Optimal control of discrete-time hybrid automata under safety and liveness constraints

Carla Seatzu[a] Dmitry Gromov[b] Jörg Raisch[b, c]
Daniele Corona[a] Alessandro Giua[a]

[a]*Dip. di Ing. Elettrica ed Elettronica, Università di Cagliari, Italy*
Email: {daniele.corona,seatzu, giua}@diee.unica.it

[b]*Lehrstuhl für Systemtheorie technischer Prozesse, Otto-von-Guericke-Universität Magdeburg, Germany*
Email: {dmitry.gromov, joerg.raisch}@e-technik.uni-magdeburg.de

[c]*Systems and Control Theory Group, Max-Planck-Institut für Dynamik komplexer technischer Systeme, Magdeburg*

Abstract

In this contribution we address an optimal control problem for a class of discrete-time hybrid automata under safety and liveness constraints. The solution is based on a hierarchical decomposition of the problem, where the low-level controller enforces safety and liveness constraints while the high-level controller exploits the remaining degrees of freedom for performance optimization. Lower-level control is based on a discrete abstraction of the continuous dynamics. The action of low-level control can be interpreted as restricting invariants in the hybrid automaton representing the plant model. A state feedback solution for the high-level control is provided, based on the off-line construction of an appropriate partition of the state space. ©2006 Elsevier Ltd. All rights reserved.

Key words: Hybrid automata; switched systems; safety constraints; liveness constraints

[1] In press on *Nonlinear Analysis*, 2006. Work partially done in the framework of the HYCON Network of Excellence, contract number FP6-IST-511368

1 Introduction

Hybrid automata are dynamic systems that consist of both continuous dynamics (modeled by a set of differential or difference equations) and a switching scheme (modeled by invariants and guards). Hybrid automata (and other modeling paradigms for hybrid systems) have been widely investigated because of their importance in many application areas. Often, the control objective for such systems is to minimize a cost function while respecting safety and liveness constraints. There are a number of abstraction-based control synthesis approaches that address safety and liveness issues while largely ignoring performance optimization aspects [8, 19, 16, 11]. On the other hand, very interesting papers on the optimal control of hybrid systems have been presented, but the proposed approaches are often not able to handle "hard" safety constraints. Among these we mention [4, 10, 14, 18, 21, 20, 22, 24, 25].

In this paper we provide a method for synthesizing a state feedback control strategy which minimizes a given cost function under certain safety and liveness constraints.

More precisely, the contribution of this paper is threefold and can be summarized in the following items.

- Firstly, we investigate how the action of an approximation based discrete supervisor can be interpreted as restricting invariants of the hybrid automaton plant model. This has been briefly described in the conference papers [7, 9]. In this contribution, we provide a much more detailed exposition.
- Secondly, we extend our previous results on the optimal control and stabilization of switched systems [6] and hybrid automata [4, 7, 5] where a technique was presented to solve an infinite time horizon optimal control problem for an hybrid automaton whose continuous dynamics are affine, when a quadratic performance index is considered.

  Here we generalize our previous results in two ways:

  (a) we take into account the existence of forbidden regions assuming that the invariant set inv_i of a location $i \in L$ may be a *proper* subset of $\mathbb{R}^n$, i.e., $inv_i \subsetneq \mathbb{R}^n$;

  (b) we assume that an infinite number of switches is allowed.

  We show that a state feedback solution based on the off-line construction of an appropriate switching region, that we call *switching table*, can be computed. Each point of the table uniquely determines the corresponding optimal mode.
- The third contribution consists in combining the above approaches in order to deal with the optimal control of discrete-time hybrid automata under safety and liveness constraints. More precisely, the problem is divided in two hierarchical levels. The low-level controller enforces safety and liveness

constraints, and can be interpreted as restricting invariants in the hybrid automaton representing the plant model. The high-level control uses the remaining degrees of freedom to perform optimization.

The paper is structured as follows. In Section 2, we recall some basic facts on hybrid automata, introduce the plant model and formalize the specifications. In Section 3, the safety and liveness requirements are addressed using ℓ-complete abstraction of the continuous plant dynamics. In Section 4, the remaining degrees of freedom are used to minimize a quadratic cost function. In Section 5, a numerical example is provided.

Finally, a remark regarding terminology. As time, i.e. the domain of signals, is discrete throughout this paper, the words "continuous" and "discrete" will always refer to the range of signals: continuous signals live in dense subsets of some Euclidean space, whereas discrete signals live in discrete, and for the purpose of this paper, finite sets; continuous (respectively discrete) systems are characterized by continuous (respectively discrete) signals.

2 Plant Model and Specifications

In this section we first define the class of Hybrid Automata (HA) on which we focus attention. Then we formally describe the safety specifications and the optimal control problem.

2.1 Hybrid Automata

Like a continuous-time hybrid automaton [12, 1], a discrete-time hybrid automaton $\mathcal{H}A$ consists of a "classic" automaton extended with a continuous state. The latter, denoted by $x(k) \in \mathbb{R}^n$, evolves in discrete time $k \in \mathbb{N}_0$ with arbitrary dynamics. The hybrid automaton considered here is a structure $\mathcal{H}A = (L, X, f, inv, E)$ which, in complete analogy to, e.g. [2], is defined as follows:

- $L = \{1, \dots, \alpha\}$ is a finite set of locations.
- $X \subseteq \mathbb{R}^n$ is a continuous state space.
- $f_i : X \to X$ is a function that associates to each location $i \in L$ a discrete time difference equation of the form

$$x(k+1) = f_i(x(k)). \tag{1}$$

- $inv : L \to 2^X$ is a function that associates to each location $i \in L$ an invariant $inv_i \subseteq X$.

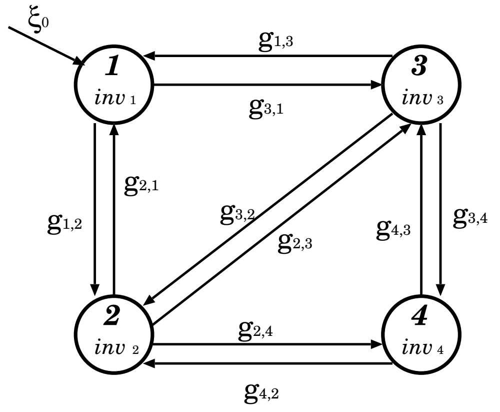

Fig. 1. A graph describing an hybrid automaton.

- $E \subset L \times 2^X \times L$ is the set of edges. An edge $e_{i,j} = (i, g_{ij}, j) \in E$ is an arc between locations i and j with associated guard region g_{ij}. The set of edges can be interpreted as the discrete part of the overall hybrid automaton as shown in Fig.1. This is a directed graph with vertices corresponding to the locations (state-graph). The state-graph is assumed to be connected.

The pair $(l(k), x(k))$ represents the hybrid state at time k, where $l(k)$ is the discrete location $l(k) \in L$ and $x(k) \in \mathbb{R}^n$ is the continuous state.

Starting from initial state $\xi_0 = (i, x_0) \in L \times X$, $x_0 \in inv_i$, the continuous state x may evolve according to the corresponding discrete-time transition function f_i, i.e., $x(k+1) = f_i(x(k))$, until it is about to leave the invariant inv_i, i.e. $f_i(x(k)) \notin inv_i,\ k \in \mathbb{N}_0$. This enforces a switch to another location j satisfying the guard constraint $x(k) \in g_{ij}$ and $x(k) \in inv_j$; the future evolution of the continuous state is now determined by the transition function f_j, provided that the condition $f_j(x(k)) \in inv_j$ holds. If several potential "follow-up" locations satisfy the constraint, this degree of freedom can be exploited by an appropriate discrete control scheme. Thus, the sequence $l(k)$ of discrete locations can be interpreted as a constrained control input. Note that the hybrid automaton may also switch to a "new" location j *before* being forced to leave its "old" location i, if the corresponding guard constraint and "new" invariant are satisfied.

It may also happen that for some state $(l(k), x(k))$ the system evolution cannot be extended to the interval $[k+1, \infty)$. This situation is referred as *blocking*. The notion of *liveness*, in turn, corresponds to the fact that the system evolution can always be extended to infinity. In this paper the liveness property of interest for the considered system is assumed to equivalent to nonblocking. This is described formally in Section 2.3. Obviously, the initial hybrid automaton does not necessarily possess the liveness property. A first goal of low-level control is to assure this property along with safety specifications.

2.2 The Plant Model

In this paper we assume that the uncontrolled plant is modeled as a specific discrete-time hybrid automaton satisfying the following assumptions:

A1. The difference equation (1) is linear and time-invariant, i.e.

$$x(k+1) = A_i x(k) \quad \forall i \in L, k \in \mathbb{N}_0. \tag{2}$$

A2. $inv_i = X \quad \forall i \in L$.
A3. Transitions between any two locations $i, j \in L$ are allowed.
A4. $g_{ij} = X \quad \forall i, j \in L$.

Note that assumption A3 does not reduce the generality of the approach, since possible restrictions on discrete transitions can be considered as specifications – this will be illustrated in Sec. 3.2.1. Under the assumptions A1 - A4, the uncontrolled plant is a switched linear system with a free input signal $l : \mathbb{N}_0 \to L$.

It will turn out in Section 3.3 that adding low-level control to the plant model will add nontrivial invariants to the plant automaton. This may be interpreted as adding state space constraints that force the plant dynamics to respect safety and liveness constraints.

2.3 Safety and Liveness Specification

To formalize *safety* specifications, the continuous plant state space X is partitioned via a function $q_y : X \to Y_d$, where Y_d is a finite set of symbols. To express dynamic safety constraints, certain sequences of input/output symbols are declared illegal or, in other words, the evolution of the hybrid automaton needs to be restricted such that only legal (L, Y_d)-sequences are generated. It is assumed that this set of sequences can be realized by a finite automaton P. The procedure of "building" such an automaton is described in detail in Sec. 3.2.1

The *liveness* requirement implies that $\forall\ i\ \in\ L$, $\forall\ k\ \in \mathbb{N}_0$, the following must hold: $x(k) \in inv_i$, $f_i(x(k)) \notin inv_i \Rightarrow \exists e = (i, g_{ij}, j)$, $x(k)\ \in\ g_{ij}$ and $x(k+1) = f_j(x(k))\ \in\ inv_j$. Note that the liveness condition guarantees the existence of an evolution $(i(k), x(k))$, $k \in \mathbb{N}_0$, from every initial hybrid state (i, x_0).

2.4 Optimal Control Problem

Subject to plant model, safety and liveness constraints, we aim at minimizing the cost function

$$J = \sum_{k=0}^{\infty} x(k)' Q_{l(k)} x(k), \tag{3}$$

where, for each $k \in \mathbb{N}_0$ and $l(k) \in L$, $Q_{l(k)}$ is a positive semidefinite real matrix.

This problem will now be approached using a two-level control hierarchy. Safety and liveness requirements are being taken care of by the low-level control. This is described in Section 3. The remaining degrees of freedom are used to minimize the cost function (3). This is described in Section 4.

3 The low-level task

In a first step, the hybrid plant automaton is approximated by a finite state machine employing the ℓ-complete approximation approach [16, 11]. Subsequently, Ramadge and Wonham's supervisory control theory [e.g. 17] is used to synthesize a least restrictive supervisor. Note that, in general, controller synthesis and approximation refinement are iterated until a nontrivial supervisor guaranteeing liveness and safety for the approximation can be computed or computational resources are exhausted. In the former case, attaching the resulting supervisor to the hybrid plant model amounts to introducing restricted invariants. The resulting hybrid automaton represents the plant under low-level control and can be guaranteed to respect both safety and liveness constraints.

3.1 Ordered set of discrete abstractions

The low-level control deals with a continuous system (2) with discrete external signals. $l : \mathbb{N}_0 \to L$ is the discrete control input and $y_d : \mathbb{N}_0 \to Y_d$ the discrete measurement signal generated by

$$y_d(k) = q_y(x(k)). \tag{4}$$

The set of output symbols, Y_d, is assumed to be finite: $Y_d = \{y_d^{(1)}, \ldots, y_d^{(\beta)}\}$, and $q_y : X \to Y_d$ is the output map. Without loss of generality, the latter is supposed to be surjective (*onto*). The output map partitions the state space into a set of disjoint subsets $Y^{(i)} \subset X,\ i = 1, \dots, \beta$, i.e.

$$\bigcup_{i=1}^{\beta} Y^{(i)} = X,$$

$$Y^{(i)} \cap Y^{(j)} = \emptyset \quad \forall i \neq j\,.$$

To implement supervisory control theory, the hybrid plant model is approximated by a purely discrete one. This is done using the method of ℓ-complete approximation [16, 11], which is described in the following paragraphs.

Denote the external behavior of the hybrid plant model by $\mathcal{B}_{plant}$, i.e. $\mathcal{B}_{plant} \subseteq (L \times Y_d)^{\mathbb{N}_0}$ is the set of all pairs of (discrete valued) input/output signals $w = (l, y_d)$ that (2) and (4) admit. In general, a time-invariant system with behavior $\mathcal{B}$ is called ℓ-complete if

$$w \in \mathcal{B} \Leftrightarrow \sigma^k w|_{[0,\ell]} \triangleq w|_{[k,k+\ell]} \in \mathcal{B}|_{[0,\ell]}\ \forall k \in \mathbb{N}_0,$$

where σ is the unit shift operator and $w|_{[0,\ell]}$ denotes the restriction of the signal w to the domain $[0, \ell]$ [23]. For ℓ-complete systems we can decide whether a signal belongs to the system behavior by looking at intervals of length ℓ. Clearly, an ℓ-complete system can be represented by a difference equation in its external variables with lag ℓ. Hence, an ℓ-complete system a finite range for its external signals can be realized by a finite state machine. However, the hybrid plant model $\mathcal{B}_{plant}$ is, except for trivial cases, not ℓ-complete. For such systems, the notion of *strongest ℓ-complete approximation* has been introduced in [11]: a time-invariant dynamical system with behavior $\mathcal{B}_\ell$ is called strongest ℓ-complete approximation for $\mathcal{B}_{plant}$ if

(i) $\quad \mathcal{B}_\ell \supseteq \mathcal{B}_{plant}$,

(ii) $\quad \mathcal{B}_\ell$ is ℓ-complete,

(iii) $\quad \mathcal{B}_\ell \subseteq \tilde{\mathcal{B}}_\ell$ for any other ℓ-complete $\tilde{\mathcal{B}}_\ell \supseteq \mathcal{B}_{plant}$,

i.e. if it is the "smallest" ℓ-complete behavior containing $\mathcal{B}_{plant}$. Obviously, $\mathcal{B}_\ell \supseteq \mathcal{B}_{\ell+1}\ \forall \ell \in \mathbb{N}$, hence the proposed approximation procedure may generate an ordered set of abstractions. Clearly, $w \in \mathcal{B}_\ell \Leftrightarrow w|_{[0,\ell]} \in \mathcal{B}_{plant}|_{[0,\ell]}$. For $w|_{[0,\ell]} = (l_0, \dots, l_\ell, y_d^{(i_0)}, \dots, y_d^{(i_\ell)})$ this is equivalent to

$$\begin{aligned} &f_{l_{\ell-1}}\left(\dots f_{l_1}\left(f_{l_0}\left(q_y^{-1}(y_d^{(i_0)})\right) \cap \left(q_y^{-1}(y_d^{(i_1)})\right)\right) \dots (q_y^{-1}(y_d^{(i_{\ell-1})})\right) \cap q_y^{-1}(y_d^{(i_\ell)}) \\ &\quad \triangleq X(w|_{[0,\ell]}) \neq \emptyset. \end{aligned} \tag{5}$$

Note that for a given string $w|_{[0,\ell]}$, $X(w|_{[0,\ell]})$ represents the set of possible values for the continuous state variable $x(\ell)$ if the system has responded to the input string $l(0) = l_0, \ldots, l(\ell-1) = l_{\ell-1}$ with the output $y_d(0) = y_d^{(i_0)}, \ldots, y_d(\ell) = y_d^{(i_\ell)}$. Note also that (5) does not depend on $l(\ell)$. For linear and affine systems evolving on discrete time $\mathbb{N}_0$, (5) can be checked *exactly*, as all involved sets are polyhedra.

As both input and output signal evolve on finite sets L and Y_d, $\mathcal{B}_\ell$ can be realized by a (nondeterministic) finite automaton. In [16, 11], a particularly intuitive realization is suggested, where the approximation state variable stores information on past values of l and y_d. More precisely, the automaton state set can be defined as

$$X_d := \bigcup_{j=0}^{\ell-1} X_{d_j}, \ \ell \geq 1,$$

where $X_{d_0} = Y_d$, and X_{d_j} is the set of all strings $(l_0, \ldots, l_{j-1}, y_d^{(i_0)}, \ldots, y_d^{(i_j)})$ such that

$$(l_0, \ldots, l_j, y_d^{(i_0)}, \ldots, y_d^{(i_j)}) \in \mathcal{B}|_{[0,j]}.$$

The temporal evolution of the automaton can be illustrated as follows:
From initial state $x_d(0) \in X_{d0}$, it evolves through states

$$x_d(j) \in X_{dj}, \ 1 \leq j \leq \ell - 1$$

while

$$x_d(j) \in X_{d_{\ell-1}}, \ j \geq \ell - 1.$$

Hence, until time $\ell - 1$, the approximation automaton state is a complete record of the system's past and present, while from then onwards, it contains only information on the "recent" past and present.

As the states $x_d^{(i)} \in X_d$ of the approximation realization are strings of input and output symbols, we can associate $x_d^{(i)}$ with a set of continuous states, $X(x_d^{(i)})$, in completely the same way as in (5).

Note that we can associate $y_d^{(i_k)}$ as the unique output for each discrete state $x_d(k) = (l_{k-j}, \ldots, l_{k-1}, y_d^{(i_{k-j})}, \ldots, y_d^{(i_k)}) \in X_d, \ j < \ell$. Thus, the output is just the last symbol in the symbolic description of the state. It is then a straightforward exercise to provide a transition function $\delta : X_d \times L \to 2^{X_d}$ such that the resulting (non-deterministic) Moore-automaton $M_\ell = (X_d, L, Y_d, \delta, \mu, X_{d_0})$ with state set X_d, input set L, output set Y_d, output function $\mu : X_d \to Y_d$, and initial state set X_{d_0} is a realization of $\mathcal{B}_\ell$. Note that the state of M_ℓ is instantly deducible from observed variables [15].

To recover the framework of supervisory control theory [e.g. 17] as closely as

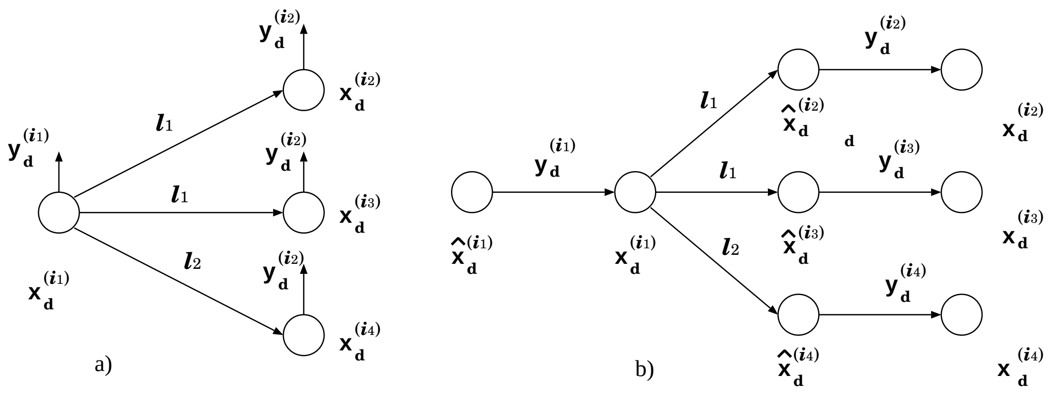

Fig. 2. Moore-automaton (left) and an equivalent automaton without outputs (right). Note that $y_d^{(i_k)} = \mu(x_d^{(i_j)}) \in Y_d$ is the output symbol associated with the discrete state $x_d^{(i)}$.

possible, we finally convert M_ℓ into an equivalent automaton without outputs, $G_\ell = (\tilde{X}_d, \Sigma, \tilde{\delta}, \tilde{X}_{d_0})$, where $\Sigma = L \cup Y_d$, L represents the set of controllable events and Y_d the set of uncontrollable events.

Technically, this procedure is carried out according to the following scheme (for an illustration, see Fig.2):

- Each state $x_d^{(j)} \in X_d$ is split into two states: $x_d^{(j)}$ and $\hat{x}_d^{(j)}$. Thus, the new state set is formed as $\tilde{X}_d = X_d \bigcup \hat{X}_d$. Initial states are replaced by their complements, $\tilde{X}_{d_0} = \hat{X}_{d_0}$.
- The new transition function $\tilde{\delta}$ is defined as a union of two transition functions with nonintersecting domains:

$$\tilde{\delta}(\tilde{x}_d^{(i)}, \sigma^{(j)}) = \begin{cases} \backsim\tilde{x}_d^{(i)}, & \tilde{x}_d^{(i)} \in \hat{X}_d,\ \sigma^{(j)} = \mu(\tilde{x}_d^{(i)}) \in Y_d, \\ \delta(\tilde{x}_d^{(i)}, \sigma^{(j)}), & \tilde{x}_d^{(i)} \in X_d,\ \sigma^{(j)} \in L, \\ \text{undefined}\ , & \text{otherwise}, \end{cases}$$

  where $\backsim$ denotes an operation of taking the complementary state, i.e. $\backsim\hat{x}_d^{(i)} \triangleq x_d^{(i)}$ and vice versa. Note that the first event always belongs to the set Y_d, and the following evolution consists of sequences where events from L and Y_d alternate.

Note that since the function μ is scalar-valued, the set of feasible events for each state $x \in \hat{X}_d$ contains only one element. Thus, any two states x and $\backsim x$ form a fixed pair, where the states $\hat{X}_d$ are in some sense fictitious and play an auxiliary role.

3.2 Specifications and supervisor design

3.2.1 Formal specifications

In the following, we consider specifications, which consist of the independent specifications for the input and the output, respectively.

The specification on outputs expresses both static constraints (through restricting the set of allowed outputs $Y_d^* \subset Y_d$) and dynamic constraints. Dynamic constraints are usually represented as a set of forbidden strings such as, e.g., "the symbol $y_d^{(j)}$ follows immediately upon the symbol $y_d^{(i)}$", or "the symbol $y_d^{(i)}$ appears three times one after another without any other symbol in between". The set of all allowed strings is then realized as a finite automaton $P_Y = (S_Y, Y_d, \delta_Y, s_{Y0})$.

The specifications for inputs, in turn, reflect structural restrictions on the allowed sequence of input symbols. They can be extracted from the state-graph in Fig.1 and realized by a finite automaton $P_L = (S_L, L, \delta_L, s_{L0})$ according to the algorithm shown in Fig.3.

```
Initialize: $S_L = \{1, \ldots, |L|\}$,
            $\xi_0 = (i, x_0)$ : $s_{L0} = \{i\}$      % initial location
For $j, k = 1$ to $|L|$,  $j \neq k$
    If $(j, \cdot, k) \in E$
        $\delta_L(j, k) = k$
    End If
    $\delta_L(j, j) = j$
End For
```

Fig. 3. Transformation of state-graph to a finite automaton.

The last stage is the composition of input and output specifications to obtain the overall specification P. Note that to be compatible with the approximation automaton G_ℓ, the overall specification has to have a special transition structure, namely, it must generate only sequences of events that consist of alternating symbols $y \in Y_d$ and $l \in L$, where the first symbol must belong to the set Y_d (see Fig.2(right)). Thus, the resulting specification automaton is obtained as an "ordered" product of P_L and P_Y:

$$P = P_Y \vee P_L = (P_Y \mid\mid P_L) \times \Omega = (S, Y_d \cup L, \delta, s_0), \tag{6}$$

where the automaton Ω is given by $(\{0,1\}, Y_d \cup L, \omega, \{0\})$,

$$\omega(x,\sigma) = \begin{cases} 1, & x = 0,\ \sigma \in Y_d, \\ 0, & x = 1,\ \sigma \in L, \\ \text{undefined}\ , & \text{otherwise.} \end{cases}$$

To characterize the resulting specification automata, we need the notion of *current-state observability* [cf. 3, 13]:

Definition 3.1 *A finite state machine $A = (Q, \Sigma, \phi)$ is said to be* current-state observable *if there exists a nonnegative integer K such that for every $i \geq K$, for any (unknown) initial state $q(0)$, and for any admissible sequence of events $\sigma(0)\ldots\sigma(i-1)$ the state $q(i)$ can be uniquely determined. The parameter K is referred to as the* index of observability*.*

In the following, we assume that the automata P_L and P_Y are current-state observable with indices of observability K_{P_L} and K_{P_Y}, respectively.

The notion of current-state observability can be extended to the overall specification automaton. Its index of observability K_P is given by

$$K_P = \begin{cases} 2K_{P_Y} - 1, & K_{P_Y} > K_{P_L}, \\ 2K_{P_L}, & K_{P_Y} \leq K_{P_L}. \end{cases} \tag{7}$$

Furthermore, to stay within the time-invariant framework we have to restrict ourselves to specifications that can be realized by strongly current-state observable automata:

Definition 3.2 *A finite state machine $A = (Q, \Sigma, \phi)$ is said to be* strongly current-state observable *if it is current-state observable with observability index K and if for each state $q \in Q$ there exists another state $q' \in Q$ such that the state q can be reached from q' by a sequence of K events, i.e. $\forall q \in Q\ \exists q' \in Q, \text{s.t. } q = \phi(q', s), s \in \Sigma^*, |s| = K$, where Σ^* is the Kleene closure of Σ and ϕ the extension of the automaton transition function to strings in Σ^*.*

Note that each state of such an automaton can be deduced from the string consisting of the past K events, independently from the initial state.

3.2.2 Supervisor design

Given an approximating automaton G_ℓ and a deterministic specification automaton (6) with observability index K_P, supervisory control theory checks, whether there exists a nonblocking supervisor and, if the answer is affirmative, provides a least restrictive supervisor SUP via "trimming" of the product of G_ℓ and P. Hence the state set of the supervisor, X_{SUP}, is a subset of $\tilde{X}_d \times S$.

The functioning of the resulting supervisor is very simple. At time k it "receives" a measurement symbol which triggers a state transition. In its new state $x_{sup}^{(j)}$, it enables a subset $\Gamma(x_{sup}^{(j)}) \subseteq L$ and waits for the next feedback from the plant. As shown in [11], the supervisor will enforce the specifications not only for the approximation, but also for the underlying hybrid plant model.

In the following, we will be interested in the special case of *quasi-static* specifications. To explain this notion, let $p_{app} : X_{SUP} \to \tilde{X}$ denote the projection of $X_{SUP} \subseteq \tilde{X}_d \times S$ onto its first component. If p_{app} is injective, i.e. if

$$p_{app}(x_1) = p_{app}(x_2) \Rightarrow x_1 = x_2, \tag{8}$$

and, moreover, the specification is strongly current-state observable, then the specification automaton is called *quasi-static with respect to the approximation automaton G_l.*

Proposition 3.3 *P is quasi-static with respect to G_ℓ if*

$$2\ell - 1 \geq K_P. \tag{9}$$

Proof. (Sketch) Let x_1 and x_2 be two states of the supervisor SUP with $p_{app}(x_1)$, $p_{app}(x_2) \in \tilde{X}_d$. There are two cases:

(1) $p_{app}(x_1), p_{app}(x_2) \in \tilde{X}_{d_j}$, $1 \leq j < \ell - 1$. Each element from $\tilde{X}_{d_j}$ stores a record of the complete past and present of y_d and l. Since the specification automaton is assumed to be deterministic, this record unambiguously determines the current state of the specification automaton. Thus, (8) holds.

(2) $p_{app}(x_1), p_{app}(x_2) \in \tilde{X}_{d_{\ell-1}}$. In this case an element from $\tilde{X}_{d_{\ell-1}}$ contains information only on "recent" past values of y_d and l. Precisely speaking, it contains information about the last ℓ output symbols and the last $\ell - 1$ control symbols. Thus, the complete record has length of $2\ell - 1$ symbols, which is sufficient to unambiguously determine the current state of P.

3.3 Plant model under low-level control

For the case of quasi-static specifications, each supervisor state $x_{sup}^{(i)}$ corresponds exactly to a state $\tilde{x}_d^{(i)} = p_{app}(x_{sup}^{(i)})$ of the approximating automaton, which, in turn, can be associated with a set $X(\tilde{x}_d^{(i)}) = X(p_{app}(x_{sup}^{(i)}))$.

For $k \geq \ell - 1$, attaching the discrete supervisor to the plant model is therefore equivalent to restricting the invariants for each location $l_j \in L$ according to

$$inv_{l_j} = \bigcup_{\substack{i,\ \text{s.t.}\ l_j \in \Gamma(x_{sup}^{(i)}) \\ p_{app}(x_{sup}^{(i)}) \in X_{d_{\ell-1}}}} X(p_{app}(x_{sup}^{(i)})) \bigcup f_i(X(p_{app}(x_{sup}^{(i)}))). \qquad (10)$$

Note that for the initial time segment, i.e. $k < \ell - 1$, (10) is more restrictive than the discrete supervisor computed in Sec.3.2.

Hence the action of supervisory control is to restrict the invariants from $inv_j = X$ to inv_j given by (10) and, accordingly, to restrict the guards from $g_ij = X$ to $g_ij = inv_i \cap inv_j$ where inv_i and inv_j are computed according to (10)

The union of all invariants $inv_{l_j},\ j = 1, \ldots, \alpha$, forms the refined state set that contains only safe points, i.e. points for which exists at least one sequence of control symbols such that the resulting behavior satisfies the specifications.

The resulting hybrid automaton represents the plant model under low-level control (for $k \geq \ell$). As control system synthesis has been based on an ℓ-complete approximation, it is guaranteed that the resulting hybrid automaton satisfies safety and liveness requirements. The remaining degrees of freedom in choosing $l(k)$ can be used in a high-level controller addressing performance issues.

4 The high-level task

The high-level task requires the solution of an optimal control problem of the form (3).

The aim of this section is that of showing in detail that a state feedback solution of (3) can be obtained by computing off-line appropriate partitions of the state space, that we call *switching regions*, extending to the case at hand previous results on the optimal control of switched systems [20], based on dynamic programming arguments. In particular, we present the following three main results.

- Firstly, we recall how one can extend the results of [20] to the case of HA with invariants in order to compute an optimal state feedback control law for the problem (3) when a finite number of switches N is allowed.
- Then, we show how the proposed approach can be easily extended to the case of an infinite number of allowed switches.

- Finally, we show how to deal with the case of hybrid systems whose dynamics are all unstable.

For sake of simplicity we will deal with completely connected automata. These results can be easily extended to the case of generic automata using the same arguments of [4], where continuous-time HA were taken into account.

4.1 The optimal control problem with a finite number of switches

Let us now consider an optimal control problem of the form:

$$\begin{cases} V_N^*(i_0, x_0) \triangleq \min\limits_{\mathcal{I},\mathcal{K}} \left\{ F(\mathcal{I},\mathcal{K}) \triangleq \sum\limits_{k=0}^{\infty} x(k)' Q_{i(k)} x(k) \right\} \\ \\ \text{s.t.} \qquad x(k+1) = A_{i(k)} x(k) \\ \qquad\quad i(k) = i_r \in L, \ \text{ for } \ k_r \le k < k_{r+1}, \quad r = 0,1,\ldots,N \\ \qquad\quad x(k) = inv_{i(k)}, \ \text{ for } \ k = 0,1,\ldots,+\infty \\ \qquad\quad 0 = k_0 \le k_1 \le \ldots \le k_N < k_{N+1} = +\infty \end{cases} \tag{11}$$

where Q_i are positive semi-definite matrices, (i_0, x_0) is the initial state of the system, and $N < +\infty$ is the maximum number of allowed switches, that is given a priori.

In this optimization problem there are two types of decision variables:

- $\mathcal{I} \triangleq \{i_1, \ldots, i_N\}$ is a finite sequence of modes;
- $\mathcal{K} \triangleq \{k_1, \ldots, k_N\}$ is a finite sequence of switching time indices.

Problem (11) is clearly well posed provided that the following hypothesis are verified.

Assumption 4.1 The invariant sets inv_i, $i \in L$, guarantee the liveness of the HA. ■

Note that Assumption 4.1 is generally not easy to verify. Nevertheless, in the case at hand, its satisfaction is guaranteed a priori by the low-level task, namely by the procedure used to construct the invariant sets.

Moreover, to ensure a finite optimal cost for any $x_0 \in \mathbb{R}^n$ and any $i_0 \in L$ we assume the following:

Assumption 4.2 There exists at least one mode $i \in L$ such that A_i is strictly

Hurwitz and $inv_i = \mathbb{R}^n$. ■

Note that this condition is sufficient but usually not necessary to get a finite optimal cost.

In [20] it was shown that under the assumption that $inv_i = \mathbb{R}^n$ for all $i \in L$, the optimal control law for the optimization problem (11) takes the form of a state-feedback, i.e., it is only necessary to look at the current system state in order to determine if a switch from linear dynamics $A_{i_{r-1}}$ to A_{i_r}, should occur.

For a given mode $i \in L$ when r switches are still available, it is possible to construct a table $\mathcal{C}_r^i$ that partitions the state space $\mathbb{R}^n$ into α regions $\mathcal{R}_j$'s, $j = 1, \cdots, \alpha = |L|$. Whenever $i_{N-k} = i$ we use table $\mathcal{C}_r^i$ to determine if a switch should occur: as soon as the continuous state x reaches a point in the region $\mathcal{R}_j$ for a certain $j \in L \setminus \{i\}$ we will switch to mode $i_{N-k+1} = j$; no switch will occur if the continuous system's state x belongs to $\mathcal{R}_i$.

In [20] it was constructively showed how the tables $\mathcal{C}_r^i$ can be computed off-line using a dynamic programming argument: first the tables $\mathcal{C}_1^i$ $(i \in L)$ for the last switch are determined, then, by induction the tables $\mathcal{C}_r^i$ can be computed once the tables $\mathcal{C}_{r-1}^i$ are known.

Remark 4.3 In order to provide a graphical representation of $\mathcal{C}_r^i$ we associate a different color to each dynamics A_j, $j \in L$. The region $\mathcal{R}_j$ of $\mathcal{C}_r^i$ is represented according to the defined color mapping. ■

Note that when $inv_i = \mathbb{R}^n$ for all $i \in L$, the regions $\mathcal{R}_j$'s are homogeneous, namely if $x \in \mathcal{R}_j$ then $\lambda x \in \mathcal{R}_j$ for all $\lambda \in \mathbb{R}$. This implies that they can be computed by simply discretizing the unitary semisphere. Clearly, this is no more valid when $inv_i \subsetneq \mathbb{R}^n$ for some $i \in L$, as in the case of interest here where a discretization of all state space of interest is necessary.

To show how the procedure of [20] can be extended to the case we are considering here, let $y \in \mathbb{R}^n$ be a generic vector, and let $\mathcal{D}$ be an appropriate set of points in the portion of the state space of interest, that define the considered state space discretization grid. Moreover, given a discrete state $i \in L$ and a continuous state $y \in \mathbb{R}^n$, we define the set

$$succ(y) = \{j \in L \mid y \in inv_j\}$$

which denotes the indices associated to the locations whose invariant set includes y.

The procedure to compute the switching regions is based on dynamic programming. Let us denote $T_r(i, y)$ the optimal remaining cost when the current continuous state is y, the current dynamics is A_i and r switches are still available. Thus, when $r = 0$, i.e., when no more switch may occur, $T_0(i, y) = y' Z_i y$ if A_i is Hurwitz and the system trajectory starting in y and evolving with dynamics A_i until the origin is reached, always keeps within inv_i. The matrix Z_i is the solution of the discrete Lyapunov equation $A_i' Z_i + Z_i A_i = -Q_i$. In all the other cases, $T_0(i, y) = +\infty$.

The optimal remaining cost $T_r(i, y)$ for $r = 1, \ldots, N$, is computed recursively starting from $r = 1$ towards increasing values of r. More precisely, we first choose a finite time horizon $k_{\max}$ that is large enough to approximate the infinite time horizon. Then, for any $r = 1, \ldots, N$, any $y \in \mathcal{D}$ and any $i \in L$, we compute the value of the cost to infinity when the initial state is (i, y), r switches are still available, the generic dynamics A_i is used for the first k times sampling, then the system switches to dynamics A_j. This cost, that we denote $T(i, y, j, k, r)$, is the sum of the cost due to the evolution with dynamics A_i for k times sampling, plus the optimal remaining cost from the new state $(j, A_i^k y)$ reached after the switch when $r - 1$ switches are still available, i.e.,

$$T(i, y, j, k, r) = y' \left(\sum_{h=0}^{k} (A_i^h)' Q_i (A_i^h) \right) y + T_{r-1}(j, A_i^k y). \tag{12}$$

The previous equation expresses the dynamic programming argument used to efficiently compute the optimal switching law. When r switches are still available, we consider a nominal trajectory starting from a discretization grid point y and evaluate its cost assuming we remain in the current dynamic i for a time k; the cost remaining after the after the switch the switch is evaluated on the basis of the table previously constructed for the $r-1$ switch. Note that for a fixed value of i, y, j and r we only need to find the optimal value of k with a one-dimensional search.

Note that only those evolutions that do not violate the invariant constraints should be taken into account. This means that if $z = A_i^k y$ is the generic continuous state reached from y evolving with dynamics A_i for k steps, an evolution with dynamics A_j, $j \in L$, should be considered if and only if $j \in succ(z)$.

Finally, we define the switching tables as mappings $\mathcal{C}_r^i : \mathcal{D} \to L$, and the generic region $\mathcal{R}_j$ as

$$\mathcal{R}_j = \{ y \in \mathcal{D} \mid \mathcal{C}_k^i(y) = j \}.$$

The procedure to compute the switching regions is briefly summarized in the following algorithm.

Algorithm 4.4 (Tables construction)

Input:

$$A_i \in \mathbb{R}^{n\times n},\ \ Q_i \in \mathbb{R}^{n\times n},\ \ inv_i \subseteq \mathbb{R}^n\ \ (i \in L),\ \ N,\ \ k_{\max},\ \ \mathcal{D}.$$

Output:

$$\mathcal{C}_r^i,\ \ r = 0, 1, \ldots, N,\ \ i \in L.$$

Notation:

$$\bar{Q}_i(k) = \sum_{h=0}^{k} (A_i^h)' Q_i (A_i^h),\ \ Z_i = \lim_{k\to\infty} \bar{Q}_i(k).$$

```
(1) Initialization: $r = 0$ remaining switches
      for $i = 1 : \alpha$
          for all $y \in \mathcal{D}$
              Cost assignment:

$$T_0(i,y) = \begin{cases} y' Z_i y \text{ if } A_i \text{ is Hurwitz and the system trajectory starting} \\ \qquad \text{in } y \text{ and evolving with dynamics } A_i \text{ until the origin} \\ \qquad \text{is reached, always keeps within } inv_i \\ +\infty \quad \text{otherwise} \end{cases}$$

          end ($y$)
      end ($i$)
(2) for $r = 1 : N$
          for $i = 1 : \alpha$
              for all $y \in \mathcal{D}$
                  Computation of the remaining cost:
                  let $k = 0$, $\Delta = \emptyset$
                  while $k \le k_{\max}$
                      $z = A_i^k y$
                      if $z \notin inv_i$
                          for all $j \in succ(z)$
                              $T(i,y,j,k,r) = y'\bar{Q}_i(k)y + T_{r-1}(j,k)$,
                              $\Delta = \Delta \cup \{(j,k)\}$
                          end ($j$)
                          $k = k_{\max} + 1$
                      else
                          for all $j \in succ(z) \setminus \{i\}$
                              $T(i,y,j,k,r) = y'\bar{Q}_i(k)y + T_{r-1}(j,k)$,
                              $\Delta = \Delta \cup \{(j,k)\}$
                          end ($j$)
                          $k = k + 1$
                      end if
```

```
                end while
                if $i \in succ(y)$ and $A_i$ is Hurwitz
                    $T(i, y, i, k_{\max}, r) = y' Z_i y$,
                    $\Delta = \Delta \cup \{(i, k_{\max})\}$
                end if
                Cost assignment: $T_r(i, y) = \min_{(j,k)\in\Delta} T(i, y, j, k, r)$
                Color assignment: $(j^*, k^*) = \arg\min_{(j,k)\in\Delta} T(i, y, j, k, r)$,
$\mathcal{C}_r^i(y) = \begin{cases} j^* & \text{if } k^* = 0 \\ i & \text{otherwise} \end{cases}$
            end $(y)$
        end $(i)$
    end $(r)$
```

Algorithm 4.4 first computes the optimal remaining cost $T_0(i, y)$ when no more switches are available. This is obviously done for any $i \in L$ and any $y \in \mathcal{D}$. Then, the switching tables are computed backwards, starting from that corresponding to one available switch, until that corresponding to N available switches ($r = 1 : N$). More precisely, for any r, any mode A_i and any sampling point $y \in \mathcal{D}$, we compute the optimal remaining cost starting from (i, y) when r switches are available. In order to do this we compare all the costs that can be obtained starting from (i, y), evolving with A_i for k sampling instants, then switching to any mode A_j, and up to then evolving with the optimal evolution from (j, z), $z = A_j^k y$, when $r-1$ switches are available. Note that k may only take finite values, namely $k = 0, 1, \dots, k_{\max}$. This clearly does not affect the validity of the solution provided that $k_{\max}$ is taken large enough. If the minimum cost is obtained for $k = k^* = 0$ and $j = j^*$, this means that when the current state is (i, y) and r switches are still available, the cost is minimized if we immediately switch to mode A_j. Therefore, we assign the color corresponding to A_j to the point y in the table $\mathcal{C}_r^i$. On the contrary, if $k^* > 0$ it means that if the state is (i, y) and r switches are available, it is convenient to continue evolving with dynamics A_i. Therefore we assign the color corresponding to A_i to the point y in the table $\mathcal{C}_r^i$.

The computational cost of the proposed approach is of the order $\mathcal{O}(q^n N s^2)$ where n is the dimension of the state space and q is the number of samples in each direction (i.e., q^n is the cardinality of $\mathcal{D}$). Therefore, the complexity is a quadratic function of the number of possible dynamics and linear in the number of switches.

Remark 4.5 Two important cautions should be taken in order to ensure the uniqueness of the tables and (as it will be discussed in the following) the Zeno-freeness when the procedure is extended to the case of $N = \infty$.

The argument (j^*, k^*) that minimizes the cost $T(i, y, j, k, r)$ may be not unique and this may cause ambiguity in the construction of the tables. To this aim we introduce the following lexicographic ordering.

Let $\Gamma = \{(j, k) \in \Delta \mid (j, k) = \arg\min T(i, y, j, k)\}$ be the set of solutions of problem

$$T_r(i, y) = \min_{(j,k)\in\Delta} T(i, y, j, k, r), \tag{13}$$

and assume that Γ has cardinality greater than one. Let (j', k') and (j'', k'') be any two couples in Γ with $j' \neq j''$. We say that $(j', k') \prec (j'', k'')$ iff $j' < j''$. Finally, if $j' = j''$ we say that $(j', k') \prec (j'', k'')$ iff $k'' < k'$.

Choosing the minimal element in Γ with respect to $\prec$, the optimal solution of Problem (13) is unique.

The second caution that should be taken is the following. Consider the case in which at a given value of the switching index k, the arguments that minimize the remaining cost $T(i, y, j, k)$ starting from point y in dynamics A_i are (j^*, k^*) with $k^* = 0$. It may be the case that the system, once entered in dynamics A_{j^*} requires an immediate switch to another dynamics, say p causing the presence of 2 switches in zero time. This behavior is undesirable, because it leads to a potential risk of a Zeno behavior when the number of available switches goes to infinite.

To avoid this it is sufficient to take (p, k^*) instead of (j^*, k^*), or more precisely, to consider $(j^*, k^*) = \arg\min\ T(j^*, y, j, k)$ at the previous switching index $r - 1$. When this extra precaution is taken, we can ensure that a spacing condition $k_{r+1} - k_r > 0$ is always verified during an optimal evolution.

■

4.2 The optimal control problem with an infinite number of switches

In [6] it was shown that under the assumption that $inv_i = \mathbb{R}^n$ for all $i \in L$, the above procedure can be extended to the case of $N = \infty$, provided that (i) for at least one $i \in L$, A_i is Hurwitz, and (ii) for all $i \in L$, $Q_i > 0$.

Analogous results can be proved here under Assumptions 4.1 and 4.2, and under the additional following hypothesis.

Assumption 4.6 For all $i \in L$, $Q_i > 0$. ■

Proposition 4.7 For any continuous initial state x_0, $x_0 \neq 0$, and $\forall\ \varepsilon > 0$,

$\exists\ \bar{N}$ such that for all $N > \bar{N}$,

$$\frac{V_N^*(i,x_0) - V_{\bar{N}}{}^*(j,x_0)}{V_N^*(i,x_0)} < \varepsilon,$$

for all $i, j \in L$.

Proof. See Appendix. $\square$

According to the above result, one may use a given fixed relative tolerance ε to approximate two cost values, i.e.,

$$\frac{V_N^*(i,x) - V_{N'}{}^*(j,x)}{V_N^*(i,x)} < \varepsilon \implies V_N^*(i,x) \cong V_{N'}{}^*(j,x).$$

This result enables us to prove the following important theorem.

Theorem 4.8 Given a fixed relative tolerance ε, if $\bar{N}$ is chosen as in Proposition 4.7 then for all $i, j \in L$ it holds that $\mathcal{C}^i_{\bar{N}+1} = \mathcal{C}^j_{\bar{N}+1}$.

Proof. It trivially follows from the fact that, by Proposition 4.7, $V^*_{\bar{N}+1}(i,x_0) = V^*_{\bar{N}+1}(j,x_0)$ for all $i, j \in L$, and from the uniqueness of the optimal tables as discussed in Remark 4.5. $\square$

This result also allows one to conclude that

$$\forall\ i \in L, \quad \mathcal{C}_\infty = \lim_{N\to\infty} \mathcal{C}^i_N,$$

i.e., *all tables converge to the same one.*

To construct the table $\mathcal{C}_\infty$ the value of $\bar{N}$ is needed. We do not provide so far any analytical way to compute $\bar{N}$, therefore our approach consists in constructing tables until a convergence criterion is met.

Table $\mathcal{C}_\infty$ can be used to compute the optimal feedback control law that solves an optimal control problem of the form (11) with $N = \infty$. More precisely, when an infinite number of switches is available, we only need to keep track of the table $\mathcal{C}_\infty$. If the current continuous state is x and the current mode is A_i, on the basis of the knowledge of the color of $\mathcal{C}_\infty$ in x, we decide if it is better to still evolve with the current dynamics A_i or switch to a different dynamics, that is univocally determined by the color of the table in x.

Let us finally observe that table $\mathcal{C}_\infty$ is *Zeno-free*, i.e., it guarantees that no Zeno instability may occur when it is used to compute the optimal feedback control law. This property is guaranteed by the procedure used for their construction as discussed in Remark 4.5.

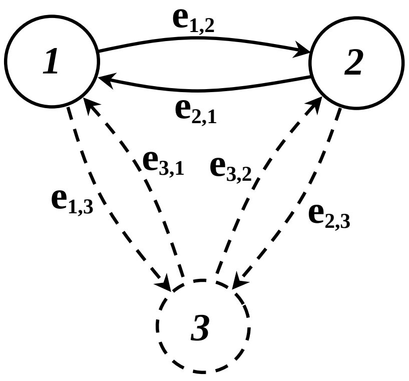

Fig. 4. *Graph of the automaton HA (continuous) and $\overline{HA}$ (continuous and dashed) described in the example.*

4.3 The optimal control of switched systems with unstable dynamics

In this section we show that the above results still apply when Assumption 4.2 is relaxed. To this aim, let us first introduce the following definitions.

Definition 4.9 (Forbidden region) *A forbidden region for the HA is a set $X_f \subset X : X_f = X \setminus \bigcup_{i=1}^{s} inv_i$, where α is the number of locations.* ■

Thus X_f is a region forbidden to *all* dynamics of the HA.

Definition 4.10 (Augmented HA and OP) *An* augmented automaton $\overline{HA} = (\overline{L}, \overline{act}, \overline{inv}, \overline{E})$ *of $HA = (L, act, inv, E)$ and the corresponding optimal control problem $\overline{OP}$ of OP, are related as follows:*

(i) $\overline{HA}$ includes a new Hurwitz dynamics $A_{\alpha+1}$ and $\overline{OP}$ includes a corresponding weight matrix $Q_{\alpha+1} = q\tilde{Q}_{\alpha+1}$ (with $rank(\tilde{Q}_{\alpha+1}) > 0$, and $q > 0$).
(ii) A new invariant $inv_{\alpha+1} = X$ is associated to the new dynamics.
(iii) The edges $e_{i,\alpha+1} \in \overline{E}$ and $e_{\alpha+1,i} \in \overline{E}$ are defined $\forall\, i \in \overline{L}$.

■

Thus the augmented automaton $\overline{HA}$ is the same as HA except for an extra location $(\alpha+1)$ completely connected to all the locations in the HA. Its invariant set coincides with $inv_{\alpha+1} = X$ and its dynamics is $A_{\alpha+1}$. The corresponding $\overline{OP}$ weights location $(\alpha + 1)$ with matrix $Q_{\alpha+1} > 0$.

The following important result holds.

Proposition 4.11 *Assume that there exists an exponentially stabilizing switching law for problem $OP(HA)$. Then there also exists a sufficiently large value* *of $q > 0$ in the $\overline{OP}(\overline{HA})$, such that the table $\overline{\mathcal{C}}_\infty$, solution of $\overline{OP}(\overline{HA})$, $i = 1, \ldots, \alpha + 1$, contain the color of $A_{\alpha+1}$ at most in X_f.* ■

Proposition 4.11 allows one to consider the solution of $\overline{OP}(\overline{HA})$ equivalent to the solution of $OP(HA)$. This follows from the fact that the dynamics $A_{\alpha+1}$ does not influence at all any solution of the augmented problem. Therefore it can be removed from the augmented automaton.

This result is formally proved in [6], in absence of state space constraints. As before this result can be trivially extended if the liveness of the automaton is guaranteed (see Assumption 4.1). In fact, by definition, it holds that, for any initial couple $(i_0, x_0) \in X \setminus X_f$ of the HA, the hybrid trajectory $(i(k), x(k))$, solution of $OP(HA)$, always keeps within $X \setminus X_f$.

Let us finally observe that the convergence to a unique table $\overline{\mathcal{C}}_\infty$ is due to the fact that we are dealing with strongly connected HA. If such were not the case, then α different tables $\overline{\mathcal{C}}^i_\infty$, $i = 1, \ldots, \alpha$, would be obtained as a solution of $\overline{OP}$, and all of them should be used to compute the optimal feedback control law.

4.4 Robustness of the procedure

The high-level procedure we suggest in this paper provides a switching table, i.e., a partition of the state space, that the controller consults on-line in order to establish which is the current mode that ensures the minimality of the cost function. If no disturbance is acting on the system and no numerical error affects the switching table construction, the optimality of the solution, as well as the safeness and liveness of the closed-loop system, are guaranteed.

In practice, two different problems may occur.

- The first one concerns disturbances acting on the continuous system that may change its nominal trajectory. If the disturbance does not bring the system inside the forbidden region X_f, this does not affect the validity of the result: the controller continues taking its decisions on the basis of the current continuous state, the minimality of the cost is guaranteed, and the safety and liveness constraints are still satisfied.

  On the contrary, the procedure obviously fails if the disturbance is such that the trajectory enters the forbidden region X_f: in this case the safety and liveness constraints cannot be satisfied any more.
- The second problem is related to the inevitable numerical errors that affect the construction of the switching tables due to the state space discretization required by our approach. For each switch, the table is computed for all points that belong to the discretization grid. From these points the nominal

trajectories are studied piecewise using eq. (12): after the switching the remaining evolution may start from a point that does not belong to the grid and the actual remaining trajectory is an approximation of the nominal one.

The actual trajectories during the system evolution will be close to the nominal ones used to compute the tables if the discretization step is sufficiently small. A numerical error in the computation of the tables is not critical if the forbidden region is empty. In such a case, the provided solution may be sub-optimal but it is still viable.

However, assume that the forbidden region is not-empty. If the forbidden regions constraints the optimal solution, it is likely that an optimal nominal trajectory needs to pass as close as possible to the forbidden region without entering it. However, an evolution that differs from a nominal one may graze or even hit the forbidden region. In this case, the safety and liveness constraints cannot be satisfied any more.

A solution we suggest to overcome both problems is the following.

In the low level task we define a "tolerance region" X_{tr} around the forbidden state space set X_d, then we redefine the forbidden state space set as $X'_d = X_d \cup X_{tr}$. This clearly implies a reduction of the invariant sets of each location $(inv'_1, \ldots, inv'_\alpha)$, and consequently, a wider forbidden region $X'_f \supset X_f$ in the high level task.

The tolerance region should be large enough to make sure that a trajectory that differs from a nominal one — either because a disturbance is acting on the system, or because of the numerical errors in the construction of the tables — may pass within X'_f but never reaches X_f.

In this way we improve the robustness of the procedure. We have to pay a cost for this: the optimality of the solution with respect to the chosen performance index is no more guaranteed, and the computed solution will be suboptimal.

5 Numerical example

Let us consider a HA with two locations, whose graph is depicted in Figure 4 (the part sketched with continuous lines). Moreover, let

$$A_1 = \begin{bmatrix} 0.981 \; 0.585 \\ -0.065 \; 0.981 \end{bmatrix}, \; A_2 = \begin{bmatrix} 0.981 \; 0.065 \\ -0.585 \; 0.981 \end{bmatrix}$$

be the corresponding continuous dynamics, whose eigenvalues have unitary norm. In particular, in both cases it holds $\lambda_{1,2} = 0.9808 \pm j0.1951$. Two generic

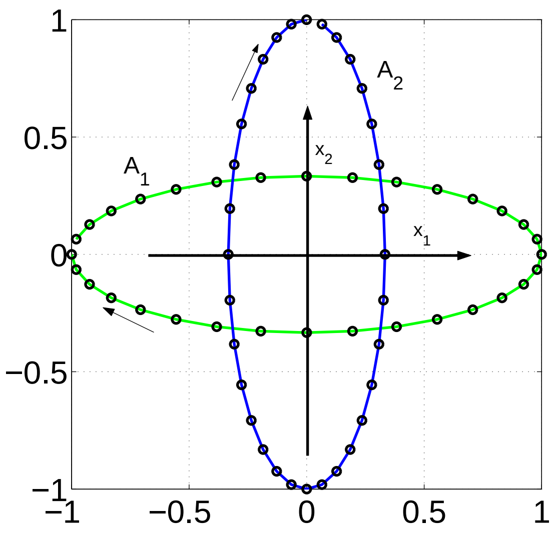

Fig. 5. *Discrete time trajectories of dynamics A_1 and A_2, with eigenvalues along the unitary circle.*

trajectories relative to dynamics A_1 and A_2, respectively, are reported in Figure 5.

Assume that

$$X = \{x \in \mathbb{R}^2 \mid x_1^2 + 9x_2^2 \leq 40 \ \wedge \ 9x_1^2 + x_2^2 \leq 40\},$$

where $x_1^2 + 9x_2^2 \leq 40$ and $x_1^2 + x_2^2 \leq 40$ are the equations of the trajectories trough the point of coordinates $x_1 = x_2 = 2$ and evolving with dynamics A_1 and A_2, respectively.

Finally, assume that the safety constraint is given by the forbidden state space set

$$X_d = \{x \in \mathbb{R}^2 | H'x \leq h\}$$

where

$$\begin{aligned} H &= \begin{bmatrix} 0 & 0 & 1 & -1 \\ 1 & -1 & -1 & -1 \end{bmatrix} \\ h &= \begin{bmatrix} 0.8 & -0.2 & 0 & 0 \end{bmatrix}, \end{aligned} \qquad (14)$$

i.e., X_d is the trapezoid depicted in Fig.6.

The set $X \setminus X_d$ can be blocking, i.e., there exists some initial point in $X \setminus X_d$ such that, regardless of the switching strategy, any trajectory starting from these points always hit the set X_d.

In order to guarantee liveness, the previous setup is passed to the procedure described in Section 3. In terms of safety specification it means that we mark all output symbols $y_d \in \hat{Y}_d$, where $\hat{Y}_d = \{y_d \in Y_d | q_y^{-1}(y_d^{(i)}) \cap X_d \neq \emptyset\}$, as forbidden and require that the allowed sequences of output symbols do not contain such forbidden symbols. Then, the obtained supervisor is transformed into the appropriate invariant sets inv_1 and inv_2 to be associated to the dynamics A_1 and A_2, respectively. A new forbidden region $X_f = X \setminus (inv_1 \cup inv_2) \supset X_d$ is defined according to Definition 4.9.

The invariant sets of locations 1 and 2 are reported in Fig.6.(a) and (b), respectively, while the set X_f is sketched in figure (c).

Within the given constraints we want to solve an optimal control problem[2] OP of the form (3), where $Q_1 = Q_2 = I$. For this purpose we consider the augmented problem $\overline{OP}(\overline{HA})$, with the following data:

$$A_3 = \begin{bmatrix} 0.9838 & 0 \\ 0 & 0.9838 \end{bmatrix}, \quad Q_3 = qQ_1, \quad inv_3 \equiv X$$

where $q = 10^3$, and A_3 is Hurwitz. The graph of the augmented automaton is depicted in Fig.4 (continuous and dashed part).

Remark 5.1 *Given the symmetry of the two dynamics, it can be easily shown that the solution of $OP(HA)$ when $inv_i \equiv X$, $i = 1, 2$, is to use dynamics A_2 when $x_1x_2 > 0$ and dynamics A_1 when $x_1x_2 < 0$. This result is very intuitive if we observe the trajectories of the given dynamics (Figure 5) and if we use the identity matrices as weight matrices in problem (3).*

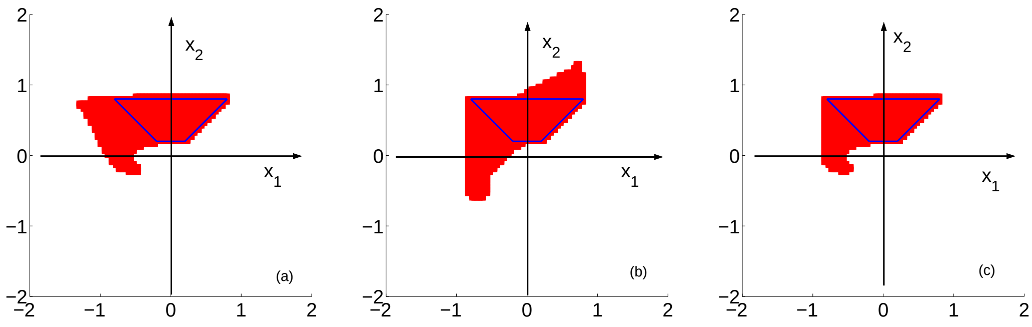

Fig. 6. *Invariants (in white) of locations 1 (a) and 2 (b) and (c) the forbidden region $X_f = X \setminus (inv_1 \bigcup inv_2)$ defined in Def. 4.9. The interior of the blue trapezoid is the forbidden region X_d.*

Note that the augmented problem $\overline{OP}(\overline{HA})$ satisfies the conditions given in Definition 4.10. The switching table procedure, applied to $\overline{OP}(\overline{HA})$ for a recursively increasing number of switches, converges after $N = 15$ switches. Moreover the tables $\mathcal{C}^i_\infty$, $i = 1, 2, 3$ are coincident with a unique table $\mathcal{C}_\infty$, because $\overline{HA}$ is strongly connected.

[2] Note that neither A_1 nor A_2 are Hurwitz, hence an infinite number of switches is necessary.

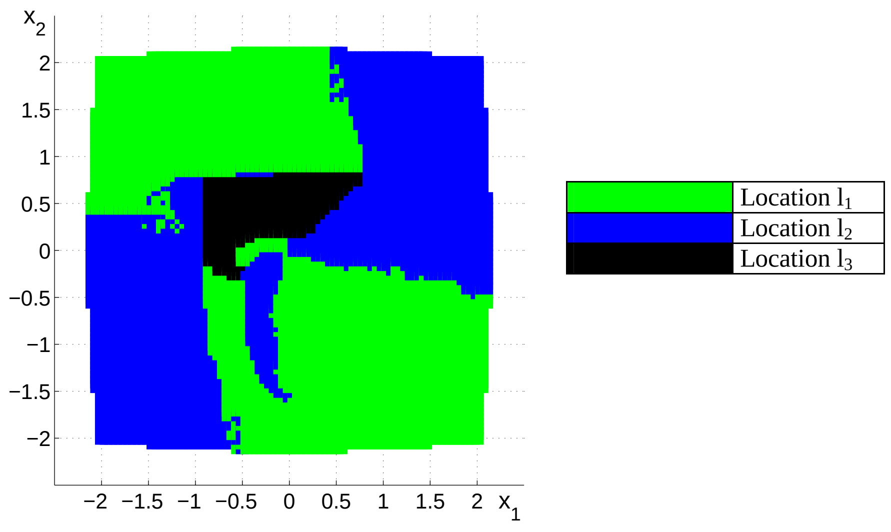

Fig. 7. *Switching table of the problem* $\overline{OP}(\overline{HA})$ *defined in the example.*

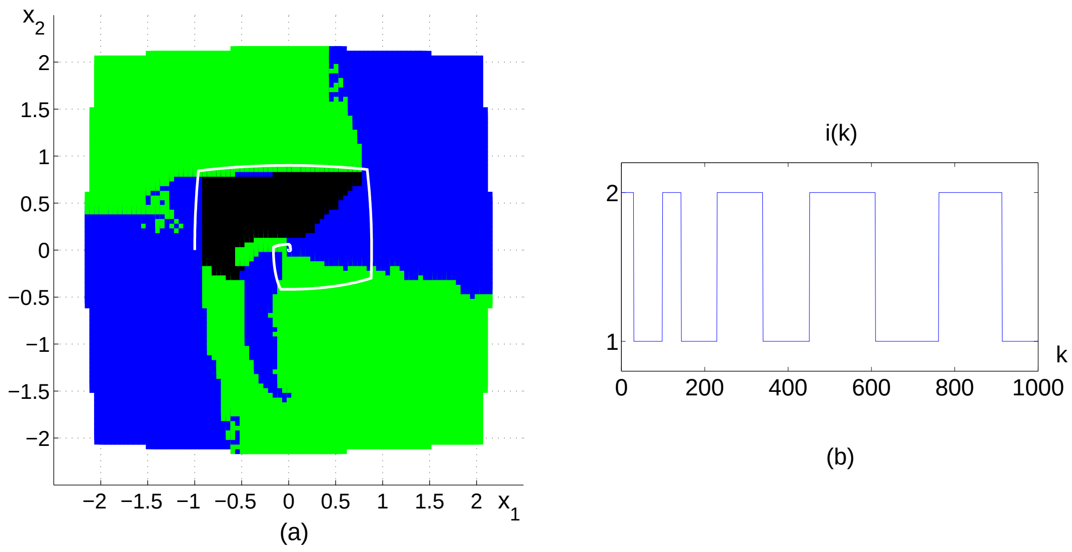

Fig. 8. *Trajectories* $x(k)$ *(a) and* $i(k)$ *(b) of the optimal solution of* $OP(HA)$ *obtained by using the table in Figure 7 for an initial point* $(i_0 = 1, x_0 = [-1\ 0]')$.

The table $\mathcal{C}_\infty$ is depicted in Figure 7 and some relevant considerations can be immediately done.

(i) The color of the augmented dynamics exactly covers the region X_f.
(ii) In $inv_1 \setminus X_f$ $(inv_2 \setminus X_f)$ the color is that relative to dynamics A_2 (A_1).
(iii) Around the origin the solution of $OP(HA)$ coincides with the solution described in Remark 5.1, relative to the case of $inv_i = X$ for $i = 1, 2$.

From the above table we deduce that there exists a finite optimal solution for any initial hybrid state $(i_0, x_0) \in X \setminus X_f$ of the HA; moreover, if $(i_0, x_0) \in X_f$ the optimal solution of $\overline{HA}$ uses dynamics A_3 for the minimum time required to leave X_f; from then on the optimal solution of HA is used. This can be viewed by the simulations depicted in Figure 8(a). The optimal cost from the point $(i_0 = 1, x_0 = [-1\ 0]')$ is $J = 196.6$. For completeness also the index trajectory $i(k)$ is reported in Figure 8(b).

The total computational time (Matlab 7, on an Intel Pentium 4 with 2 GHz and 256 Mb RAM) for constructing the table in Figure 7 is about 40 hours. This time is big, because a very dense space discretization was considered (approximately 2×10^3 points). It is important, however, to point out that this computational effort is spent off-line. The on-line part of the procedure consists in measuring the hybrid state $(i(k), x(k))$ and comparing its value with the switching table to decide the optimal strategy.

6 Conclusion

We addressed the problem of designing a feedback control law for a discrete time hybrid automaton $\mathcal{H}A$. We showed that this law can be designed so that the system's behavior satisfies two levels of specifications. The *low level* specification prescribes liveness and safety conditions for the $\mathcal{H}A$. We showed that the action of the low-level controller can be implemented restricting the invariants of $\mathcal{H}A$. The *high level* specification deals with the optimization of the trajectories: within the degree of freedom left by the low level task, for a given initial state the high-level task finds the evolution that minimizes a given performance index. The robustness of the procedure has also been discussed. One perspective of interest for future developments is to provide structural conditions of the $\mathcal{H}A$ that guarantee the existence of admissible optimal control laws.

References

[1] R. Alur and D.L. Dill. A theory of timed automata. In *Theoretical Computer Science*, 1994.

[2] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proceedings of the IEEE*, 88(7), 2000.

[3] P.E. Caines, R. Greiner, and S.Wang. Dynamical logic observers for finite automata. In *Proceedings of the 27th Conference on Decision and Control*, pages 226–233, Austin, Texas, 1988.

[4] D. Corona, A. Giua, and C. Seatzu. Optimal control of hybrid automata:

an application to the design of a semiactive suspension. *Control Engineering Practice*, 12:1305–1318, 2004.

[5] D. Corona, A. Giua, and C. Seatzu. Optimal feedback switching laws for autonomous hybrid automata. In *Proc. IEEE Int. Sym. on Intelligent Control*, Taipei, Taiwan, 2004.

[6] D. Corona, A. Giua, and C. Seatzu. Stabilization of switched system via optimal control. In *Proceedings of 16th IFAC World Congress*, Praga, Czech Republic, 2005.

[7] D. Corona, C. Seatzu, A. Giua, D. Gromov, E. Mayer, and J. Raisch. Optimal hybrid control for switched affine systems under safety and liveness constraints. In *Proceedings of CACSD'05*, Taipei, Taiwan, 2004.

[8] J.E.R. Cury, B.H. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Transactions on Automatic Control*, 43(4):564 – 568, April 1998.

[9] D. Gromov, E. Mayer, J. Raisch, D. Corona, C. Seatzu, and A. Guia. Optimal control of discrete time hybrid automata under safety and liveness constraints. In *ISIC*, Limassol, Cyprus, June 2005.

[10] S. Hedlund and A. Rantzer. Convex dynamic programming for hybrid systems. *IEEE Trans. Automatic Control*, 47(9):1536–1540, September 2002.

[11] T. Moor and J. Raisch. Supervisory control of hybrid systems within a behavioural framework. *Systems and control letters*, 38:157–166, 1999. Special issue on *Hybrid Control Systems*.

[12] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In *Hybrid Systems, LNCS 736, Springer Verlag*, 1993.

[13] Cüneyt M. Özveren and Allan S. Willsky. Observability of discrete event dynamic systems. *IEEE Trans. Automatic Control*, 35(7):797 – 806, 1990.

[14] B. Piccoli. Necessary conditions for hybrid optimization. In *Proc. 38th IEEE Conf. on Decision and Control*, Phoenix, Arizona USA, December 1999.

[15] J. Raisch. Discrete abstractions of continuous systems – an input/output point of view. *Mathematical and Computer Modelling of Dynamical Systems*, 6(1):6 – 29, 2000. Special issue on Discrete Event Models of continuous systems.

[16] J. Raisch and S. O'Young. Discrete approximation and supervisory control of continuous systems. *IEEE Transactions on Automatic Control*, 43(4):569–573, 1998.

[17] P.J. Ramadge and W.M. Wonham. The control of discrete event systems. *Proc. IEEE, Special Issue on Discrete Event Dynamic Systems*, 77(1):81–98, 1989.

[18] P. Riedinger, F. Kratz, C. Iung, and C. Zanne. Linear quadratic optimization for hybrid systems. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 3059–3064, Phoenix, USA, December 1999.

[19] J. Schröder. *Modelling, State observation and Diagnosis of Quantised*

Systems. Springer-Verlag, 2002.

[20] C. Seatzu, D. Corona, A. Giua, and A. Bemporad. Optimal control of continuous-time switched affine systems. *IEEE Transactions on Automatic Control,* to appear, 2005.

[21] M.S. Shaikh and P.E. Caines. On the optimal control of hybrid systems: analysis and algorithms for traiectory and scheduled optimization. pages 2144–2149, Hawaii, USA, December 2003.

[22] H.J. Sussmann. A maximum principle for hybrid optimal control problems. In *Proc. 38th IEEE Conf. on Decision and Control*, pages 425–430, Phoenix, USA, December 1999.

[23] J.C. Willems. Models for dynamics. *Dynamics reported*, 2, 1989.

[24] X. Xu and P.J. Antsaklis. An approach to switched systems optimal control based on parameterization of the switching instants. In *Proc. IFAC World Congress*, Barcelona, Spain, 2002.

[25] X. Xu and P.J. Antsaklis. Results and perspectives on computational methods for optimal control of switched systems. In *Hybrid Systems, Computation and Control*, Prague, Czech Republic, 2003.

Appendix: Proof of Proposition 4.7

Let us first introduce some preliminary results.

Property 6.1 Let $N, N' \in \mathbb{N}$. If $N < N'$ and the switched system evolves along an optimal trajectory, then for any continuous initial state x_0, and for all $i, j \in L$,

$$+\infty > V_N^*(i, x_0) \geq V_{N'}^*(j, x_0).$$

Proof. We first observe that by Assumptions 4.1 and 4.2 $V_N^*(i, x_0)$ is finite for any $N \geq 1$. In fact, regardless of the value of the initial dynamics i and the continuous state x_0, we can always switch to the stable dynamics whose cost to infinity is finite. Now, we prove the second inequality by contradiction. Assume that $\exists\ j \in L$ such that $V_{N'}^*(j, x_0) > V_N^*(i, x_0)$. Then it is obvious that the same evolution that generates $V_N^*(i, x_0)$ is also admissible for (11) starting from (j, x_0) when a larger value N' of switches is allowed (it is sufficient to switch immediately from mode j to mode i). This leads to a contradiction. □

Proposition 6.2 Given a continuous initial state x_0, for any $\varepsilon' > 0$, there exists $\bar{N} = \bar{N}(x_0) \in \mathbb{N}$ such that for all $N > \bar{N}$, $V_N^*(i, x_0) - V_{\bar{N}}{}^*(j, x_0) < \varepsilon'$ for all $i, j \in L$.

Proof. By definition $V_N^*(i, x_0) \geq 0$ for all $i \in L$, hence V_N^* is a lower bounded non-increasing sequence (by Property 6.1). By the Axiom of Completeness it converges in $\mathbb{R}$, hence it is a Cauchy sequence. □

Now, the statement of Proposition 4.7 can be proved.

We first observe that by Assumption 4.6 $V_N^*(i,x_0)$ is lower bounded by a strictly positive number. Moreover, the optimal costs are quadratic functions of x_0, i.e., if $x_0 = \lambda y_0$, then $V_N^*(i,\lambda y_0) = \lambda^2 V_N^*(i,y_0)$. Finally, by Proposition 6.2 $\forall\, y_0$ and $\forall\, \varepsilon' > 0$, $\exists\, \bar{N}(y_0)$ such that $\forall\, N > \bar{N}(y_0)$, $V_N^*(i,y_0) - V_{\bar{N}}{}^*(j,y_0) < \varepsilon'$. Hence if we define

$$\bar{N} = \max_{y_0\,:\,||y_0||=1} \bar{N}(y_0) \quad \Rightarrow$$

$$\frac{V_N^*(i,x_0) - V_{\bar{N}}{}^*(j,x_0)}{V_N^*(i,x_0)} = \frac{\lambda^2[V_N^*(i,y_0) - V_{\bar{N}}{}^*(j,y_0)]}{\lambda^2 V_N^*(i,y_0)}$$

$$\leq \frac{\varepsilon'}{\min\limits_{y_0\,:||y_0||=1} V_N^*(i,y_0)} = \varepsilon.$$

$\square$