition relation defined as $\{\langle \vec{\nu}, \sigma, \vec{\nu}' \rangle \mid \exists \delta \in \Delta : (\sigma_\delta = \sigma) \wedge (\vec{\nu} \in G_\delta) \wedge (\vec{\nu}' = A_\delta(\vec{\nu}))\}$.

Note that the LTS $[\![\mathcal{T}]\!]$ can be non-deterministic.

Initially, an STS is in one of its initial states. A transition can only be fired if its guard is satisfied and when fired, the variables are updated according to its update function. If no transition can be fired from a state $\vec{\nu} \in \mathcal{D}_V$, i.e. $\forall \delta \in \Delta : \vec{\nu} \notin G_\delta$, we say that this state is *blocking*.

Given an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, $\mathsf{reachable}(\mathcal{T}) \subseteq \mathcal{D}_V$ is defined as the set of states that are reachable from an initial state in $[\![\mathcal{T}]\!]$.

An STS may be defined with explicit locations. This is equivalent to having a finite variable of enumerated type, which encodes the locations. Therefore, in our examples, we generally represent STS using locations.

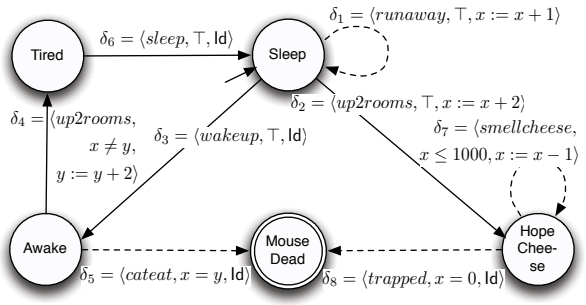

Fig. 1. The cat and mouse example

Example 1: The STS of Fig. 1 illustrates a modified version of the cat and mouse example given in [15]. *Id* denotes the identity function. Fig. 1 will be used in this paper, with different values for the guards G_2 and G_3 (initially $\top$). The STS has explicit locations ℓ and two natural variables: x (resp. y) identifies the room number occupied by the mouse (resp. the cat). A system state is a triple $\langle \ell, x, y \rangle$. The initial condition is given by the state $\langle Sleep, 1, 0 \rangle$. When the cat wakes up, she can eat the mouse if both are in the same room, or move and sleep again. In the location HopeCheese, if the mouse is in one of the first 1000 rooms, he can smell the cheese and moves to the room 0, where he is killed by a trap.

III. State avoidance 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.

A. Means of observation

We consider systems with partial observation, where there is an uncertainty about the current state of the system. This partial observation is formally defined by a mask $M : \mathcal{D}_V \to Y$, which corresponds to a mapping from the state space $\mathcal{D}_V$ to the (possibly infinite) observation space Y. So, Y can be seen as a partition of $\mathcal{D}_V$, where each equivalence class contains the states with the same mask.

Example 2: For the system of Fig. 1, the localization of the cat is unknown. So, the mask $M : Loc \times \mathbb{N} \times \mathbb{N} \to Loc \times \mathbb{N}$ is defined as follows: $M(\langle \ell, x, y \rangle) = \langle \ell, x \rangle$.

In the sequel, we consider three kinds of partial observation:

1) two locations (or more) give the same observation: in this case, the controller is not sure about the exact location of the system.
2) some variables are hidden: the controller cannot determine the value of those variables.
3) the value of a numerical variable is unknown if this value belongs to a specified interval. This mask implements variables that are *partially hidden*.

B. Means of control

The control mechanism is similar to the one defined in [18], [3] : the alphabet $\Sigma = \Sigma_c \cup \Sigma_{uc}$ is partitioned into Σ_c, the set of controllable actions, and Σ_{uc}, the set

of uncontrollable ones. As a consequence, the set Δ is partitioned accordingly to Δ_c and Δ_{uc}.

C. Controller and controlled system

The controller aims to restrict the system's behavior and to prevent it from reaching some bad states. The controller with partial observation is formally defined as follows:

Definition 3 (Controller): Given an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, and a mask $M : \mathcal{D}_V \mapsto Y$, a controller for $\mathcal{T}$ is a pair $\mathcal{C} = \langle \mathcal{S}, E \rangle$, where (i) $\mathcal{S} : Y \rightarrow 2^{\Sigma_c}$ is a supervisory function which defines, for an observation $y \in Y$, a set $\mathcal{S}(y)$ of controllable actions to forbid in any state $\vec{\nu}$ such that $y = M(\vec{\nu})$, and (ii) $E \subseteq \mathcal{D}_V$ is a set of states to forbid, which restricts the set of initial states.

The behavior of the controlled system is defined as follows:

Definition 4 (Controlled STS): Given an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, a mask $M : \mathcal{D}_V \mapsto Y$, and a controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$, the system $\mathcal{T}$ controlled by $\mathcal{C}$, is an STS $\mathcal{T}_{/\mathcal{C}} = \langle V, \Theta_{/\mathcal{C}}, \Sigma, \Delta_{/\mathcal{C}} \rangle$, where $\Theta_{/\mathcal{C}} = \Theta \setminus E$ and $\Delta_{/\mathcal{C}}$ is defined using the following rule:

$$\frac{\langle \sigma, G, A \rangle \in \Delta \quad G_{/\mathcal{C}} = G \setminus \{\vec{\nu} \in \mathcal{D}_V | \sigma \in \mathcal{S}(M(\vec{\nu}))\}}{\langle \sigma, G_{/\mathcal{C}}, A \rangle \in \Delta_{/\mathcal{C}}}$$

The supervisory function $\mathcal{S}$ allows us to restrict the guards of the controlled system. Indeed, a transition δ can no longer be fired in $\mathcal{T}_{/\mathcal{C}}$ from a state $\vec{\nu}$, if its action $\sigma_\delta \in \mathcal{S}(M(\vec{\nu}))$. This function satisfies the *S-observability* condition meaning that if $\vec{\nu}$ and $\vec{\nu}'$ have the same observation, then $\mathcal{S}$ will have the same control decision for both states.

D. Definition of the problems

We focus on two variants of the state avoidance control problem :

Problem 1 (Basic state avoidance control problem): For an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, a mask $M : \mathcal{D}_V \mapsto Y$ and a predicate Bad, i.e. a set of forbidden states, the basic state avoidance control problem consists in building a controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$ such that $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}}) \cap Bad = \varnothing$.

A solution to this first problem does not ensure that the controlled system is non-blocking. To ensure this important property, we define a second problem.

Problem 2 (Non-blocking state avoidance control problem): This problem consists in defining a controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$ such that (i) $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}}) \cap Bad = \varnothing$, and (ii) $\forall \vec{\nu} \in \mathsf{reachable}(\mathcal{T}_{/\mathcal{C}}), \exists \delta \in \Delta_{/\mathcal{C}} : \vec{\nu} \in (G_{/\mathcal{C}})_\delta$.

We can immediately notice that a trivially correct controller (for both problems) is one where $E = \mathcal{D}_V$.

Therefore, the notion of permissiveness has been introduced to compare the quality of different controllers for a given STS.

Definition 5 (Permissiveness): Given an STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, and a mask $M : \mathcal{D}_V \mapsto Y$, a controller $\mathcal{C}_1 = \langle \mathcal{S}_1, E_1 \rangle$ is more permissive than a controller $\mathcal{C}_2 = \langle \mathcal{S}_2, E_2 \rangle$, iff $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}_1}) \supseteq \mathsf{reachable}(\mathcal{T}_{/\mathcal{C}_2})$. When the inclusion is strict, we say that $\mathcal{C}_1$ is strictly more permissive than $\mathcal{C}_2$.

Indeed, in our settings, it seems more coherent 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 in the controlled system, since the observations are (masked) states of the system and not actions. However, it can be shown :

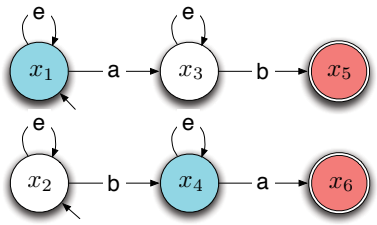

Fig. 2. System without a most permissive controller

Proposition 1: In general, there is no most permissive controller solving Problem 1 or 2.

Proof: We consider the following example to prove this property.

For the LTS of Fig. 2, the set of initial states $Q_0 = \{x_1, x_2\}$ and all transitions are controllable. The set $Bad = \{x_5, x_6\}$ and the mask M is defined as follows:

$$M(x) = \begin{cases} y_1 & \text{if } x \in \{x_1, x_4\} \\ y_2 & \text{if } x \in \{x_2, x_3\} \\ y_3 & \text{if } x \in \{x_5, x_6\} \end{cases}$$

There are three possibilities to avoid the set Bad:

- to forbid the transition a in the observation state y_1: $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}_1}) = \{x_1, x_2, x_4\}$.
- to forbid the transition b in the observation state y_2: $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}_2}) = \{x_1, x_2, x_3\}$.
- to forbid the transitions a and b everywhere: $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}_3}) = \{x_1, x_2\}$.

$\mathcal{C}_1$ and $\mathcal{C}_2$ are both more permissive than $\mathcal{C}_3$, but are not comparable. Thus, there is no most permissive controller. ■

In consequence, we define a maximal solution to Problem 1 or 2 as follows :

Definition 6 (A Maximal Controller): A controller $\mathcal{C}$ solving Problem 1 or 2 is maximal, if there does not exist a strictly more permissive controller $\mathcal{C}'$, which solves this problem.

Unfortunately, we can prove the following property :

Proposition 2: If we restrict the problem in finding a maximal controller $\mathcal{C}$, the basic and non-blocking state avoidance control problems are undecidable.

Proof: Under full observation, the computation of the maximally permissive controller solving the state avoidance control problem is undecidable [15].

The restriction of this problem to the maximal basic state avoidance control problem is trivial using the identity function as mask and proves the undecidability of the second problem.

The restriction of the maximal basic state avoidance control problem to the maximal non-blocking state avoidance control problem is also trivial adding an uncontrollable self-loop on each state and proves the undecidability of the second problem. ■

Hence, our aim is to find solutions that are correct and as close as possible to a maximal solution to be of good practical value. Our experiments will validate our solutions.

IV. Symbolic Computation of the controller

We present a theoretical framework to synthesize a controller which solves Problem 1; we then extend this result to the non-blocking case. From Proposition 2, it is clear that this framework, where no approximation is done, cannot ensure the termination of the computations. In section V, we explain how to obtain an algorithm, based on this framework, which always terminates.

The general idea of the control is to compute, using fixpoint computation, the set $I(Bad)$ of states that can lead to Bad triggering only uncontrollable transitions or that can be blocking after control (for the non-blocking case). Then, based on this set of states, we compute the controller, whose aim is to disable, for each observation $y \in Y$, all the controllable actions that may lead to a state in $I(Bad)$. Our algorithms are symbolic in the sense that they do not enumerate the state space.

A. The basic state avoidance control problem

We describe here a symbolic method to compute a controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$ that solves Problem 1.

Computation of $I(Bad)$: This set of states and more generally $I(.)$ is given by the function $\mathrm{Coreach}_{uc} : 2^{\mathcal{D}_V} \rightarrow 2^{\mathcal{D}_V}$ defined below. This set corresponds to the set of states that lead to Bad firing only uncontrollable transitions.

Classically, we first define the function $\mathrm{Pre}_{uc}(B)$, which computes the set of states from which a state of B is reachable by triggering exactly one uncontrollable transition.

$$\mathrm{Pre}_{uc}(B) = \bigcup_{\delta \in \Delta_{uc}} \mathrm{Pre}(\delta, B) \text{ , where} \tag{1}$$

$$\mathrm{Pre}(\delta, B) = G_\delta \cap A_\delta^{-1}(B) \tag{2}$$

We recall that G_δ is the set of states from which δ can be fired and $A_\delta^{-1}(B)$ is the set of states that lead to B by δ.

Further, $\mathrm{Coreach}_{uc}(Bad)$ is obtained by the computation of the following fixpoint equation:

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

Note that by the Tarski Theorem [20] the limit of the fixpoint $\mathrm{Coreach}_{uc}(Bad)$ actually exists as the function $\mathrm{Coreach}_{uc}$ is monotonic (but may be uncomputable).

Computation of the controller $\mathcal{C}$ and of the controlled system $\mathcal{T}_{/\mathcal{C}}$: We first define a function $\mathcal{F} : \Sigma \times 2^{\mathcal{D}_V} \rightarrow 2^Y$, where for an action $\sigma \in \Sigma$ and a set $B \subseteq \mathcal{D}_V$ of states to forbid, $\mathcal{F}(\sigma, B)$ specifies the set of observation states for which the action σ has to be forbidden, i.e. the set of observations $y \in Y$ such that there exists $\vec{\nu} \in \mathcal{D}_V$ with $M(\vec{\nu}) = y$, from which a transition labelled by σ leads to B.

$$\mathcal{F}(\sigma, B) = \begin{cases} \bigcup_{\delta \in \mathrm{Trans}(\sigma)} M(\mathrm{Pre}(\delta, B) \setminus B) & \text{if } \sigma \in \Sigma_c \\ \varnothing & \text{otherwise} \end{cases} \tag{4}$$

The controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$ is defined as follows:

- the supervisory function $\mathcal{S}$ is:
$$\forall y \in Y,\ \mathcal{S}(y) = \{\sigma \in \Sigma \mid y \in \mathcal{F}(\sigma, I(Bad))\} \tag{5}$$
- the set E is:
$$E = I(Bad) \tag{6}$$

The computation of the function $\mathcal{F}$ is performed offline and, given an observation y, the set $\mathcal{S}(y)$ is computed online with (5), which uses the function $\mathcal{F}$. Since Σ is finite, $\mathcal{S}(y)$ is computable.

The controlled system $\mathcal{T}_{/\mathcal{C}}$ is computed using Def. 4 with the system $\mathcal{T}$ and the controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$ defined as above.

Proposition 3: Given a system $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, a mask $M : \mathcal{D}_V \rightarrow Y$ and a predicate Bad, i.e. a set of forbidden states, the controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$, where $\mathcal{S}$ and E are computed by (5) and (6), solves Problem 1.

Proof: We prove by induction on the length n of the executions that $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}}) \cap I(Bad) = \varnothing$. This implies that $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}}) \cap Bad = \varnothing$.

- Base ($n = 0$): the initial states of the controlled system $\mathcal{T}_{/\mathcal{C}}$ are defined by $\Theta_{/\mathcal{C}} = \Theta \setminus E = \Theta \setminus I(Bad)$. Thus, the execution of $\mathcal{T}_{/\mathcal{C}}$ starts in a state that does not belong to $I(Bad)$.
- Induction: suppose the proposition holds for paths of transitions of length less or equal to n. For paths of length $n+1$, we have by induction hypothesis that each state $\vec{\nu}$ reachable with a path of length n does not belong to $I(Bad)$. We show that no transition $\delta \in \Delta$ can be fired from this state $\vec{\nu} \notin I(Bad)$ to a state $\vec{\nu}' \in I(Bad)$. Indeed :
  - either $\delta \in \Delta_c$, then this transition cannot be fired since $\sigma_\delta \in \mathcal{S}(M(\vec{\nu}))$ by (4) and (5).
  - or $\delta \in \Delta_{uc}$, then $\vec{\nu} \in I(Bad)$, which is impossible by hypothesis.

■

Example 3: For the STS of Fig. 1 and the mask of Example 2, we define Bad as $\{\langle MouseDead, k_1, k_2 \rangle | k_1, k_2 \in \mathbb{N}\}$. The controllable (resp. uncontrollable) transitions are those drawn in plain (resp. dashed) lines. Then, $I(Bad) = \{\langle HopeCheese, k_1, k_2 \rangle | k_1 \in [0, 1000] \wedge k_2 \in \mathbb{N}\} \cup \{\langle Awake, k_1, k_1 \rangle | k_1 \in \mathbb{N}\} \cup \{\langle MouseDead, k_1, k_2 \rangle | k_1, k_2 \in \mathbb{N}\}$.

The computation of $\mathcal{F}$ gives: $\mathcal{F}(wakeup, I(Bad)) = \{\langle Sleep, k_1 \rangle | k_1 \in \mathbb{N}\}$, $\mathcal{F}(up2rooms, I(Bad)) = \{\langle Sleep, k_1 \rangle | k_1 \in [0, 998]\}$ and $\mathcal{F}(\sigma, I(Bad)) = \varnothing$, $\forall \sigma \in \Sigma \setminus \{wakeup, up2rooms\}$.

Then, the supervisory function $\mathcal{S}$ is defined as follows: (i) $\mathcal{S}(y) = \{wakeup, up2rooms\}$, $\forall y \in \{\langle Sleep, k_1 \rangle | k_1 \in [0, 998]\}$, (ii) $\mathcal{S}(y) = \{wakeup\}$, $\forall y \in \{\langle Sleep, k_1 \rangle | k_1 \geq 999\}$ and (iii) $\mathcal{S}(y) = \varnothing$, otherwise. The controlled system is given by Fig. 1, with the guards $G_2 = (x \geq 999)$ and $G_3 = \bot$. ◇

B. The non-blocking state avoidance control problem

We describe here a symbolic method to compute a controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$ that solves Problem 2.

Computation of $I(Bad)$: This set of states and more generally $I(.)$ is given by the function $\mathrm{Coreach}_{uc}^{nb} : 2^{\mathcal{D}_V} \rightarrow 2^{\mathcal{D}_V}$ defined below. This set corresponds to the set of states that would be blocking in the controlled system and of states that lead to a forbidden state firing only uncontrollable transitions.

To compute $\mathrm{Coreach}_{uc}^{nb}(Bad)$, we first compute $\mathrm{Coreach}_{uc}(Bad)$ (defined by (3)). Then, if we make the forbidden states unreachable by cutting all the controllable transitions that lead to a bad state, the corresponding controlled system $\mathcal{T}_{/\mathcal{C}}$ could have new blocking states. We must add these blocking states to the set of forbidden states. The function $\mathrm{Pre}_{bl}(B)$ computes, for a set $B \subseteq \mathcal{D}_V$ of states to forbid, the set of states, that would be blocking in the controlled system, if the states of B were no longer reachable. The computation of the blocking states is based on the function $\mathcal{F}$ defined at (4). To ensure the convergence in the computation of $\mathrm{Coreach}_{uc}^{nb}(Bad)$, Pre_{bl}, and therefore $\mathcal{F}$, must be monotonic. Thus, we use the monotonic function $\hat{\mathcal{F}}$ instead of $\mathcal{F}$ in the computation of the controller for the non-blocking case.

$$\hat{\mathcal{F}}(\sigma, B) = \begin{cases} \bigcup_{\delta \in \mathrm{Trans}(\sigma)} M(\mathrm{Pre}(\delta, B)) & \text{if } \sigma \in \Sigma_c \\ \varnothing & \text{otherwise} \end{cases}$$

Note that $\hat{\mathcal{F}}$ is more restrictive than $\mathcal{F}$ and thus a controller computed w.r.t. $\mathcal{F}$ is more permissive than a controller computed w.r.t. $\hat{\mathcal{F}}$.

We now explain how to compute the blocking states in the controlled system $\mathcal{T}_{/\mathcal{C}}$. A state $\vec{\nu} \in \mathcal{D}_V$ is blocking in $\mathcal{T}_{/\mathcal{C}}$, if the two following conditions are satisfied in the system $\mathcal{T}$:

1) the state $\vec{\nu}$ has no outgoing uncontrollable transition.
2) every outgoing controllable transition δ of $\vec{\nu}$ is forbidden by control in the observation state $M(\vec{\nu})$, i.e. $M(\vec{\nu}) \in \hat{\mathcal{F}}(\sigma_\delta, B)$

Formally, that gives the two following conditions:

1) $\forall \delta \in \Delta_{uc} : \vec{\nu} \notin G_\delta$
2) $\forall \delta \in \Delta_c : (\vec{\nu} \notin G_\delta) \vee (M(\vec{\nu}) \in \hat{\mathcal{F}}(\sigma_\delta, B))$

Because $\hat{\mathcal{F}}(\sigma, B) = \varnothing$ $(\forall \sigma \in \Sigma_{uc})$, the function Pre_{bl} can be expressed as follows:

$$\mathrm{Pre}_{bl}(B) \quad = \quad B \cup \left[\bigcap_{\delta \in \Delta} \left(\overline{G_\delta} \cup (M^{-1}(\hat{\mathcal{F}}(\sigma_\delta, B))) \right) \right]$$

Adding the blocking states to the forbidden states can provide new states leading uncontrollably to a forbidden state. Consequently, to compute the set $\mathrm{Coreach}_{uc}^{nb}(Bad)$, we define the following fixpoint equation:

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

The controller and the controlled system are defined similarly to what is done at the point IV-A.

Proposition 4: Given a system $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$, a mask $M : \mathcal{D}_V \rightarrow Y$ and a predicate Bad, i.e. a set of forbidden states, the controller $\mathcal{C} = \langle \mathcal{S}, E \rangle$, computed according to Def. 3 w.r.t. (7), solves Problem 2.

Proof: Since $\mathrm{Coreach}_{uc}(Bad) \subseteq \mathrm{Coreach}_{uc}^{nb}(Bad)$, it can be proved similarly to the proof of Prop. 3 that Bad is not reachable in this more restrictive controlled system.

Let us suppose that the controlled system does not satisfy the non-blocking property. Then, there exists at least a blocking state $\vec{\nu} \in \mathcal{D}_V$, which is reachable in the controlled system. By definition of the fixpoint (7), $\vec{\nu} \in \mathrm{Coreach}_{uc}^{nb}(Bad)$, and so is any state $\vec{\nu}' \in \mathcal{D}_V$ such that there is a sequence of uncontrollable transitions from $\vec{\nu}'$ to $\vec{\nu}$. According to the above algorithm, $\vec{\nu}$ and $\vec{\nu}'$ are both non reachable. ■

C. Improvement of the control algorithm for finite systems

In [19], the authors define a controller which, to our knowledge, is the most permissive controller satisfying the S-observability condition known in the literature. However, this algorithm is only defined for finite LTS. We show that with this restriction our controller is as good as the one they obtain[1].

Proposition 5: For finite systems, our algorithm solving Problem 1 gives a controller which is as permissive as the one obtained in [19].

Proof: Let us first explain the method given in [19]. The system to control is modelled by a finite LTS $G = \langle X, x_0, \Sigma, \delta \rangle$, where X is the set of states, x_0 is the initial state, Σ is the set of actions and $\delta : \Sigma \times X \rightarrow X$ is the transition relation. The control specification is given by a set Q of allowable states, i.e. $Q = \overline{Bad}$. The partial observation is formalized by a mask $M : X \rightarrow Y$, where Y is the finite observation space. The algorithm of [19] is composed of two steps:

1) computation of $Q^\uparrow \subseteq Q$. $Q^\uparrow = \bigcap_{j=0}^{\infty} Q_j$, where Q_j is recursively defined as follows:

$$Q_j = \begin{cases} Q & \text{if } j = 0 \\ Q \cap (\bigcap_{\sigma \in \Sigma_{uc}} \{x \in X | \\ (\langle \sigma, x \rangle \in \delta) \Rightarrow \delta(\sigma, x) \in Q_{j-1}\}) & \text{otherwise} \end{cases}$$

2) compution of the function A, where $\forall y \in Y : A(Q^\uparrow, y) = \{\sigma \in \Sigma_c | \exists x \in Q^\uparrow : (M(x) = y) \wedge (\langle \sigma, x \rangle \in \delta) \wedge (\delta(\sigma, x) \notin Q^\uparrow)\}$. The forbidden actions in a state $x \in X$ are given by $A(Q^\uparrow, M(x))$.

Note that all the computations terminate in a finite amount of time. In particular, there is an n such that $Q^\uparrow = \bigcap_{j=0}^{n} Q_j$.

We remark that $Q^\uparrow = \overline{\mathrm{Coreach}_{uc}(Bad)}$, because $\forall j \geq 0, \bigcap_{i \leq j} Q_i = \bigcup_{i \leq j} \mathrm{Pre}_{uc}^i(Bad)$, where $\mathrm{Pre}_{uc}^0(Bad)$ denotes Bad and $\mathrm{Pre}_{uc}^i(Bad) = \mathrm{Pre}_{uc}(\mathrm{Pre}_{uc}^{i-1}(Bad))$ $(\forall i > 0)$.

Moreover, to prevent from reaching $\mathrm{Coreach}_{uc}(Bad)$, our function $\mathcal{S}$ is defined by $\mathcal{S}(y) = \{\sigma \in \Sigma_c | \exists x \notin \mathrm{Coreach}_{uc}(Bad), \exists x' \in \mathrm{Coreach}_{uc}(Bad) : (M(x) = y) \wedge (\langle x, \sigma, x' \rangle \in \rightarrow)\}$. Thus, $A(Q^\uparrow, y) = \mathcal{S}(y)$, $\forall y \in Y$. ■

Let us now explain how to improve our algorithm, based on the observations done in the following example.

Example 4: For the LTS of Fig. 3, the set of initial states $X_0 = \{x_1, x_2\}$ and all transitions are controllable. The set

[1]To our knowledge, there is no control algorithm defined for infinite systems with partial observation. For this reason, the comparison is done for the finite case.

$Bad = \{x_5, x_6\}$ and the mask M is defined as follows : (i) $M(x) = y_1$, $\forall x \in \{x_1, x_4, x_7\}$, (ii) $M(x) = y_2$, $\forall x \in \{x_2, x_3\}$ and (iii) $M(x) = y_3$, $\forall x \in \{x_5, x_6\}$.

Our algorithm forbids the transition b in the observation state $M(x_3)$ and the transition a in the observation state $M(x_4)$. However, it is sufficient to forbid b in $M(x_3)$ which makes the state x_4 no longer reachable and thus the controlled system more permissive.

Based on this example, we give an improved algorithm to compute a controller solving Problem 1 for finite systems.

Algorithm 1: Improved algorithm for finite systems

data : An STS $\mathcal{T} = \langle V, \Theta, \Sigma, \Delta \rangle$ such that $[\![\mathcal{T}]\!]$ is finite, a set of states $I(Bad)$ and a mask $M : X \to Y$.
returns: A controller $\mathcal{C}$ such that $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}}) \cap I(Bad) = \varnothing$.

begin
  $\forall y \in Y$, $\mathcal{S}(y) \leftarrow \varnothing$ and $\mathcal{C} \leftarrow \langle \mathcal{S}, I(Bad) \rangle$
  while $\mathsf{reachable}(\mathcal{T}_{/\mathcal{C}}) \cap I(Bad) \neq \varnothing$ **do**
    Let $\vec{\nu} \in ((\mathrm{Pre}_c(I(Bad)) \setminus I(Bad)) \cap \mathsf{reachable}(\mathcal{T}_{/\mathcal{C}}))$ and $\delta \in \Delta_c$ such that $(\vec{\nu} \in G_\delta) \wedge (A_\delta(\vec{\nu}) \in I(Bad))$
    $\mathcal{S}(M(x)) \leftarrow \mathcal{S}(M(x)) \cup \{\sigma_\delta\}$
    $\mathcal{C} \leftarrow \langle \mathcal{S}, I(Bad) \rangle$
  return $(\mathcal{C})$
end

where $\mathrm{Pre}_c(B) = \bigcup_{\delta \in \Delta_c} \mathrm{Pre}(\delta, B)$, for $B \subseteq \mathcal{D}_V$.

The idea of this algorithm is to choose a state $\vec{\nu} \notin I(Bad)$, which is reachable in the current controlled system, and a transition δ leading to $I(Bad)$ from $\vec{\nu}$, and to forbid σ_δ in the observation state $M(\vec{\nu})$. This operation is repeated until the set $I(Bad)$ is no longer reachable in the current controlled system. So, the main difference with the algorithm of section IV is that we verify that a state is still reachable in the current controlled system, before deciding to forbid an action in the corresponding observation state.

Algorithm 1 solves Problem 1 and it outperforms or gives the same result than the one defined in section IV (and thus the one in [19]), but the complexity is greater by a factor $O(|\mathcal{D}_V|.|\Sigma|)$.

V. Effective Computation by Means of Abstract Interpretation

As seen in the previous section, the actual computation of the controller, which is based on a fixpoint equation to compute $I(Bad)$, is generally not possible for undecidability (or complexity) reasons. To overcome the undecidability problem, we use *abstract interpretation* techniques (see e.g. [5], [8], [12]), to compute an over-approximation of the fixpoint $I(Bad)$. This over-approximation ensures that the forbidden states Bad are not reachable in the controlled system, but at the price of forbidding more states than needed. Thus, we obtain a valid controller, but a stricter one.

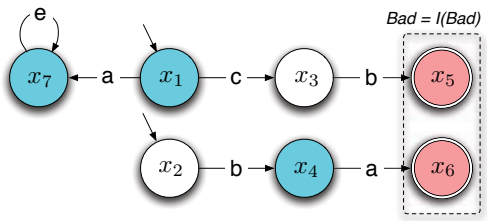

Fig. 3. Improvement of the control algorithm

A. Outline of the abstract interpretation techniques

Abstract interpretation gives a theoretical framework to the approximate solving of fixpoint equations of the form $c = F(c)$, where c is a set of states of the STS:

1) the concrete domain, *i.e.* the sets of states $2^{\mathcal{D}_V}$ is substituted by a simpler (possibly infinite) abstract domain Λ (static approximation), both domains having a lattice structure. The concrete lattice $(2^{\mathcal{D}_V}, \subseteq, \cup, \cap, \varnothing, \mathcal{D}_V)$ and the abstract lattice $(\Lambda, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ are linked by a Galois connection $2^{\mathcal{D}_V} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \Lambda$, which ensures the correctness of the method [5].
2) the fixpoint equation is transposed into the abstract domain. So, the equation to solve has the form: $l = F^\sharp(l)$, with $l \in \Lambda$ and $F^\sharp \sqsupseteq \alpha \circ F \circ \gamma$
3) a widening operator ∇ (dynamic approximation) ensures that the fixpoint computation converges after a finite number of steps to some upper-approximation l_∞.
4) the concretization $c_\infty = \gamma(l_\infty)$ is an over-approximation of the least fixpoint of the function F.

For our experiments, we chose the abstract lattice of *convex polyhedra* [6]. A convex polyhedron on the 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 square triangle.

In this lattice, $\sqcap$ is the classical intersection, $\sqcup$ is the convex hull and $\sqsubseteq$ is the inclusion. The widening operator $P_1 \nabla P_2$ roughly consists in removing from P_1 all the constraints not satisfied by P_2 [6]. In other words, its principle is: if the value of a variable or a linear expression grows between two steps of the fixpoint computation, then one guesses that it can grow indefinitely.

B. Computation of the controller and of the controlled system using abstract interpretation

The function corresponding to $\mathrm{Pre}_{uc} : 2^{\mathcal{D}_V} \mapsto 2^{\mathcal{D}_V}$ is named $\mathrm{Pre}^\sharp_{uc} : \Lambda \mapsto \Lambda$, and is defined in the following way. For $l \in \Lambda$, we have:

$$\mathrm{Pre}^\sharp_{uc}(l) = \bigsqcup_{\delta \in \Delta_{uc}} \mathrm{Pre}^\sharp(\delta, l) \text{ , where} \tag{8}$$

$$\mathrm{Pre}^\sharp(\delta, l) = \alpha(G_\delta \cap A_\delta^{-1}(\gamma(l))) \tag{9}$$

$\mathrm{Coreach}^\sharp_{uc}(Bad)$ is the least fixpoint of the function $\lambda l.\alpha(Bad) \sqcup \mathrm{Pre}^\sharp_{uc}(l)$ and we compute l_∞, defined as the limit of the sequence defined by $l_1 = \alpha(Bad)$ and $l_{i+1} = l_i \nabla \mathrm{Pre}^\sharp_{uc}(l_i)$. The abstract interpretation theory ensures that this sequence stabilizes after a finite number of steps, and that $\gamma(l_\infty)$ is an over-approximation of $I(Bad)$. So we obtain $I'(Bad) = \gamma(l_\infty)$. Finally, we define the controller as in section IV-A, using $I'(Bad)$ instead of $I(Bad)$. We do not detail the effective computation of the other fixpoint, since the same kind of transformations are involved.

Quality of the approximations: The method presented here always computes a safe controller, but without any guarantee that this controller is a maximal one. The approximation of $I(Bad)$ is more precise, if we make less approximations during the computation. Even if a better approximation $I(Bad)$ does not always mean we get a better controller, generally it is the case. There are classical techniques to improve the quality of the approximations:

- the choice of the abstract lattice is the main issue: if it is not adapted to the kind of guards or assignments of the STS, the over-approximations are too rough. The practice shows that if the guards are linear constraints, and if the assignments functions are also linear, the lattice of convex polyhedra [6] works quite well.
- the computation of the fixpoint with the widening operator may be improved by several means: we can use a widening "up to" instead of the standard widening operator [8], we can use one of the fixpoint computation strategies defined in [2] and we can refine our abstract lattice (See [12] for more details).

VI. Implementation and experiments

We implemented the algorithms of sections IV and V. Our tool, named SMACS (Symbolic MAsked Controller Synthesis), is written in Objective CAML [17], and uses the APRON library [1] and a generic fixpoint solver [7]. Its input is a description of the STS, with explicit locations and linear guards and assignments, a description of the property in terms of a set of bad states and a description of the mask, which can be of three kinds defined in section III-A. If no mask is specified, the analysis is performed on a system under full observation. The result of SMACS is a description of the controlled system, written in the same syntax as its input. We experimented our tool on some examples: a toy example, the cat and mouse example, the standard readers and writers example (with n writers are m readers), and a trains example. Those examples are detailed in [13]. In all those examples, there exists a most permissive controller and SMACS finds it in less than 20 ms. In the case of the readers/writers example, we obtain the best solution only when we compute the fixpoint within the lattice of convex polyhedra; with the lattice of intervals, the over-approximations are too rough and we obtain an empty controlled system.

VII. Conclusion and Future Works

We have proposed algorithms for the synthesis of state-feedback controllers through partial observation of infinite state systems modelled by STS. One can notice that our algorithm can be used to verify *safety properties*, because a safety problem can be reduced to a state avoidance control problem (see [16] for details). To our knowledge, the improved version of our algorithm provides a better solution than what was previously proposed in the literature with the hypothesis taken. Our tool SMACS implements our algorithms and allowed us to make an empirical validation of our methods and shows its feasibility and usability. For infinite systems, our algorithms use abstract interpretation techniques that provide an over-approximation of the set $I(Bad)$. Further works will look at possible refinements in the abstract domain to obtain, when needed, more permissive controllers. We will study the synthesis of controllers with memory to provide even more permissive controllers. We also want to study the problem when liveness properties must be fulfilled.

References

[1] The APRON library. http://apron.cri.ensmp.fr/.

[2] F. Bourdoncle. *Sémantiques des Langages Impératifs d'Ordre Supérieur et Interprétation Abstraite*. PhD thesis, Ecole Polytechnique, 1992.

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

[4] K. Chatterjee, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Algorithms for omega-regular games of incomplete information. *Logical Methods in Computer Science*, 3(3:4), 2007.

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

[6] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL '78*, pages 84–96, 1978.

[7] Fixpoint: an OCaml library implementing a generic fix-point engine. http://pop-art.inrialpes.fr/people/bjeannet/bjeannet-forge/fixpoint/.

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

[9] T.A. Henzinger, R. Majumdar, and J.-F. Raskin. A classification of symbolic transition systems. *ACM Trans. Comput. Logic*, 6(1):1–32, 2005.

[10] R.C. Hill, D.M. Tilbury, and S. Lafortune. Covering-based supervisory control of partially observed discrete event systems for state avoidance. In *9th International Workshop on Discrete Event Systems*, May 2008.

[11] L.E. Holloway, B.H. Krogh, and A. Giua. A survey of Petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems: Theory and Application*, 7:151–190, 1997.

[12] B. Jeannet. Dynamic partitioning in linear relation analysis. Application to the verification of reactive systems. *Formal Methods in System Design*, 23(1):5–37, July 2003.

[13] G. Kalyon, T. Le Gall, H. Marchand, and T. Massart. Control of infinite symbolic transition sytems under partial observation. Technical report of the verification group 103, Université Libre de Bruxelles, http://www.ulb.ac.be/di/ssd/gkalyon/publications.html, October 2008.

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

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

[16] T. Le Gall, B. Jeannet, and H. Marchand. Supervisory control of infinite symbolic systems using abstract interpretation. In *CDC/ECC'05*, December 2005.

[17] The programming language Objective CAML. http://caml.inria.fr/.

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

[19] S. Takai and S. Kodama. Characterization of all m-controllable subpredicates of a given predicate. *International Journal of Control*, 70:541–549(9), 10 July 1998.

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

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