Supervisory Control of Infinite Symbolic Systems using Abstract Interpretation

Tristan Le Gall, Bertrand Jeannet, Hervé Marchand

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

Supervisory Control of Infinite Symbolic Systems using Abstract Interpretation

Tristan Le Gall, Bertrand Jeannet and Hervé Marchand
IRISA, Campus Univ. de Beaulieu, 35042 Rennes, France
First.Last@irisa.fr

Abstract— **In this paper, we investigate the control of infinite systems, modeled by symbolic transition system for safety properties. We first redefine the concept of controllability by applying it to the guards of symbolic transitions, instead of to the events. We then define synthesis algorithms based on symbolic transformations and abstract interpretation techniques so that we can ensure finiteness of the computations.**

I. Introduction

In the fields of manufacturing systems, robotics, etc, many applications require high reliability and safety. Traditionally, these requirements are checked *a posteriori* using simulation techniques and/or property verification. Control theory of discrete event systems allows the use of constructive methods that ensure, *a priori*, required properties on the system behavior. There exist different theories for control of Discrete Event Systems since the 80's [19], [2], [11]. Usually, the starting point of these theories is: given a model for the system and the control objectives, a controller must be derived by various means such that the resulting behavior of the closed-loop system meets the control objectives. The basic models are discrete-event systems, and can be formulated as, e.g., formal languages [19], Petri nets [11], or finite state machines, also called labeled transition systems [3].

When modeling realistic systems, it is often convenient to manipulate state/event variables instead of simply atomic states/events. Within this framework, the states (as well as the events) can be seen as a particular instantiation of the vector of variables. In this case, the above techniques and associated tools do not explicitly take into account the data as the underlying model of transition systems implies the variables to be instantiated during the state space exploration and analysis. This enumeration of the possible values of the variables leads to the classical state space explosion when these variables take their value in a finite set, but may also renders the computation infeasible whenever the domain of the variables is infinite. For instance, one can consider timed automata which extend finite transition with clocks and allow transitions upon thresholds and reset operations[18], [1], [20].

In this paper, we model the system to be controlled by Symbolic Transition Systems (STS) which are transition systems with variables. This model allows to represent infinite systems whenever the variables take their values in an infinite domain. This model has a finite structure and offers a compact way to specify systems handling data. Such a model manipulates variables and actions holding parameters that are used to exchange information. The semantics of an STS is given by an infinite labeled transition system (LTS) over an infinite alphabet (action+valued parameters). An STS starts in an initial state and then proceeds by firing transitions, updating the variables according to the guards and assignments of the transitions that have been fired, and exchanging informations through the valued actions.

For control purposes, due to the infiniteness of the alphabet of the underlying LTS, it is very restrictive to keep constant the status of each actions. We thus have chosen to redefine the concept of controllability by applying it to the guards of symbolic transitions, instead of to the events. These guards are predicates on both the variables and the communication parameters and indicate whether the transition becomes uncontrollable or not. To control such systems, we adopt an internal control point of view, which means that the control consists in restricting the guards of the system so that the obtained system satisfies some expected properties. In this setting, the control can be seen as a way to refine an incomplete specification so that it respects requirements. This has to be opposed to the external control which consists in synthesizing a supervisor acting upon the system by disabling events. We here focus on *safety requirements*, modeled by observers that encodes the negation of a safety property. These observers are equipped with a dedicated location *Violate* that is reached whenever the property is violated. It is well known that the underlying control problem (i.e ensuring the safety requirement) can be reduced to a *state avoidance control problem* by performing the product between the system and the observer and by considering as forbidden the set of states of the form $\langle v, \mathit{Violate} \rangle$. This particular problem can then be solved by computing the set of states that leads to the forbidden states by only taking uncontrollable transitions. To deal with infinite-state specifications and properties, the algorithms are *symbolic*: they do not attempt to enumerate the domain of the specifications variables, but deal with the variables by means of symbolic computations. Unfortunately, the exact computation of the above set of states, which is based on a fix-point computation, is generally not possible for undecidability (or complexity) reasons. To overcome the undecidability, we make the use of Abstract Interpretation techniques (see e.g. [5], [7], [14]), that over-approximate this set such that the fix-point can be effectively computed

0-7803-9568-9/05/$20.00 ©2005 IEEE

and symbolically represented.

II. Symbolic Transition Systems

The model of Symbolic Transition Systems (STS) is a transition system with variables. This model allows to represent infinite systems whenever the variables take their values in an infinite domain. Note that in Definition 1, there is no explicit notion of control location, insofar as the control structure of an automaton can be encoded by a specific program counter variable.

Variables, Predicates, Assignments. In the sequel we shall assume a set of typed variables. We denote $\mathcal{D}_v$ the domain in which a variable v takes its values. For a set of variables $V = \{v_1, \ldots, v_n\}$, we denote $\mathcal{D}_V$ the product domain $\mathcal{D}_{v_1} \times \ldots \times \mathcal{D}_{v_n}$. An element of $\mathcal{D}_V$ is thus a vector of values for the variables in V. Depending on the context, a predicate $P(V)$ on a set of variables V may be considered either as a set $P \subseteq \mathcal{D}_V$, or as a logical formula, the semantics of which is a function $\mathcal{D}_V \to \{true, false\}$. We denote by $\bar{E}$ the complement of the set $E \subseteq D$ included in a domain D. Given a predicate $G(\vec{v}, \vec{p})$, the projection over $\vec{p}$ is denoted by $\exists \vec{p} : G(\vec{v}, \vec{p})$ whereas its corresponding set formulation is given by $Proj_{\mathcal{D}_{\vec{v}}}(G) = \{\vec{\nu} \mid \exists \vec{\pi} \in \mathcal{D}_{\vec{p}} : (\vec{\nu}, \vec{\pi}) \in G\}$. Similarly, the universal projection is denoted $\forall \vec{p} : G(\vec{v}, \vec{p})$ and its corresponding set formulation by

$$Proj^{\forall}_{\mathcal{D}_{\vec{v}}}(G) = \{\vec{\nu} \mid \forall \vec{\pi} \in \mathcal{D}_{\vec{p}} : (\vec{\nu}, \vec{\pi}) \in G\}$$

Given a function $f : E \to F$ and $Y \subseteq F$, we denote $f^{-1}(Y) = \{e \in E \mid f(e) \in Y\}$.

STS Definition. Let us now formally define the Symbolic Transition Systems (STS for short).

Definition 1: An STS is a tuple $\langle V, \Theta, Q, \Sigma, \Delta \rangle$ where

- $V = (v_1, \ldots, v_n)$ *is a nonempty, finite set of typed* variables*. We denote* $Q \triangleq \mathcal{D}_V$
- $\Theta \subseteq Q$*, a predicate on* V*, is the* initial condition*.*
- Σ *is a nonempty, finite* alphabet*. For each action* $\sigma \in \Sigma$*, its* signature $sig(\sigma) = \langle t_1, \ldots, t_k \rangle$ *is a tuple of types specifying the types of the communications parameters* [1] *held by the action.*
- Δ *is a finite set of* transitions*. A transition* $\delta = \langle \sigma, p, G, A \rangle$*, also denoted* $[\sigma(\vec{p}) : G(\vec{v}, \vec{p})?\ \vec{v}' = A(\vec{v}, \vec{p})]$ *is given by*
  1) *an action* σ *and a parameter vector* $\vec{p} = \langle p_1, \ldots, p_k \rangle$*, that are variables local to the transition.* $\vec{p}$ *is supposed to be well-typed w.r.t.* $sig(\sigma)$*. We denote* $\mathcal{D}_\sigma \triangleq \mathcal{D}_{\vec{p}}$*.*
  2) *a guard* $G(\vec{v}, \vec{p}) \subseteq Q \times \mathcal{D}_\sigma$*, a predicate on the variables and the communication parameters.*
  3) *an assignment of the form* $\vec{v}' = A(\vec{v}, \vec{p})$*, with* $A : Q \times \mathcal{D}_\sigma \to Q$*, which defines the values of the variables during an execution.*

[1] Communication parameters represent the data communication between two systems modelized by two STS.

Given a transition δ*, we will denote by* σ^δ*,* G^δ *and* A^δ *its action, guard and affectation.* ◇

An STS starts in an initial state with values of the variables satisfying Θ and then proceeds by firing transitions, updating the variables according to the guards and assignments of the transitions that have been fired, and exchanging informations through the valued actions. An STS is *deterministic* whenever for any transitions $\delta_1, \delta_2 \in \Delta$, $\sigma^{\delta_1} = \sigma^{\delta_2} \Rightarrow G^{\delta_1} \wedge G^{\delta_2} = false$. We assume in the sequel that all STS are deterministic.

Example 1: A simple example of STS is depicted in Figure 1. This system expects a START input carrying an integer parameter p, and saves the value of p into the variable x. Then, the value of x is emitted to the environment via the output MSG and x is decreased by 1 or stop when it receives the STOP signal as far as $x > 0$.

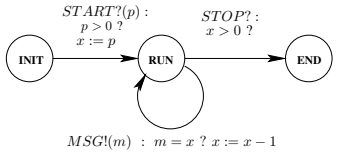

Fig. 1. : simple STS $\mathcal{S}$

Notice that for readability purpose we considered here explicit locations, that can be easily encoded by means of an enumerated variable.

The semantics of an STS $\mathcal{M} = (V, \Theta, \Sigma, \Delta)$ is given by an infinite Labeled Transition System $[[\mathcal{M}]] = (\tilde{\Sigma}, Q, Q_o, \to)$, where $Q = \mathcal{D}_V$. A state $\vec{\nu} \in Q$ is thus a valuation of the variables $\vec{v}$ of the STS. Q_o is a set of states of Q that satisfy the initial condition Θ. $\tilde{\Sigma}$ is an (infinite) alphabet that is formed with valued actions,

$$\tilde{\Sigma} = \{(\sigma, \vec{\pi}) \mid \sigma \in \Sigma \wedge \vec{\pi} \in \mathcal{D}_\sigma\}.$$

The transition relation $\to \subseteq Q \times \tilde{\Sigma} \times Q$ is finally defined by

$$\frac{\delta = [\sigma(\vec{p}) : G(\vec{v}, \vec{p})\ ?\ \vec{v}' = A(\vec{v}, \vec{p})] \in \Delta \quad \vec{\pi} \in \mathcal{D}_\sigma \ \wedge\ G(\vec{\nu}, \vec{\pi}) \ \wedge\ \vec{\nu}' = A(\vec{\nu}, \vec{\pi})}{\vec{\nu} \xrightarrow{\langle \sigma, \vec{\pi} \rangle} \vec{\nu}'}$$

An execution of an STS is given by a run

$$\nu_0 \xrightarrow{(\sigma_0, \vec{\pi}_0)} \nu_1 \ldots \xrightarrow{(\sigma_n, \vec{\pi}_n)} \nu_{n+1}$$

accepted by $[[\mathcal{M}]]$ and a trace of this run is given by $(\sigma_0, \vec{\pi}_0) \ldots (\sigma_n, \vec{\pi}_n)$. We denote by $\mathcal{L}(\mathcal{M}) \subseteq \tilde{\Sigma}^*$ the set of traces associated to $\mathcal{M}$. In the remain of the paper, we will denote abusively by $\vec{v}$ and $\vec{p}$ both the formal variables and their values ν and π.

Blocking property. A state $\vec{v}$ is said to be blocking whenever whatever the transition, its guard is unsatisfiable, i.e. $\forall \delta \in \Delta, \forall \vec{p} \in \mathcal{D}_{\sigma^\delta} : \neg G^\delta(\vec{v}, \vec{p})$. The system, itself is said to be blocking whenever there exists a trace that leads to a blocking state.

Safety properties and observers Given an STS, an *invariance property* is defined by a set of states E and specifies

the fact that from any state in that set, *all* transitions remain in $E(\vec{v})$. Dually, one can define an invariance property by a set of states $E(\vec{v})$ that must not be reached along the execution of the system. This is the latter interpretation that will be used in the remainder of this paper and we will refer to it as a *forbidden state invariance property* $E(\vec{v})$. Now, one can also want to specify more general properties related to the notion of traces/runs of the system. In this setting, a *safety property* P is a set of traces/runs such that

$$\rho \notin P \Rightarrow \forall \rho' : \rho \cdot \rho' \notin P.$$

In other words, as soon as the property is not satisfied then it is false forever. From a computational point of view, it is always possible to reduce a safety property to a forbidden state invariance property by means of *observers*. Given a (deterministic) STS $\mathcal{M}$, an observer is a deterministic STS $\mathcal{O}$ which is *non-intrusive*, i.e. an observer verifies $\mathcal{L}(\mathcal{M} \times \mathcal{O}) = \mathcal{L}(\mathcal{M})$[2]. It is equipped with a dedicated variable Violate. The *language recognized by an observer* $\mathcal{O}$, denoted $\mathcal{L}_{\mathsf{Violate}}(\mathcal{O})$, is the set of the projection of runs that contain states in which the boolean variable Violate is *true* (this set will be denoted by $\mathsf{Violate}^{\mathcal{O}}$). In fact, an observer encodes the negation of a safety property $\varphi_{\mathcal{O}}$ and $\mathcal{L}_{\mathsf{Violate}}(\mathcal{O})$ corresponds to the set of traces that violate $\varphi_{\mathcal{O}}$. Thus, the use of observers allows to reduce a safety property to a forbidden state invariance property that has to be checked on the product $\mathcal{M} \times \mathcal{O}$:

Proposition 1: let $\mathcal{M}$ be a deterministic STS, and $\mathcal{O}$ an observer for $\mathcal{M}$ defining the safety property $\varphi_{\mathcal{O}}$. Then $\mathcal{M}$ satisfies $\varphi_{\mathcal{O}}$ if and only if $\mathcal{M} \times \mathcal{O}$ satisfies the forbidden state invariance property $Q^{\mathcal{M}} \times \mathsf{Violate}^{\mathcal{O}}$. ◇

Based on Proposition 1, we will only consider forbidden state invariance property in the remainder of the paper.

III. Control of Symbolic Transition Systems

A. How to control an STS

In Ramadge & Wonham Theory, the control is performed by means of the notion of controllable event: the alphabet is partitioned into two disjoint sets Σ_c and Σ_{uc} of controllable (resp. uncontrollable) events, and only controllable events can be forbidden by the supervisor. We could do similarly with STS, by partitioning the infinite event set $\tilde{\Sigma}$, either by partitioning the finite action set Σ, or more generally by associating to each $\sigma \in \Sigma$ an uncontrollability guard $c_\sigma(\vec{p})$: the uncontrollability status of an event would thus depend on the values of communication parameters and not only on the action. However, we can be more flexible by allowing the uncontrollability of an event to depend also on the state of the system, as suggested by the following example.

Example 2: Let us consider a plane navigation system, with an automatic navigation module, and an action TurnLeft. *If the automatic module is on (particular state of the system), it is reasonable to consider that* TurnLeft *is uncontrollable, whereas if it is off, the pilot controls itself this event.* ◇

To take into account these modeling needs, we chose to make the uncontrollability notion hold on symbolic transitions instead of events, by equipping each transition δ with a uncontrollability guard $G^{\delta}_{uc}(\vec{v}, \vec{p})$ that indicates in each state $\vec{v}$ whether the event $\sigma^{\delta}(\vec{p})$ is (un)controllable or not. This leads us to the following definition:

Definition 2: An STS to be controlled *is an STS $(V, \Theta, \Sigma, \Delta)$ defined as in Definition 1, in which Δ is a finite set of transitions, such that each transition $\delta = \langle \sigma, p, G, G_{uc}, A \rangle$ has an uncontrollable guard $G_{uc}(\vec{v}, \vec{p}) \subseteq G(\vec{v}, \vec{p})$ which specifies that $\forall \vec{v} \in Q, \forall \vec{p} \in \mathcal{D}_\sigma$:*

$$G_{uc}(\vec{v}, \vec{p}) \Rightarrow \textit{ the event } \sigma(\vec{p}) \textit{ is uncontrollable whenever the system is in state } v$$

Given a state $\vec{v}$ and a set of valued parameters $\vec{p}$, a transition can be fired whenever $G(\vec{v}, \vec{p})$ is satisfied, and it is uncontrollable if the uncontrollable guard $G_{uc}(\vec{v}, \vec{p})$ is satisfied. Thus, according to the value of the variables and the parameters, a supervisor will be able to avoid the transition to be fired only if $G_{uc}(\vec{v}, \vec{p})$ is not satisfied.

Remark 1: If one wants to keep the event partitioning point of view, G_{uc} would only depend on the communication parameters and would be attached to actions instead of symbolic transitions. ◇

In the classical theory of Ramadge & Wonham the controllability status of an action/event is global. In the field of STS, the controllability status of an event $\sigma(\vec{p})$ depends not only on parameters $\vec{p}$ but also on the current state. However, the dependance on symbolic transitions is not a real one, as we consider only deterministic STSs. In this setting, the control will then consist in restricting the guards of the system and the initial condition Θ so that the obtained system satisfy the safety property, with the constraint that the new guards G' satisfy

$$G_{uc} \Rightarrow G' \Rightarrow G \tag{1}$$

B. The State Avoidance Control Problem: An exact characterization

Let $\mathcal{M} = (V, \Theta, \Sigma, \Delta)$ be an STS and $E(\vec{v})$ be a set of states of the associated LTS $[[\mathcal{M}]]$, defined as a predicate that we wish to forbid by control. The principle is similar to the one classically used for the control of Finite Discrete event Systems. Our aim is to modify the initial system so that the controlled system could not enter the states of $E(\vec{v})$. However, it is not possible to perform this transformation on the associated LTS. Hence to perform control, we need to manipulate the states at the syntactic level of the STS. This has to be done by means of symbolic transformer predicates

1) Symbolic transformer predicates: First, we define a function Pre_{uc} which selects the set of states from which it is possible to reach $X(\vec{v}) \subseteq Q$ by triggering uncontrollable transitions. This function is first defined for a particular transition δ (Eq. 2) and thus for all the transitions of the

[2]where $\times$ denotes the classical synchornous product between STS such that $\mathcal{L}(\mathcal{M}_1 \times \mathcal{M}_2) = \mathcal{L}(\mathcal{M}_1) \cap \mathcal{L}(\mathcal{M}_2)$ (see [15] for details).

STS $\mathcal{M}$ (Eq. 3):

$$\mathrm{Pre}_{uc}(\delta)(X)(\vec{v}) = \exists\vec{p}\exists\vec{v}' : G_{uc}(\vec{v},\vec{p}) \wedge \vec{v}' = A(\vec{v},\vec{p}) \wedge X(\vec{v}') \quad (2)$$

$$\mathrm{Pre}_{uc}(X)(\vec{v}) = \bigvee_{\delta\in\Delta} \mathrm{Pre}_{uc}(\delta)(X)(\vec{v}) \quad (3)$$

A set formulation of these functions, for which X is seen as a set of states is the following:

$$\mathrm{Pre}_{uc}(\delta)(X) = \mathit{Proj}_Q\left(G_{uc} \cap A^{-1}(X)\right) \quad (4)$$

$$\mathrm{Pre}_{uc}(X) = \bigcup_{\delta\in\Delta} \mathrm{Pre}_{uc}(\delta)(X) \quad (5)$$

With these operators, it is then possible to compute the set of states that can reach E in an uncontrollable way:

$$\mathrm{Coreach}_{uc}(E) = \mathrm{lfp}(\lambda X.E \cup \mathrm{Pre}_{uc}(X)), \quad (6)$$

which represents the set of states from which it is possible to reach a state of E by only traversing uncontrollable transitions. Moreover, whenever the system is in a state that does not belong to $\mathrm{Coreach}_{uc}(E)$, then it is not possible to reach $\mathrm{Coreach}_{uc}(E)$ via uncontrollable transitions.

2) The control of an STS: Based on these transformer predicates, the control will consist of restricting the guards of the initial system in order to avoid $\mathrm{Coreach}_{uc}(E)$. To this end we introduce an auxiliary function $\mathcal{C}$, which can be seen as a supervisor. It specifies the transitions that have to be forbidden. Note that this function depends on both the current state of the system and the parameters.

$$\mathcal{C}(\delta)(\vec{v},\vec{p}) = \neg\mathrm{Coreach}_{uc}(E)(\vec{v}) \wedge \mathrm{Coreach}_{uc}(E)\left(A^\delta(\vec{v},\vec{p})\right) \quad (7)$$

The corresponding set formulation of 7 is given by:

$$\mathcal{C}(\delta) = (A^\delta)^{-1}(\mathrm{Coreach}_{uc}(E)) \setminus (\mathrm{Coreach}_{uc}(E) \times \mathcal{D}_{\sigma^\delta}) \quad (8)$$

By definition of $\mathrm{Coreach}_{uc}(E)$, we have that, for all transitions δ, $\mathcal{C}(\delta) \Rightarrow \neg G^\delta_{uc}$, which entails that no uncontrollable transition is forbidden.

The controlled system $\mathcal{M}' = (V, \Theta', \Sigma, \Delta')$ where $\Theta' = \Theta \setminus \mathrm{Coreach}_{uc}(E)$ and Δ' is defined from Δ by

$$\frac{(\sigma,\vec{p},G_{uc},G,A) \in \Delta \quad G' = G \wedge \neg\mathcal{C}(\delta)}{(\sigma,\vec{p},G_{uc},G',A) \in \Delta'}$$

It is easy to show that there is no run of $\mathcal{M}'$ that can reach $\mathrm{Coreach}_{uc}(E)$, and thus E. Indeed, the initial states of $\mathcal{M}'$ do not belong to $\mathrm{Coreach}_{uc}(E)$, and no transition of Δ' can make evolve the system from the current state that belongs to $\neg\mathrm{Coreach}_{uc}(E)$ into $\mathrm{Coreach}_{uc}(E)$ (*cf.* Equation (8)). We finally deduce that given an STS $\mathcal{M} = (V, \Theta, \Sigma, \Delta)$ and a forbidden state property E, then there exists a way to control $\mathcal{M}$ w.r.t. E if and only if $\Theta \not\subseteq \mathrm{Coreach}_{uc}(E)$.

3) The blocking property: As usual, when controlling the system, blocking, that were not present in the system, may appear. Assuming the initial system $\mathcal{M}$ is non-blocking, we now explain how the non-blocking property can be ensured on the controlled system $\mathcal{M}'$. The method is actually similar to the one we just presented. The idea is to add to $\mathrm{Pre}_{uc}(X)$ the states that necessarily lead to X whatever the transition. This set denoted Y is given by

$$\begin{aligned}\vec{v} \in Y &\Leftrightarrow \forall\delta\in\Delta, \forall\vec{p}\in\mathcal{D}_\sigma \ : \ G^\delta(\vec{v},\vec{p}) \Rightarrow A^\delta(\vec{v},\vec{p}) \in X \\ &\Leftrightarrow \bigwedge_{\delta\in\Delta} \forall\vec{p} \ : \ G^\delta(\vec{v},\vec{p}) \Rightarrow \left((A^\delta)^{-1}(X)\right)(\vec{v},\vec{p})\end{aligned}$$

Which in turn can be rephrased as follows

$$Y = \bigcap_{\delta\in\Delta} \mathit{Proj}^\forall_Q\left(\overline{G^\delta} \cup (A^\delta)^{-1}(X)\right) \quad (9)$$

If $\vec{v} \in Y$, then it means that

- either there exists an uncontrollable transition starting from $\vec{v}$ that leads to a state of X. In this case $\vec{v} \in \mathrm{Pre}_{uc}(X)$ and this state will not be reachable in the controlled system.
- or there exists a controllable transition starting from $\vec{v}$ that leads to a state of X. Now, as X has to be forbidden by control, then the transition will be removed by control and there will be no more admissible transitions starting from $\vec{v}$ and the system will be in deadlock.

We thus deduce the pre-condition operator

$$\mathrm{Pre}^{nb}_{uc}(X) = \mathrm{Pre}_{uc}(X) \cup \bigcap_{\delta\in\Delta} \mathit{Proj}^\forall_Q\left(\overline{G^\delta} \cup (A^\delta)^{-1}(X)\right) \quad (10)$$

from which we can compute a new set of co-reachable states and thus deduce a non-blocking controlled system.

The technique we developed in the previous section requires to solve a state co-reachability problem (i.e. the computation of $\mathrm{Coreach}_{uc}(E)$). This problem is actually solved by a fix-point computation. Compared to the classical techniques used to control LTS ([19], [3]), the big change induced by taking into account the data and not only the (finite) control of the systems is that the fix-points become not computable[3]. Nevertheless, it is possible to overcome the undecidability by resorting to approximations, using the theoretical framework of Abstract Interpretation. This is the aim of the next section.

C. Effective computation of the controlled system

The idea of the Abstract Interpretation (see e.g. [5], [7], [14]) is then to over-approximate $\mathrm{Coreach}_{uc}(E)$ using techniques such that the fix-point can be computed and symbolically represented. The importants points related to such techniques are the following: assume that we are able to compute such an over-approximation. The main fact is that if $\mathrm{Coreach}_{uc}(E)$ is replaced in the equation (7) by an over-approximation, then the property which says that the forbidden states E are not reachable in the controlled

[3]the reachability problem of a 2-counter machine is not computable [12]

system is preserved. However, it is worthwhile noting that the price to be paid is the maximal permissiveness of the controlled system. Finally, note that if the non-blocking is requested, then the over-approximation leads us to forbid states that would not have been blocking with an exact computation.

It remains to explain how such an over-approximation of $\mathrm{Coreach}_{uc}(E)$ can be computed using abstract interpretation techniques. We first give an outline of the techniques and then show how to apply them to our problem.

a) abstract interpretation techniques: Abstract Interpretation is a theory of approximate solving of fix-point equations. Most analysis problems, among others reachability analysis, come down to solving a fix-point equation of the form $x = F(x), x \in \wp(S)$, where S is the state space of the system, $\wp(S)$ is a complete lattice of sets of states, ordered by inclusion, and F is roughly the "predecessor states" function.

The exact computation of such an equation is generally not possible for undecidability (or complexity) reasons. The fundamental principles of Abstract Interpretation are:

1) to substitute to the concrete domain $\wp(S)$ a simpler abstract domain A (static approximation). The concrete lattice $(\wp(S), \subseteq)$ and abstract lattice $(A, \sqsubseteq)$ are linked by a Galois connection $\wp(S) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$ which ensures the correctness of the method [5].
2) to transpose the fix-point equation into the abstract domain, so that one have to solve an equation

$$y = F^\sharp(y), y \in A, \text{ with } F^\sharp \sqsupseteq \alpha \circ F \circ \gamma$$

3) to use a widening operator (dynamic approximation) to make the iterative computation of the least fix-point of $F^\sharp$ converge after a finite number of steps to some upper-approximation (more precisely, a post-fix-point).

Approximations are conservative so that the obtained result is an upper-approximation of the exact result.

b) Application to the computation of the controlled system: The equation of the fix-point is generally derived in a compositional manner from the syntax of the system to be analyzed, here the STS $\mathcal{M} = (V, \Theta, \Sigma, \Delta)$. Assume that we have an abstract lattice $A(\sqsubseteq, \sqcup, \sqcap, \top, \bot)$ for our abstract state space A, with $\wp(Q) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} A$. Formally we assume that the Galois connection can be extended to $\wp(Q \times \mathcal{D}_\sigma) \leftrightarrows A_\sigma$. The functions of §III-B are rewritten as follows:

$$\begin{aligned} \mathrm{Pre}^\sharp_{uc}(\delta)(Y \in A) &= \alpha\left(Proj_Q(G^\delta_{uc} \cap (A^\delta)^{-1}(\gamma(Y)))\right) \\ \mathrm{Pre}^\sharp_{uc}(Y \in A) &= \bigsqcup_{\delta \in \Delta} \mathrm{Pre}^\sharp_{uc}(\delta)(Y) \end{aligned}$$

and by solving the equation $Y = \alpha(E) \sqcup \mathrm{Pre}^\sharp_{uc}(Y)$, we obtain the fix-point, noted $\mathrm{Coreach}^\sharp_{uc}(E)$. The controller $\mathcal{C}$ and the associated controlled system are thus computed as in the preceding section using $\gamma(\mathrm{Coreach}^\sharp_{uc}(E))$.

Example 3: Assume the variables and parameters of the STS $\mathcal{M}$ are either boolean or real. The state space of the underlying LTS is then of the form $Q = \mathbb{B}^n \times \mathbb{R}^p$. The concrete lattice $\wp(Q) \simeq \mathbb{B}^n \to \wp(\mathbb{R}^p)$ may be abstracted by the lattice $\mathbb{B}^n \to \mathsf{Pol}[p]$ of functions which for each valuation of the boolean variables associates a convex polyhedra of dimension p, as in [7], [9]. A more flexible solution for combining Booleans and reals is proposed in [14], [13]. ◇

c) Example: In order to illustrate the previous section, let us consider the following simple example depicted in Figure 2. This STS has two integer variables x and y and an implicit program counter variable which denotes the locations INIT, RUN et ERR.

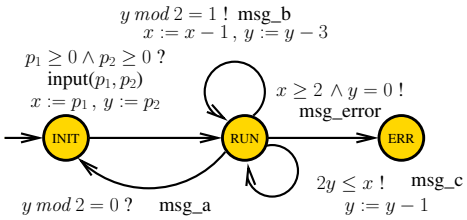

Fig. 2. A toy example

After an initialization phase, x and y can decrease and the STS comes back in its initial state if y is even. The transitions have guards either fully controllable ($G_{uc} = \emptyset$), or fully uncontrollable ($G_{uc} = G$). Controllable guards are represented with a "?" (e.g. INIT $\to$ RUN) and the uncontrollable guards by an "!" (e.g. RUN $\to$ ERR). Assume the forbidden state property is given by $\mathbb{Z}^2 \times \{\mathrm{ERR}\}$. The exact set of x and y values that define, in the RUN location, the states from which the underlying LTS leads to forbidden states by firing an uncontrollable trajectory is

$$\{(x, y) \mid \left\{ \begin{array}{ll} 2y \leq x \wedge x \geq 2 & \text{if } y \text{ is even} \\ 2y \leq x + 5 \wedge x \geq 2 & \text{if } y \text{ is odd} \end{array} \right\}$$

However, the convex hull of this set has states that do not have to be forbidden (e,g the state $x = 3 \wedge y = 2$). Thus, our method based of convex polyhedra, would not lead to an exact computation.

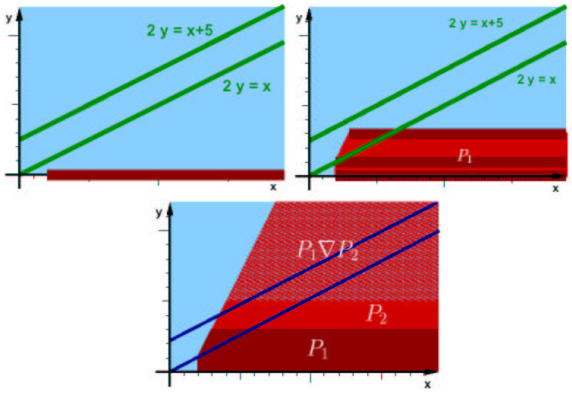

Fig. 3. Set of "bad" states and steps of the fix-point computation

Let us denote by δ_1 and δ_2 the two transitions RUN $\rightarrow$ RUN. At first, we have the set of forbidden states P_0 defined by the conjunction of the constraints $\{y = 0, x \geq 2\}$ (Figs. 3). Using Equation 11, one can compute the set $P_1 = \mathrm{Pre}^\sharp(P_0) = \{0 \leq y \leq 3, x \geq 2, y \leq 2x - 3\}$ (Figs. 3). P_1 is given by the convex hull obtained by considering the 3 semi-lines which represent P_0, $\mathrm{Pre}^\sharp(\delta_1)(P_0)$ and $\mathrm{Pre}^\sharp(\delta_2)(P_0)$. Further, one can compute $P_2 = \mathrm{Pre}^\sharp(P_1) = \{0 \leq y \leq 5, x \geq 2, y \leq 2x-3\}$. $P3$, obtained by widening : $P3 = P_1 \nabla P_2 = \{y \geq 0, x \geq 2, y \leq 2x - 3\}$, is a post-fix-point (Figs. 3). We thus compute the post fix-point in 2 steps, by doing approximations during the convex hull computations as well as widening.

Remark 2 (BRP): We also applied the NBAC *tool [14] to the* Bounded Retransmission Protocol *(BRP) [8], where an emitter and a receiver communicates via 2 unreliable transmission lines. The synthesis problem we were interested in was to compute the conditions on the environment (the transmission lines) for the protocol not to fail. For this, we added 2 counters k and l for counting the number of transmission errors in each line. We synthesized that the global condition we were looking at is (not surprisingly) $k + l < rmax$, where $rmax$ is the maximum number of retransmission performed by the emitter [17].* $\diamond$

d) Quality of Control: The efficiency of an abstract interpretation-based technique depends widely on the quality of the over-approximations. In our case, if our approximations are bad, the controller might be not permissive enough. Thus we have to choose an abstract lattice and a widening operator that are appropriate for the kind of systems we control. In particular, the widening should be applied only when needed [4], and the iterative computation with widening should be followed after convergence by a few decreasing iterations without widening [5]. A standard widening operator can also be improved by using the "widening upto" technique of [7], that exploit information about the system. However, we cannot get the most permissive controller, unless the abstraction and the widening induce no approximation at all, which may happen with simple systems.

IV. Conclusion and related works

In this paper we investigate the control of Symbolic Transitions Systems and we focused on safety properties. In this framework, we discuss how the Ramadge & Wonham theory can be applied to such infinite systems. We first refine the concept of controllability by applying it to the guards of symbolic transitions of the system instead of the events, which is motivated by modeling needs. We then give the equations defining the set of forbidden states, and remind why a naive iterative method may not terminate in general. We thus introduce a method, based on abstract interpretation, that guarantees the convergence of the iteration by over-approximating the set of bad states. Some other works have been done concerning symbolic synthesis of supervisory controllers (e.g. [10], [21], [1], [16]). However, those works either did not deal with infinite systems with an infinite alphabet, or focus on specific classes of infinite systems for which the problem remain decidable. We shall also mention the works of [6], who studied the control of a model similar to STS using a notion of partial priority between events.

References

[1] K. Altisen, G. Gossler, J. Sifakis. Scheduler modelling based on the controller synthesis paradigm. *Journal of real-Time Systems*, 23(1,2):55–84, 2002.

[2] S. Balemi, G. J. Hoffmann, H. Wong-Toi, and G. F. Franklin. Supervisory control of a rapid thermal multiprocessor. *IEEE Transactions on Automatic Control*, 38(7):1040–1059, July 1993.

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

[4] F. Bourdoncle. Efficient chaotic iteration strategies with widenings. In *Proc. of Formal Methods in Programming and their Applications*, *LNCS* 735, 1993.

[5] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *4th ACM Symposium on Principles of Programming Languages, POPL'77*, Los Angeles, January 1977.

[6] G. Goessler and J. Sifakis. Priority systems. In *Proc of FMCO'03*, pages 314–329, Leiden, The Netherlands, November 2003.

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

[8] L. Helmink, M. Sellink, and F. Vaandrager. Proof-checking a data link protocol. In *Proc. of Int. Workshop TYPES'93*, *LNCS* 806, 1993.

[9] T. Henzinger, P.-H. Ho, and H. Wong-Toï. HyTech: A Model Checker for Hybrid Systems. In *Computer Aided Verification, CAV'97*, LNCS 1254, June 1997.

[10] G. Hoffman and H. Wong-Toi. Symbolic synthesis of supervisory controllers. In *Proceedings of the American Control Conference*, Chicago,IL, June 1992.

[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] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[13] B. Jeannet. Representing and approximating transfer functions in abstract interpretation of hetereogeneous datatypes. In *Static Analysis Symposium, SAS'02*, LNCS N^o 2477, Madrid (Spain)), 2002.

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

[15] B. Jeannet, T. Jéron, V. Rusu, E. Zinovieva. Symbolic Test Selection based on Approximate Analysis. In *11th Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05)* LNCS 3340, 2005.

[16] R. Kumar and V. K. Garg. On Computation of State Avoidance Control for Infinite State Systems in Assignment Program Framework. *IEEE Transactions on Automation Science and Engineering*,2(1): 87-91,2005

[17] T. Le Gall, B. Jeannet, and H. Marchand. Control of discrete and hybrid symbolic systems (in french). Technical Report 1683, IRISA, January 2005.

[18] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *12th Annual Symposium on Theoretical Aspects of Computer Science*, volume 900 of *lncs*, pages 229–242, Munich, Germany, March 1995. Springer.

[19] 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.

[20] S. Tripakis and K. Altisen. On-the-fly controller synthesis for discrete and dense-time systems. In *World Congress on Formal Methods (FM'99)*, volume 1708 of *Lecture Notes in Computer Science*, pages 233–252, Toulouse, France, September 1999. Springer Verlag.

[21] R. Ziller. Finding bad states during symbolic supervisor synthesis. In *GI/ITG/GMM Workshop: Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen*, pages 209–218, Tübingen, Germany, 25-27 February 2002.