Supervisory Control Problem using Symbolic Bisimulation Techniques

Hervé Marchand, Sophie Pinchinat
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex, France
e-mail: {hmarchan, pinchina}@irisa.fr

Abstract

In this paper, we present methods for solving the basic Supervisory Control Problem (SCP) using algorithms based on bisimulation techniques. [1] first presented the relations between bisimulation and controllability and provided algorithms for solving the SCP. We efficiently solve the same problem using Intensional Labeled Transition System (ILTS), an implicit representation of automaton, relying on algebraic methods.

keywords : Discrete Event Systems, ILTS, Supervisory Control Problem, Symbolic Bisimulation.

1 Introduction

In the Ramadge and Wonham theory of discrete events systems [2], the behaviors of systems are modeled by sequences of events over a finite alphabet. The plant is represented by some kind of automaton, a labeled transition system (LTS), that generates sequences of events (or actions) through its execution. The control of the plant is then performed by inhibiting some events in Σ_c, the set of controllable events as opposed to the set Σ_{uc} (*uc* stand for uncontrollable) of events that cannot be prevented from occurring.

In [1], the authors solved the Supervisory Control Problem by exploiting the relation between a given bisimulation and the controllability of a language. They provide algorithms on the computation of a bisimulation relating the plant and the desired behavior specification. However, their implementations are based on an extensional representation of the models. This makes the supervisory controller synthesis not practical because of the huge size of the states space when dealing with realistic applications. The proposed approach provides higher-level algorithms, relying on implicit representations (of the plant, of the specification,...). To do so, we use polynomials over a finite field. This way, instead of enumerating states, transitions, etc. and considering them explicitly, we manipulate the polynomial functions characterizing their sets of membership.

The paper is organized as follows : Section 2 recalls the results of [1]. Section 3 explains our approach, and its application to the SCP is described in Section 4. We finally conclude and address perspectives in Section 5.

2 The Supervisory Control Problem

Assuming that the plant to be controlled is modeled as a Labeled Transition System (LTS) that is a 4-tuple $G = \langle \mathcal{X}, \Sigma, x_0, \delta \rangle$, where $\mathcal{X}$ is the (finite) set of states, Σ is the set of events, $x_0 \in \mathcal{X}$ is the initial state, and δ is the partial transition function from $\mathcal{X} \times \Sigma$ to $\mathcal{X}$. In the sequel, we simply write $x \in G$ instead of $x \in \mathcal{X}$.

The behavior of the system is denoted by the prefix-closed language $\mathcal{L}(G)$ [2], generated by G[1]. Some of the events in Σ are uncontrollable, i.e., their occurrence cannot be prevented by a controller, while the others are controllable. In this regard, Σ is partitioned as $\Sigma = \Sigma_c \cup \Sigma_{uc}$, where Σ_c represents the set of controllable events and Σ_{uc} the set of uncontrollable events.

2.1 Basic Results

We now present some preliminaries for the Supervisory Control Problem that can be found in [2]. We first recall the definition of controllability :

Definition 1 *Let H and G be two LTSs, s.t. $\mathcal{L}(H) \subseteq \mathcal{L}(G)$. $\mathcal{L}(H)$ is said to be* controllable *w.r.t. $\mathcal{L}(G)$ and Σ_{uc} if $\mathcal{L}(H)\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \mathcal{L}(H)$.* •

In other words, $\mathcal{L}(H)$ is conditionally invariant w.r.t. $\mathcal{L}(G)$ under events of Σ_{uc} [2].

In the Ramadge and Wonham framework, a supervisor is given by a function $S : \mathcal{L}(G) \rightarrow \{\gamma \in 2^{\Sigma}; \Sigma_{uc} \subseteq \gamma\}$, delivering the set of actions that are allowed in G by the control after a trajectory $s \in \mathcal{L}(G)$. Write S/G for the closed loop system, consisting of the initial plant G controlled by the supervisor S.

Supervisory Control Problem [2]: *Given a plant modeled by an LTS $G = \langle \mathcal{X}, \Sigma, x_0, \delta \rangle$, $\Sigma_{uc} \subseteq \Sigma$ and a*

[1] In the following, all the languages will be assumed to be prefix-closed.

desired language $K \subseteq \mathcal{L}(G)$, The problem is to build a supervisor S such that $\mathcal{L}(S/G) \subseteq K$ is controllable, and for any other supervisor S' s.t. $\mathcal{L}(S'/G) \subseteq K$, $\mathcal{L}(S'/G) \subseteq \mathcal{L}(S/G)$.

In the sequel, we will be more interested in the computation of S/G rather than in the computation of the supervisor S itself, since one can easily extract S from S/G.

The solution of the SCP is classically called the most permissive solution : using $K^{\uparrow}$ to denote the supremal controllable sub-language of K w.r.t. $\mathcal{L}(G)$ and Σ_{uc}, we have $\mathcal{L}(S/G) = K^{\uparrow}$ (see [2]). The standard algorithm used to compute this supremal language is an iterative algorithm starting with the automaton product $H \times G$, with $\mathcal{L}(H) = K$. The iterative procedure consists of (i) removing states that violate the controllability condition, and (ii) removing states that are not reachable [2].

Definition 2 *Assume given a plant $G = \langle \mathcal{X}, \Sigma, x_0, \delta \rangle$ and let a desired behavior be modeled by the LTS H. We denote by $[H \times G]^{\uparrow}$ the LTS obtained by using the standard algorithm [2] to compute the supremal controllable sub-language of $\mathcal{L}(H)$ w.r.t. $\mathcal{L}(G)$ and Σ_{uc}. Hence $\mathcal{L}(H)^{\uparrow} = \mathcal{L}([H \times G]^{\uparrow})$* $\bullet$

We now present another algorithm [1] relying on a relation between the controllability and a bisimulation equivalence approach.

2.2 Solution of the SCP using Bisimulation

We present now the Σ'-bisimulation, where Σ' is some subset of Σ ; it reduces to strong bisimulation [3] when $\Sigma' = \Sigma$.

Definition 3 *[1, 4] Given two LTSs $H = \langle \mathcal{X}_H, \Sigma, x_{H_0}, \delta_H \rangle$ and $G = \langle \mathcal{X}_G, \Sigma, x_{G_0}, \delta_G \rangle$, and $\Sigma' \subseteq \Sigma$. A Σ'-bisimulation of H and G is a binary relation $\psi \subseteq \mathcal{X}_H \times \mathcal{X}_G$ s.t. $(x_H, x_G) \in \psi$ whenever*

1. *for all $\sigma \in \Sigma'$, for all transition $x'_H = \delta_H(x_H, \sigma)$ there exists a state x'_G s.t. $x'_G = \delta_G(x_G, \sigma)$ and $(x'_H, x'_G) \in \psi$.*
2. *vice-versa* $\bullet$

Σ'-bisimulations of H and G are closed under arbitrary unions [1]. So there exists a greatest one, noted $\underline{\leftrightarrow}_{\Sigma'}$. We now formalize the link between Σ'-bisimulation and controllability :

Theorem 1 **[1]** *Let H and G be as in Definition 2 with $\mathcal{L}(H) \subseteq \mathcal{L}(G)$. Denote by $S^{\uparrow}$ the accessible state space of $[H \times G]^{\uparrow}$. Let $\underline{\leftrightarrow}_{\Sigma_{uc}}$ be the greatest Σ_{uc}-bisimulation of H and G, then :*

1. *$(x_{H_0}, x_{G_0}) \in S^{\uparrow}$ iff $x_{H_0} \underline{\leftrightarrow}_{\Sigma_{uc}} x_{G_0}$, and*
2. *$(x_H, x_G) \in S^{\uparrow}$ iff $x_H \underline{\leftrightarrow}_{\Sigma_{uc}} x_G$ and (x_H, x_G) is reachable from (x_{H_0}, x_{G_0}) by a sequence of state transitions that never leave $\underline{\leftrightarrow}_{\Sigma_{uc}}$.* $\diamond$

From this theorem, we can easily derive an algorithm providing $[H \times G]^{\uparrow}$. It first computes the Σ_{uc}-bisimulation $\underline{\leftrightarrow}_{\Sigma_{uc}}$ of the H and G (using the algorithm of [5]) and next builds some product model from H and G according to the $\underline{\leftrightarrow}_{\Sigma_{uc}}$ relation. So the following Algorithm [1] :

Algorithm 1

1. *Compute $\underline{\leftrightarrow}_{\Sigma_{uc}}$ the greatest Σ_{uc}-bisimulation of H and G*
2. *If $x_{H_0} \not\underline{\leftrightarrow}_{\Sigma_{uc}} x_{G_0}$, then let R be the empty LTS. Otherwise, let $R = (\mathcal{X}_R, \Sigma, x_{0_R}, \delta_R)$, where :*

$$\mathcal{X}_R = \{(x_H, x_G) \in \mathcal{X}_H \times \mathcal{X}_G\}$$
$$x_{R_0} = (x_{H_0}, x_{G_0})$$

$\forall \sigma \in \Sigma$:

$$\delta_R((x_H, x_G), \sigma) = \begin{cases} (\delta_H(x_H, \sigma), \delta_G(x_G, \sigma)) \text{ if} \\ \quad \delta_H(x_H, \sigma) \underline{\leftrightarrow}_{\Sigma_{uc}} \delta_G(x_G, \sigma) \\ \mathit{Undefined\ otherwise.} \end{cases}$$

3. *Let $R^{\uparrow}$ denotes the accessible LTS of R*

Theorem 2 **[1]** $\mathcal{L}(R^{\uparrow}) = \mathcal{L}(H)^{\uparrow} = \mathcal{L}([H \times G]^{\uparrow})$. $\diamond$

3 Intensional models

This section introduces the models, named the Intensional Labeled Systems (ILTS)[6] upon which symbolic computations will be performed.

3.1 The Mathematical Framework

In the following, we write $\mathbb{Z}/p\mathbb{Z}$ for the finite field $\{0, 1, \ldots, p-1\}$, with p prime. Let Z be a finite set of k distinct variables $Z_1, ..., Z_k$. We denote by $\mathbb{Z}/p\mathbb{Z}[Z]$ the set of polynomials over variables $Z_1, ..., Z_k$ which coefficients range over $\mathbb{Z}/p\mathbb{Z}$. We recall that $(\mathbb{Z}/p\mathbb{Z}[Z], +, *)$ is a ring. Given a polynomial $P(Z) \in \mathbb{Z}/p\mathbb{Z}[Z]$, we associate its set of solutions $Sol(P) \subseteq (\mathbb{Z}/p\mathbb{Z})^k$:

$$Sol(P) \stackrel{\text{def}}{=} \{(z_1, ..., z_k) \in (\mathbb{Z}/p\mathbb{Z})^k | P(z_1, ..., z_k) = 0\} \quad (1)$$

It is worthwhile noting that in $\mathbb{Z}/p\mathbb{Z}[Z]$, $Z_1^p - Z_1, ..., Z_k^p - Z_k$ evaluate to zero. Then for any $P(Z) \in \mathbb{Z}/p\mathbb{Z}[Z]$, one for instance has $Sol(P) = Sol(P + (Z_i^p - Z_i))$. Write $P_1 \equiv P_2$ whenever $Sol(P_1) = Sol(P_2)$. We then introduce the quotient ring of polynomial functions $A[Z] = \mathbb{Z}/p\mathbb{Z}[Z]/_{<Z^p - Z>}$, where all polynomials $Z_i^p - Z_i$ are identified to zero, written for short $Z^p - Z$. $A[Z]$ can be regarded as the set of polynomial functions with coefficients in $\mathbb{Z}/p\mathbb{Z}$ for which the degree in each variable is lower than $(p-1)$. This will be very useful for algorithmic purposes. It also possible to define a representative of $Sol(P)$ (i.e. $[P]_{\equiv}$), called the *canonical generator*.

At this point, we would like to mention some useful properties to manipulate polynomials, namely :
Property 1 *For all polynomials $P_1, P_2, P \in \mathbb{Z}/_{p\mathbb{Z}}[Z]$, $Sol(P_1) \subseteq Sol(P_2)$ whenever $(1-P_1^{p-1}) * P_2 \equiv 0$. Moreover, by defining $P_1 \oplus P_2 \stackrel{def}{=} (P_1^{p-1}+P_2^{p-1})^{p-1}$, we have, $Sol(P1) \cap Sol(P_2) = Sol(P_1 \oplus P_2)$, $Sol(P_1) \cup Sol(P_2) = Sol(P_1 * P_2)$, and $(\mathbb{Z}/_{p\mathbb{Z}})^k \setminus Sol(P) = Sol(1-P^{p-1})$.* $\diamond$

Finally, we introduce the *existential/universal abstractions* (or quantifications) over polynomials w.r.t. some variables. Let $P \in \mathbb{Z}/_{p\mathbb{Z}}[Z]$, we shall write $\exists Z_i P$ for the polynomial $P|_{Z_i=0} * P|_{Z_i=1} * \ldots * P|_{Z_i=p-1}$, where $P|_{Z_i=v}$ is P obtained by instantiating any occurrence of variable Z_i by value v. Similarly, we define a dual variable abstraction over polynomials, based on universal quantification : $\forall Z_i P$ is computed as $P|_{Z_i=0} \oplus P|_{Z_i=1} \oplus \ldots \oplus P|_{Z_i=p-1}$ which solutions are elements of the form $(z_1, ..., z_{i-1}, z_{i+1}, ..., z_k)$ s.t. $\forall z_i, (z_1, ..., z_{i-1}, z_i, z_{i+1}, ..., z_k) \in Sol(P)$.

Notes on the implementation. It turns out that the best known implementation (for memory and computation performance) of polynomials over finite field (i.e. $\mathbb{Z}/_{p\mathbb{Z}}$) is based on their decomposition according to the Lagrange polynomials, leading to p-ary decision diagrams data structures. A classical instance of this approach is the well-known Shannon decomposition for the case $p=2$, with associated Binary Decision Diagrams (BDD) for the boolean functions [7].

3.2 Intensional labeled transition systems

Definition 4 *An* m-dimensional intensionally Labeled Transition System (or m-iLTS) *is a structure $T = (\mathcal{X}, Y, \mathcal{I}, \rightarrow)$, where*

- *$\mathcal{X}$ is a set of states, $\mathcal{I} \subseteq \mathcal{X}$ is a set of initial states,*
- *Y is a set of m variables $Y_1, ..., Y_m$, and*
- *$\rightarrow \subseteq \mathcal{X} \times \mathbb{Z}/_{p\mathbb{Z}}[Y] \times \mathcal{X}$. Each transition is labeled by a polynomial over the set Y.* $\bullet$

We write $q \stackrel{P(Y)}{\rightarrow} q'$ (or simply $q \stackrel{P}{\rightarrow} q'$), instead of $(q, P(Y), q') \in \rightarrow$. Then, iLTS can be understood as an "intensional" representation of classical LTSs, labeled by tuples of $(\mathbb{Z}/_{p\mathbb{Z}})^m$: each arrow of the iLTS labeled by $P(Y)$ intensionally represents as many arrows labeled by some $y \in Sol(P(Y))$. In the sequel, we shall call $Ext(T)$ the possibly non-deterministic "extensional" LTS, underlying the m-iLTS T.

Intensional approach for labels offers a "compact" way to talk about sets of transitions in the system. However, we would like to reinforce this approach in such a way that the whole system, and not only its sets of labels, can be itself described intensionally. Intuitively, the transition relation will be given by a polynomial, thus generalizing the iLTS approach.

Definition 5 *An* (n,m)-dimensional Intensional Labeled Transition System (or (n,m)-ILTS) *is a structure $S = (X, X', Y, I, \mathcal{T})$ where $X = \{X_1, ..., X_n\}$ and $X' = \{X'_1, ..., X'_n\}$ are two sets of* (source and target) states variables, *$Y = \{Y_1, ..., Y_m\}$ is a set of labels variables, I is a polynomial in $\mathbb{Z}/_{p\mathbb{Z}}[X]$ characterizing initial states and $\mathcal{T}(X, Y, X') \in \mathbb{Z}/_{p\mathbb{Z}}[X, Y, X']$ is a polynomial describing the legal transitions.* $\bullet$

Given some source state $x \in (\mathbb{Z}/_{p\mathbb{Z}})^n$ and some target state $x' \in (\mathbb{Z}/_{p\mathbb{Z}})^n$, the set $Sol(\mathcal{T}(x, Y, x'))$ denotes all the possible labels of transitions from state x to state x'. When states are viewed extensionally, we retrieve the models of Definition 4.

3.3 Σ'-Bisimulation in the ILTS framework

Let us introduce the notion of *symbolic A-Bisimulation*, where A is a polynomial that will characterize the set of events Σ' of Definition 3.

Definition 6 *Consider two m-iLTSs $T_1 = (\mathcal{X}_1, Y, \mathcal{I}_1, \rightarrow_1)$, $T_2 = (\mathcal{X}_2, Y, \mathcal{I}_2, \rightarrow_2)$, and a polynomial $A \in \mathbb{Z}/_{p\mathbb{Z}}[Y]$. A symbolic* A-bisimulation *of T_1 and T_2 is a binary relation $\mathcal{R} \subseteq \mathcal{X}_1 \times \mathcal{X}_2$ s.t. $(x_1, x_2) \in \mathcal{R}$ (or $x_1 \mathcal{R} x_2$) whenever*

1. *for all $x_1 \stackrel{P}{\rightarrow}_1 x'_1$, there exists a finite set of transitions $(x_2 \stackrel{P_i}{\rightarrow}_2 x_2^i)_{i \in I}$, such that :*
   (a) *$(1 - (P \oplus A)) * \Pi_i P_i = 0$, and*
   (b) *$x'_1 \mathcal{R} x_2^i$, for all $i \in I$.*
2. *vice versa.* $\bullet$

In Definition 6, the clause $(1-(P \oplus A)) * \Pi_i P_i = 0$ can be reinterpreted as $Sol(P) \cap Sol(A) \subseteq \bigcup_i Sol(P_i)$, that is any transition from x_1 to x'_1 satisfying the "criterion" denoted by A can be mimicked from x_2. Since A-bisimulations are closed under arbitrary unions, we can talk about the greatest A-bisimulation, written $\preceq_A$ in the following, and say that "x_1 and x_2 are A-bisimilar" whenever $x_1 \preceq_A x_2$.

Symbolic A-bisimulation between iLTSs corresponds to $Sol(A)$-bisimulation (in the sense of Definition 3) between the underlying extensional LTSs :

Theorem 3 (Expressiveness) *Let T_1 and T_2 be two iLTSs. $\mathcal{R}$ is a symbolic A-bisimulation between T_1 and T_2 iff $\mathcal{R}$ is a $Sol(A)$-bisimulation between $Ext(T_1)$ and $Ext(T_2)$.*

Proof : $\Rightarrow$) Let $\mathcal{R}$ be a symbolic A-bisimulation between T_1 and T_2. Let $x_1 \mathcal{R} x_2$ and let $x_1 \xrightarrow{y}_1 x_1'$ in $Ext(T_1)$, with $y \in Sol(A)$. Then there exists $x_1 \xrightarrow{P}_1 x_1'$ with $P(y) = 0$ and $A(y) = 0$. By definition of $\mathcal{R}$, there exists some indexes i such that $x_2 \xrightarrow{P_i} x_2^i$, $(1 - P \oplus A) * \Pi_i P_i = 0$ and $x_1' \mathcal{R} x_2^i$. Because $P(y) = 0$ and $(1 - P \oplus A) * \Pi_i P_i = 0$ when applied to y, $P_i(y) = 0$ for some i which proves that $x_2 \xrightarrow{y}_2 x_2^i$, and we are done. Transition $x_2 \xrightarrow{y}_2 x_2'$ is dealt similarly.

$\Leftarrow$) Let $\mathcal{R}$ be a $Sol(A)$-bisimulation of $Ext(T_1)$ and $Ext(T_2)$. Let $x_1 \mathcal{R} x_2$ and let $x_1 \xrightarrow{P}_1 x_1'$ in T_1. As $x_1 \mathcal{R} x_2$, for each $y_0 \in Sol(A) \cap Sol(P)$, there exists $x_2 \xrightarrow{y_0}_2 x_2^{y_0}$ in $Ext(T_2)$ with $x_1' \mathcal{R} x_2^{y_0}$. Then, in T_2 there exists some P^{y_0} s.t. $x_2 \xrightarrow{P^{y_0}}_2 x_2^{y_0}$ and $P^{y_0}(y_0) = 0$. Consider the polynomial $\Pi_{y \in Sol(P \oplus A)} P^y(Y)$. Clearly by construction, $Sol(P \oplus A) \subseteq Sol(\Pi_{y \in Sol(P \oplus A)} P^y)$, which entails $(1 - P \oplus A) * \Pi_{y \in Sol(P \oplus A)} P^y(Y) \equiv 0$. This shows that the x_2^y are the good candidates, which concludes the proof $\diamond$

Assume now given two ILTSs $S_1 = (X_1, X_1', Y, I_1, \mathcal{T}_1)$, $S_2 = (X_2, X_2', Y, I_2, \mathcal{T}_2)$, and $A \in \mathbb{Z}/_{p\mathbb{Z}}[Y]$. Algorithm 2 computes the greatest symbolic A-bisimulation of S_1 and S_2.

Algorithm 2

1. *Define the polynomial* $\psi_0(X_1, X_2) \stackrel{def}{=} 0$.
2. *Compute iteratively until stabilization* $(\psi_k(X_1, X_2))_k$ *defined by :*

$$\left\{ \begin{array}{r} \psi_{k+1}(X_1, X_2) \text{ is the canonical generator of} \\ \psi_k(X_1, X_2) \oplus \forall X_1' \forall Y[(1 - \mathcal{T}_1(X_1, Y, X_1') \oplus A(Y)) * \\ \exists X_2'(\mathcal{T}_2(X_2, Y, X_2') \oplus \psi_k(X_1', X_2'))] \\ \oplus \forall X_2' \forall Y[(1 - \mathcal{T}_2(X_2, Y, X_2') \oplus A(Y)) * \\ \exists X_1'(\mathcal{T}_1(X_1, Y, X_1') \oplus \psi_k(X_1', X_2'))] \end{array} \right.$$

3. *Call* $\psi_A(X_1, X_2)$ *the result.*

Theorem 4 *With the preceding notations,*

$$\psi_A(x_1, x_2) = 0 \Leftrightarrow x_1 \preceq_A x_2 \Leftrightarrow x_1 \underline{\leftrightarrow}_{Sol(A)} x_2. \qquad \diamond$$

The proof is similar to the one of Theorem 2.12 of [6].

4 Application to the SCP

In this section, we present the application of the techniques developed in the previous section to solve the SCP. The non-blocking aspect of the SCP is here discard, but would also fit within this framework.

4.1 Event-controlled ILTS

In order to match Section 2, we need to introduce the notion of Event-controlled ILTS, which are assumed to be deterministic[2]. So the definition :

[2] i.e. $\forall x, y,\ Sol(T(x, y, X'))$ as at most one element

Definition 7 *An Event-controlled ILTS is given by a deterministic (n, m)-ILTS and two polynomials P_c and P_{uc} of $\mathbb{Z}/_{p\mathbb{Z}}[Y]$ s.t.*

$$Sol(P_c) \cap Sol(P_{uc}) = \emptyset \qquad (2)$$
$$Sol(P_c) \cup Sol(P_{uc}) \subseteq (\mathbb{Z}/_{p\mathbb{Z}})^m \qquad (3)$$

$\bullet$

For such an Event-controlled ILTS and with the preceding notations, $Sol(P_c)$ (resp. $Sol(P_{uc})$) corresponds to the set of controllable (resp. uncontrollable) events as in Section 2, i.e. $Sol(P_c) = \Sigma_c$ and $Sol(P_{uc}) = \Sigma_{uc}$. Note that Equation (2) ensures that an event is either controllable or uncontrollable.

4.2 The Desired Behavior

Let $T = (X, X', Y, I, \mathcal{T}, P_{uc}, P_c)$ be an Event-controlled *ILTS* modeling the plant. Let $\mathcal{O}(X, Y, X')$ be a polynomial in $\mathbb{Z}/_{p\mathbb{Z}}[X, Y, X']$, called a *control objective* on T. Consider now the Event-controlled (n,m)-ILTS $T_{\mathcal{O}} = (X, X', Y, I, \mathcal{T} \oplus \mathcal{O}, P_{uc}, P_c)$. $T_{\mathcal{O}}$ models the desired behavior. Intuitively, we only consider in the explicit LTS $Ext(T)$ the set of states/events/transitions that satisfy $\mathcal{O}(X, Y, X') = 0$.

$\mathcal{O}$ is used to express control objectives as predicates over the states and/or the events: for example, if we want to ensure a *safety property* over a particular set of states of the plant (say $Sol(\alpha(X))$), the control objectives $\mathcal{O}$ will be defined as $\mathcal{O}(X, X') = \alpha(X) \oplus \alpha(X')$. In other words, the supervisor, we want to compute with this particular polynomial, has to ensure the invariance of the set of states $Sol(\alpha(X))$. Some other control objectives can also be considered : $\mathcal{O}$ can result from a previous SCP computation (e.g. *attractivity, reachability, optimal control* [8]).

4.3 The supervisor Computation

Assume we have two Event-controlled ILTSs T and $T_{\mathcal{O}}$ as defined in the previous section. First of all, remark that $T_{\mathcal{O}}$ is the initial system T restricted by polynomial $\mathcal{O}$. Hence the proposition:

Proposition 1 $\mathcal{L}(Ext(T_{\mathcal{O}})) \subseteq \mathcal{L}(Ext(T))$ $\diamond$

That is to say that the behavior of $T_{\mathcal{O}}$ is a sub-behavior of T. In the sequel we rename the states variables (source and target) of T and $T_{\mathcal{O}}$ such that $T = (X_1, X_1', Y, I, \mathcal{T})$ and $T_{\mathcal{O}} = (X_2, X_2', Y, I_{\mathcal{O}}, \mathcal{T}_{\mathcal{O}})$. Let $\psi_{P_{uc}}$ be the result of Algorithm 2 applied on T and $T_{\mathcal{O}}$, i.e. the greatest symbolic P_{uc}-bisimulation of T and $T_{\mathcal{O}}$

Build now the ILTS $T_C = (X_C, X_C', Y_C, I_C, \mathcal{T}_C)$, where

- $X_C = X_1 \cup X_2$, $X_C' = X_1' \cup X_2'$, and $Y_C = Y$

- $I_C(X_C) = I(X_1) \oplus I_{\mathcal{O}}(X_2) \oplus \psi_{P_{uc}}(X_1, X_2)$
- $\mathcal{T}_C(X_C, Y, X'_C) = \mathcal{T}(X_1, Y, X'_1) \oplus \mathcal{T}_{\mathcal{O}}(X_2, Y, X'_2) \oplus \psi_{P_{uc}}(X'_1, X'_2)$

By Algorithm 1, we have to consider the accessible set of states of the system T_C. To do so, we compute the orbit $\mathcal{O}rb$ of the system, as follows:

$$\begin{cases} \mathcal{O}rb_0(X_C) = I_C(X_C) \\ \mathcal{O}rb_{i+1}(X_C) = \Delta(\exists X_C((\exists Y \mathcal{T}_C(X_C, Y, X'_C)) \oplus \mathcal{O}rb_i(X_C))) \end{cases}$$

where Δ is a function which renames the variable X' in X. It is then sufficient to intersect the transition relation $\mathcal{T}_C$ with the orbit to obtain a new relation, where all the states are accessible.

$$\mathcal{T}'_C(X_C, Y, X'_C) = \mathcal{T}_C(X_C, Y, X'_C) \oplus \mathcal{O}rb(X'_C)$$

With the preceding notations, we can state the following theorem:

Theorem 5 *If $Sol(I_C)$ is non empty, then the ILTS $T'_C = (X_C, X'_C, Y, I_C, \mathcal{T}'_C)$ generates the supremal controllable sub-language of $\mathcal{L}(Ext(T_{\mathcal{O}}))$ w.r.t. $\mathcal{L}(Ext(T))$ and $Sol(P_{uc})$.*

Proof : First by Assumption T and $T_{\mathcal{O}}$ are deterministic and by Proposition 1, the behavior of $T_{\mathcal{O}}$ is included in the one of T. By Theorem 2, we have to show that $Ext(T'_C)$ is $R^{\uparrow}$ of Algorithm 1 : consider (x_1, x_2, y), s.t. $\mathcal{T}'_C(x_1, x_2, y, x'_1, x'_2) = 0$ for some x'_1 and x'_2. Because $\mathcal{T}(x_1, y, x'_1) = 0$, $\mathcal{T}_{\mathcal{O}}(x_2, y, x'_2) = 0$ and $\psi_{P_{uc}}(x'_1, x'_2) = 0$, $(x_1, x_2) \xrightarrow{y} (x'_1, x'_2)$ holds in $Ext(T'_C)$ whenever $x_1 \xrightarrow{y} x'_1$, $x_2 \xrightarrow{y} x'_2$ and $x'_1 \underline{\leftrightarrow}_{Sol(P_{uc})} x'_2$ (Theorem 4). We exactly retrieve the definition of the transition function δ_R of Algorithm 1 which, by construction of T'_C is restricted to accessible states. $\diamond$

5 Conclusion

In this paper, we have presented an alternative approach for the SCP using bisimulation techniques inspired from [1], relying on symbolic/implicit representation of automaton, named Intensional Labeled Transition System (ILTS). These models lead to more abstract algorithms based on algebraic transformations, for which efficient implementation can for instance rely on Decision Diagrams techniques. From the application point of view, automatic generation of ILTS is available in the synchronous language SIGNAL compiler[8]. It performs the boolean abstraction of any SIGNAL program according to the following interpretation : a signal variable is associated to a variable in $\mathbb{Z}/_{3\mathbb{Z}}$, where values 0, 1, -1 respectively mean *absence, presence* with *trueness* and *presence* with *falseness*[9]. Symbolic bisimulation algorithms have been implemented for this case in the formal proof system SIGALI, available in the SIGNAL plateform[6], but the algorithms would bear any extension in $\mathbb{Z}/_{p\mathbb{Z}}$, with p prime as shown in this paper. In particular, for the case $p = 2$, existing BDD packages can be efficiently used for boolean function operations.

Finally, for the SCP, we foresee several possible extensions : we are currently working on a reduction of the controlled plant, by mixing symbolic strong bisimulation (for the reduction) and symbolic P_{uc}-bisimulation (for the control synthesis) algorithms. Some other extension of the SCP can be considered : for example splitting structured events in a controllable part and an uncontrollable part could enable us to tackle more realistic applications. Also, the controllability property of events can be "local", e.g. depending on the current state of the plant . This way degraded modes can be introduced. For all the above mentioned extensions, an algebraic solution can be proposed.

References

[1] G. Barrett and S. Lafortune, "Bisimulation, the supervisory control problem and strong model matching for finite state machines," *Discrete Event Dynamic Systems*, vol. 8, no. 4, pp. 337–429, December 1998.

[2] 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*, vol. 77, no. 1, pp. 81–98, 1989.

[3] D. Park, "Concurrency and automata on infinite sequences," in *Proc. 5th GI Conf. on Th. Comp. Sci., LNCS 104*. Mar. 1981, pp. 167–183, Springer-Verlag.

[4] J. C. M. Baeten and W. P. Weijland, *Process Algebra*, vol. 18 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge Univ. Press, 1990.

[5] J.-C. Fernandez, "An implementation of an efficient algorithm for bisimulation equivalence," *Science of Computer Programming*, vol. 13, pp. 219–236, May 1989.

[6] O. Kouchnarenko and S. Pinchinat, "Intensional approachs for symbolic methods," *Electronic Notes in Theoretical Computer Science*, vol. 18, 1998.

[7] R.E. Bryant, "Graph-based algorithms for boolean function manipulations," *IEEE Transaction on Computers*, vol. C-45, no. 8, pp. 677–691, August 1986.

[8] H. Marchand, P. Bournai, M. Le Borgne, and P. Le Guernic, "A design environment for discrete-event controllers based on the Signal language," in *1998 IEEE International Conf. On Systems, Man, And Cybernetics*, San Diego, California, USA, October 1998, pp. 770–775.

[9] M. Le Borgne, H. Marchand, E. Rutten, and M. Samaan, "Formal verification of signal programs: Application to a power transformer station controller," in *Proceedings of AMAST'96*, Munich, Germany, July 1996, vol. 1101 of *Lecture Notes in Computer Science*, pp. 271–285, Springer-Verlag.