CONTROLLED INVARIANCE OF DISCRETE TIME SYSTEMS

by

Rene Vidal, Shawn Schaffert, John Lygeros and Shankar Sastry

Memorandum No. UCB/ERL M99/65

16 December 1999

COVER

CONTROLLED INVARIANCE OF DISCRETE TIME SYSTEMS

by

Rene Vidal, Shawn Schaffert, John Lygeros and Shankar Sastry

Memorandum No. UCB/ERL M99/65

16 December 1999

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Controlled Invariance of Discrete Time Systems*

René Vidal Shawn Schaffert John Lygeros Shankar Sastry

Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720-1774
{rvidal,sms,lygeros,sastry}@eecs.berkeley.edu

December 16, 1999

Abstract

An algorithm for computing the maximal controlled invariant set and the least restrictive controller for discrete time systems is proposed. We show how for discrete time linear systems the algorithm can be encoded using quantifier elimination and linear programming. It is proved that for a general discrete-time linear system with all sets specified by linear inequalities, the problem is semi-decidable. If in addition the system is in canonical controllable form, the input is scalar and unbounded, the disturbance is scalar and bounded and the initial set is a rectangle, then the problem is decidable.

1 Introduction

The design of controllers is one of the most active research topics in the area of hybrid systems. Problems that have been addressed include hierarchical control [20, 5], distributed control [18], and optimal control using dynamic programming techniques [4, 3, 25, 21] or extensions of the maximum principle [12]. A substantial research effort has also been directed towards solving control problems with reachability specifications, that is designing controllers that guarantee that the state of the system remains in a "good" part of the state space. Such control problems turn out to be very important in applications, and are closely related to the computation of the *reachable states* of a hybrid system and to the concept of *controlled invariance*. The proposed solutions extend game theory methods for purely discrete [27, 22] and purely continuous [2, 15] systems to certain classes of hybrid systems: timed automata [17, 14], rectangular hybrid automata [29] and more general hybrid automata [16, 28].

All of these techniques are concerned with hybrid systems whose continuous state evolves in continuous time, according to differential equations or differential inclusions. Unlike conventional continuous dynamical systems, little attention has been devoted systems where the continuous state evolves in discrete time, according to difference equations. Besides being interesting in their own right, this class of hybrid systems can be used to approximate hybrid systems with differential equations. Indeed, most of the techniques that have been proposed for reachability computations for general continuous dynamics involve some form of discretization of the continuous space [10, 13, 28], followed by a reachability computation on the resulting discrete time system.

In this paper we study the problem of controller synthesis for hybrid systems whose continuous dynamics are given by difference equations under reachability specifications. In Section 2, we formulate the problem, introduce the concepts of controlled invariant set and least restrictive controller, and propose an algorithm for computing them. In Section 3 we review some concepts of mathematical logic and show how the algorithm can be implemented by quantifier elimination. This immediately leads to a semi-decidability result for discrete time systems whose continuous dynamics can be encoded in a decidable theory of the reals. In section 4, we implement the proposed algorithm for discrete-time linear systems with all the sets defined by linear inequalities. The implementation is based on a more efficient method for performing quantifier elimination in the theory of linear constraints using linear programming. We also show that the problem

*Research supported and by ONR under grant N00014-97-1-0946, by DARPA under contract F33615-98-C-3614, and by ARO under MURI DAAH04-96-1-0341.

is decidable when the single-input single-disturbance discrete-time linear system is in canonical controllable form, the input is unbounded, and the safe set is a rectangle. Finally, in Section 5, we illustrate the proposed method with some examples.

2 Discrete Time Systems and Safety Specifications

2.1 Basic Definitions

Let Y be a countable collection of variables and let $\mathbf{Y}$ denote its set of valuations, that is the set of all possible assignments of these variables. We refer to variables whose set of valuations is countable as *discrete* and to variables whose set of valuations is a subset of a Euclidean space $\mathbb{R}^n$ as *continuous*. For a set $\mathbf{Y}$ we use $\mathbf{Y}^c$ to denote the complement of $\mathbf{Y}$, $2^{\mathbf{Y}}$ to denote the set of all subsets of $\mathbf{Y}$, $\mathbf{Y}^*$ to denote the set of all finite sequences of elements of $\mathbf{Y}$, and $\mathbf{Y}^\omega$ to denote the set of all infinite sequences. Since the dynamical systems we will consider will be time invariant we will use $y = \{y[i]\}_{i=0}^N$ to denote sequences. We use $\wedge$ to denote conjunction, $\vee$ to denote disjunction, $\neg$ to denote negation, $\forall$ to denote the universal quantifier, and $\exists$ to denote the existential quantifier.

The dynamics of a discrete time system are characterized by a reset relation that, given the current value of the state and input, returns the possible next states of the system. More formally:

Definition 1 (Discrete Time System (DTS)) *A discrete time system is a collection $H = (X, V, \text{Init}, f)$ consisting of a finite collection of state variables, X, a finite collection of input variables, V, a set of initial states,* $\text{Init} \subseteq \mathbf{X}$ *and a reset relation, $f : \mathbf{X} \times \mathbf{V} \to 2^{\mathbf{X}}$.*

A DTS naturally characterizes a subset of the set of sequences of $\mathbf{X} \times \mathbf{V}$.

Definition 2 (Execution of DTS) *A sequence $\chi = (x, v) \in (\mathbf{X} \times \mathbf{V})^* \cup (\mathbf{X} \times \mathbf{V})^\omega$ is said to be an execution of the discrete time system H if $x[0] \in Init$, and for all $k \geq 0$, $x[k+1] \in f(x[k], v[k])$.*

To ensure that every finite execution can be extended to an infinite execution we impose the assumption that $f(x, v) \neq \emptyset$ for all $x \in \mathbf{X}$, $v \in \mathbf{V}$; we call such a DTS *non-blocking*[1]. We denote the set of all executions of H starting at $x_0 \in \mathbf{X}$ as $\mathcal{E}_H(x_0)$, and the set of all executions of H by $\mathcal{E}_H$. Clearly, $\mathcal{E}_H = \bigcup_{x_0 \in Init} \mathcal{E}_H(x_0)$.

Our goal here is to design controllers for DTS. Assume that we are given a *plant*, modeled as a DTS, and we would like to "steer" it using its input variables, so that its executions satisfy certain properties. We assume that the input variables are partitioned into two classes, $V = U \cup D$. U are assumed to be *control variables*, that is variables whose valuations we can specify at will. D, on the other hand, are assumed to be disturbance variables, over whose valuations we have no control (we say they are determined by the *environment*) and that can potentially disrupt our plans. In this context a controller can be defined as a feedback map.

Definition 3 (Controller) *A controller, C, is a map $C : \mathbf{X}^* \to 2^{\mathbf{U}}$. A controller is called* non-blocking *if $C(x) \neq \emptyset$ for all $x \in \mathbf{X}^*$. A controller is called* memoryless *if for all $x, x' \in \mathbf{X}^*$ ending at the same state we have $C(x) = C(x')$.*

The interpretation is that, given the evolution of the plant state up to now, the controller determines the set of allowable controls for the next transition. With this interpretation in mind, we define the set of *closed loop causal executions* as

$$\mathcal{E}_{H_C} = \{(x, u, d) \in \mathcal{E}_H \mid \forall k \geq 0, u[k] \in C(x{\downarrow_k})\},$$

where $x{\downarrow_k}$ denotes the subsequence of x consisting of its first k elements. Notice that a memoryless controller can be characterized by a map $g : \mathbf{X} \to 2^{\mathbf{U}}$, and its set of closed loop causal executions is simply

$$\mathcal{E}_{H_g} = \{(x, u, d) \in \mathcal{E}_H \mid \forall k \geq 0, u[k] \in g(x[k])\}.$$

Our goal is to use controllers to steer the executions of the plant, so that they satisfy certain desirable properties. In this paper we will restrict our attention to a class of properties known as *safety properties*: given a set $F \subseteq \mathbf{X}$, we would

[1] The condition is only sufficient. It can be easily refined to a necessary and sufficient condition; however, since the emphasis of this paper is controller synthesis, we will not pursue this direction.

like to find a non-blocking controller that ensures that the state stays in F for ever. We will say that a controller C *solves the problem* $(H, \Box F)$, if and only if C is non-blocking and for all $(x, u, d) \in \mathcal{E}_{H_C}$, $x[k] \in F$ for all $k \geq 0$. If such a controller exists we say that the problem $(H, \Box F)$ *can be solved.*

Even though safety properties are not the only properties of interest[2], they turn out to be very useful in applications. Many important problems, such as absence of collisions in transportation systems, mutual exclusion in distributed algorithms, etc., are naturally encoded as safety properties. Fortunately, it can be shown that for this class of properties one can, without loss of generality, restrict attention to memoryless controllers.

Proposition 1 *The problem* $(H, \Box F)$ *can be solved by some controller if and only if it can be solved by a memoryless controller.*

Proof: The *if* part is obvious. For the *only if* part, assume, for the sake of contradiction, that there exists a controller $C : \mathbf{X}^* \to 2^{\mathbf{U}}$ that solves the problem $(H, \Box F)$, but there does not exist a memoryless controller that solves the problem. Therefore there must exist two finite executions $\chi_i = (x_i, u_i, d_i) \in \mathcal{E}_{H_C}$, $i = 1, 2$, ending at the same state, x, at times k_1 and k_2 respectively, such that $\chi_1 \neq \chi_2$, and $C(x_1) \neq C(x_2)$. Moreover, the information about the way in which the state x is reached must be essential for subsequent control decisions. Assume that x is reached via χ_1, but we choose to ignore this fact and apply controls after k_1 as though x were reached via χ_2. Then, there must exist a continuation, χ', such that the concatenation $\chi'_1 = (x'_1, u'_1, d'_1) = \chi_1\chi' \in \mathcal{E}_H$ leaves the set F. In particular, since $\chi_1 \in \mathcal{E}_{H_C}$ and C solves $(H, \Box F)$, there must exist $k > 0$ such that $x'_1[k_1 + k] \notin F$. Notice, however, that the concatenation $\chi'_2 = (x'_2, u'_2, d'_2) = \chi_2\chi'$ is also an element of $\mathcal{E}_H$. Moreover, $\chi_2\chi' \in \mathcal{E}_{H_C}$. But $x'_2[k_2 + k] = x'_1[k_1 + k] \notin F$. This contradicts the assumption that C solves the problem $(H, \Box F)$. ∎

Motivated by Proposition 1, we restrict our attention to memoryless controllers from now on.

2.2 Controlled Invariant Sets and Least Restrictive Controllers

The concept of controlled invariance turns out to be fundamental for the design of controllers for safety specifications [16]. Roughly speaking, a set of states, W, is called controlled invariant if there exists a controller that ensures that all executions starting somewhere in W remain in W for ever. More formally:

Definition 4 (Controlled invariant set) *A set* $W \subseteq \mathbf{X}$ *is called a controlled invariant set of* H *if there exists a non-blocking controller that solves the problem* $(H', \Box W)$, *where* $H' = (X, V, W, f)$ *(the same as* H, *but with* $Init' = W$*).*

We say that the controller that solves the problem $(H', \Box W)$, with $Init' = W$, *renders the set* W *invariant.* A useful and intuitive characterization of the concept of controlled invariance can be given in terms of the operator $\mathrm{Pre} : 2^{\mathbf{X}} \to 2^{\mathbf{X}}$ defined by

$$\mathrm{Pre}(W) = \{x \in W \mid \exists u \in \mathbf{U}\ \forall d \in \mathbf{D},\ f(x, u, d) \cap W^c = \emptyset\}\,.$$

The operator returns the set of states $x \in W$ for which $u \in \mathbf{U}$ can be chosen such that for all choices of $d \in \mathbf{D}$ all states that can be reached from x after one transition are also in W. The following properties of the Pre operator are easy to establish and will be useful in the subsequent discussion.

Proposition 2 *The operator* Pre *has the following properties:*

1. Pre *is contracting, that is for all* $W \subseteq \mathbf{X}$, $\mathrm{Pre}(W) \subseteq W$;
2. Pre *is monotone, that is for all* $W, W' \subseteq \mathbf{X}$ *with* $W \subseteq W'$, $\mathrm{Pre}(W) \subseteq \mathrm{Pre}(W')$; *and,*
3. *A set* $W \subseteq \mathbf{X}$ *is controlled invariant if and only if it is a fixed point of* Pre, *that is if and only if* $\mathrm{Pre}(W) = W$.

Proof: The first part follows from the definition. For the second part, notice that for all $x \in \mathrm{Pre}(W)$ there exists $u \in \mathbf{U}$ such that for all $d \in \mathbf{D}$, $f(x, u, d) \subseteq W$. Since $W \subseteq W'$, this means that for the same u, and for all d, $f(x, u, d) \subseteq W'$, and therefore, $x \in \mathrm{Pre}(W')$.

[2]Other important properties include liveness properties (ensuring that the state eventually reaches a certain set, visits a set infinitely often, etc.), stability, optimality, etc.

We now turn our attention to the third part of the proposition. For the *if* part, assume $\mathrm{Pre}(W) = W$ and consider the memoryless controller

$$g(x) = \begin{cases} \{u \in \mathbf{U} \mid \forall d \in \mathbf{D},\ f(x,u,d) \cap W^c = \emptyset\} & x \in W \\ \mathbf{U} & x \notin W. \end{cases}$$

By construction g is non-blocking. Consider an execution $(x,u,d) \in \mathcal{E}_{H_g}$ with $x[0] \in W$, and assume that for all $0 \leq k' \leq k$, $x[k'] \in W$. Then $x[k+1] \in f(x[k],u[k],d[k]) \subseteq W$ by construction of g. Therefore, $x[k] \in W$ for all $k \geq 0$ by induction.

For the *only if* part, notice that by definition $\mathrm{Pre}(W) \subseteq W$. Assume there exists a non-blocking, memoryless[3] controller, g, that solves the problem $(H', \Box W)$ with $\mathrm{Init}' = W$. Consider an arbitrary $x \in W$ and notice that by assumption there exists $u \in g(x)$ such that for all $d \in \mathbf{D}$ and for all $x' \in f(x,u,d)$, $x' \in W$. Therefore, $x \in \mathrm{Pre}(W)$ and $W \subseteq \mathrm{Pre}(W)$. ∎

Given a set F, a set $W \subseteq F$ is called a maximal controlled invariant subset of F, if it is controlled invariant and it is not a proper subset of any other controlled invariant subset of F.

Lemma 1 *The problem $(H, \Box F)$ can be solved if and only if there exists a unique maximal controlled invariant set, $\hat{W}$, with* $\mathrm{Init} \subseteq \hat{W} \subseteq F$.

Proof: For the *if* part, assume such a $\hat{W}$ exists and consider a non-blocking controller g that renders $\hat{W}$ invariant. Then $\mathcal{E}_{H_g} \subseteq \mathcal{E}_{H'_g}$, since $\mathrm{Init} \subseteq \hat{W}$. Therefore, all $\chi \in \mathcal{E}_{H_g}$ remain for ever in $\hat{W}$, and hence in F.

For the *only if* part, assume the problem $(H, \Box F)$ can be solved by a non-blocking controller g. Consider the set

$$W = \bigcup_{x_0 \in \mathrm{Init}} \bigcup_{k \geq 0} \{x[k] \mid (x,u,d) \in \mathcal{E}_{H_g}(x_0)\}.$$

By definition, $\mathrm{Init} \subseteq W$. Since g solves the problem $(H, \Box F)$, $W \subseteq F$. Moreover, for any $x[0] \in W$ consider an execution (x,u,d), with arbitrary $d \in \mathbf{D}^*$ and $u[k] \in g(x[k])$. Then by definition of W, $x[k] \in W$ for all $k \geq 0$. Therefore the controller g renders the set W invariant.

Summarizing, if the problem $(H, \Box F)$ can be solved, then there exists a controlled invariant set W with $\mathrm{Init} \subseteq W \subseteq F$. To show that there exists a unique maximal such set, let $\mathcal{W}$ be the family of all controlled invariant sets W, with $\mathrm{Init} \subseteq W \subseteq F$, $\mathcal{G}$ be the family of the corresponding non-blocking memoryless controllers that render each element of $\mathcal{W}$ invariant, and $h : \mathcal{W} \to \mathcal{G}$ be the map assigning to each $W \in \mathcal{W}$ its corresponding memoryless controller $g \in \mathcal{G}$. By the above discussion, $\mathcal{W}$ (and hence $\mathcal{G}$) is non-empty. By the well-ordering theorem [23] there exists a well-ordering relation for $\mathcal{G}$. We define the memoryless controller

$$g(x) = \min_{W \in \mathcal{W}} \{h(W) \mid x \in W\},$$

where min is taken according to the order on $\mathcal{G}$. Let

$$\hat{W} = \bigcup_{W \in \mathcal{W}} W.$$

Clearly $\mathrm{Init} \subseteq \hat{W} \subseteq F$. If we show that $\hat{W}$ is also controlled invariant, then the class of controlled invariant sets will be closed under arbitrary unions, and hence possess a unique maximal element. Let $\chi = (x,u,d)$ be an execution of H starting at $x[0] \in \hat{W}$, with arbitrary $d \in \mathbf{D}^*$ and $u[k] \in g(x[k])$. Assume, for the sake of contradiction, that there exists $k \geq 0$ such that $x[k'] \in \hat{W}$ for all $0 \leq k' \leq k$, and $x[k+1] \notin \hat{W}$. Since $x[k] \in \hat{W}$ and $u[k] \in g(x[k])$, there exists $W \in \mathcal{W}$ such that $x[k] \in W$ and $u[k] \in h(W)$. By assumption, $h(W)$ solves the problem $(H', \Box W)$ with $Init' = W$. Therefore $x[k+1] \in W \subseteq \hat{W}$, contradicting the assumption that $x[k+1] \notin \hat{W}$. ∎

Many memoryless controllers may be able to solve a particular problem. Controllers that impose less restrictions on the inputs they allow are in a sense better than controllers that impose more restrictions. For example, controllers that impose fewer restrictions allow one more freedom if additional safety specifications are imposed, or one is asked to optimize the performance of the (safe) closed loop system with respect to other objectives. To quantify this intuitive notion we introduce a partial order on the space of memoryless controllers. We write $g_1 \preceq g_2$ if for all $x \in \mathbf{X}$, $g_1(x) \subseteq g_2(x)$.

[3] Without loss of generality by Proposition 1.

Definition 5 (Least restrictive controller) *A memoryless controller $g : \mathbf{X} \to 2^{\mathbf{U}}$ that solves the problem (H, F) is a called least restrictive if it is maximal among the controllers that solve $(H, \Box F)$ in the partial order defined by $\preceq$.*

Lemma 2 *A controller that renders a set W invariant exists if and only if a unique least restrictive controller that renders W invariant exists.*

Proof: The if part is obvious. For the only if part, assume that given a set W a controller g that renders it invariant exists. Let $\mathcal{G}$ be the collection of all controllers that render W invariant and define $\hat{g} : \mathbf{X} \to 2^{\mathbf{U}}$ as

$$\hat{g}(x) = \bigcup_{g \in \mathcal{G}} g(x).$$

We claim that $\hat{g}$ renders W invariant. Let $\chi = (x, u, d) \in \mathcal{E}_{H_{\hat{g}}}(x_0)$ for some $x_0 \in W$ and assume, for the sake of contradiction that there exists $k \geq 0$ such that $x[k'] \in W$ for all $0 \leq k' \leq k$, but $x[k+1] \notin W$. Now, by definition of $\hat{g}$, there exists $g \in \mathcal{G}$ such that $u(x[k]) \in g(x[k])$. Since g renders W invariant, and $x[k] \in W$, then $x[k+1] \in W$, which contradicts the assumption that $x[k+1] \notin W$. Therefore, the class of controllers that render W invariant is closed under arbitrary unions, and hence possesses a unique maximal element. ∎

Notice that the least restrictive controller that renders a set W invariant must, by definition, allow $\hat{g}(x) = \mathbf{U}$ for all $x \notin W$. Summarizing Lemmas 1 and 2 we have the following.

Theorem 1 *The problem $(H, \Box F)$ can be solved if and only if:*

1. *there exists a unique maximal controlled invariant set $\hat{W}$ with* Init $\subseteq \hat{W} \subseteq F$, *and*
2. *there exists a unique, least restrictive, memoryless controller, $\hat{g}$, that renders $\hat{W}$ invariant.*

Motivated by Theorem 1 we state the controlled invariance problem more formally.

Problem 1 (Controlled Invariance Problem (CIP)) *Given a discrete time system H and a set $F \subseteq \mathbf{X}$ compute the maximal controlled invariant subset of F, $\hat{W}$, the least restrictive controller, $\hat{g}$, that renders it invariant, and test whether* Init $\subseteq \hat{W}$.

2.3 Computation of $\hat{W}$ and $\hat{g}$

We first present a conceptual algorithm for solving CIP for general DTS. Even though there is no straight forward way of implementing this algorithm in the general case, in subsequent sections we show how this can be done for special classes of DTS.

Algorithm 1 (Controlled Invariance Algorithm)

initialization: $W^0 = F$, $W^{-1} = \mathbf{X}$, $l = 0$
while $W^{l-1} \cap (W^l)^c \neq \emptyset$ **do**
  $W^{l+1} = \text{Pre}(W^l)$
  $l = l + 1$
end while
set $\hat{W} = \bigcap_{l \geq 0} W^l$
set $\hat{g}(x) = \begin{cases} \left\{ u \in \mathbf{U} \mid \forall d \in \mathbf{D},\ f(x,u,d) \cap (\hat{W})^c = \emptyset \right\} & x \in \hat{W} \\ \mathbf{U} & x \notin \hat{W} \end{cases}$

Theorem 2 *$\hat{W}$ is the maximal controlled invariant subset of F and $\hat{g}$ is the least restrictive controller that renders $\hat{W}$ invariant.*

Proof: To show that $\hat{W}$ is controlled invariant we show that it is a fixed point of Pre. By definition $\text{Pre}(\hat{W}) \subseteq \hat{W}$. Conversely, consider $x \in \hat{W}$ and assume, for the sake of contradiction that $x \notin \text{Pre}(\hat{W})$. Then for all $u \in \mathbf{U}$ there exists

$d \in \mathbf{D}$ and $x' \in f(x,u,d)$ such that $x' \notin \hat{W}$. Therefore, there is an l such that $x' \notin W^l$. Hence $x \notin \text{Pre}(W^l) = W^{l+1} \supseteq \hat{W}$, which is a contradiction. Therefore $\hat{W} \subseteq \text{Pre}(\hat{W})$.

To show that $\hat{W}$ is maximal, consider a controlled invariant set $W \subseteq F$. Assume, for the sake of contradiction, that there exists $x[0] \in W \setminus \hat{W}$. Therefore, there exists $l \geq 0$ such that $x[0] \notin W^l$. By definition of the operator Pre this implies that either $x[0] \notin W^{l-1}$, or for all $u \in \mathbf{U}$ there exists $d \in \mathbf{D}$ and $x' \in f(x[0],u,d)$ such that $x' \notin W^{l-1}$. In the latter case set $x[1] = x'$. By induction, for all choices of u there exists a finite sequence that leaves $W^0 = F \supseteq W$ after at most l steps. This contradicts the assumption that W is a controlled invariant.

Finally, to show $\hat{g}$ is least restrictive, consider another controller, g, that renders $\hat{W}$ invariant, and assume, for the sake of contradiction, that there exists $x \in \mathbf{X}$ and $u \in g(x) \setminus \hat{g}(x)$. By construction of $\hat{g}$, $x \in \hat{W}$. Since $u \notin \hat{g}(x)$, there exists $d \in \mathbf{D}$ and $x' \in f(x,u,d)$ such that $x' \notin \hat{W}$. This contradicts the assumption that g renders $\hat{W}$ controlled invariant. ∎

To implement the controlled invariance algorithm one needs to be able to:

1. encode sets of states, perform intersection and complementation, and test for emptiness,
2. compute the Pre of a set, and
3. guarantee that a fixed point is reached after a finite number of iterations.

For classes of DTS for which 1 and 2 are satisfied we say that the CIP is *semi-decidable*; if all three conditions are satisfied we say that the CIP is *decidable*. As an example, consider *finite state machines*, that is the class of DTS for which $\mathbf{X}$, $\mathbf{U}$ and $\mathbf{D}$ are finite. In this case, one can encode sets of states, perform intersection, complementation, test for emptiness and compute Pre by enumeration (or other more efficient representations). Moreover, by the monotonicity of W^l and the fact that $\mathbf{X}$ is finite, the algorithm is guaranteed to terminate in a finite number of steps. Therefore, the CIP is decidable for finite state machines.

In subsequent sections we show how the computation can be performed for DTS with state and input taking values in Euclidean space and transition relations given by a certain classes of functions of the state and input. Before we can present the details, however, we need to introduce some notation from mathematical logic.

3 Mathematical Logic and Quantifier Elimination

3.1 Languages, Models and Theories

The following discussion is based on [19]. For a more in depth treatment the reader is referred to [9, 8].

A *language* $\mathcal{L} = \{R_1, \ldots, R_n, f_1, \ldots, f_m, c_0, \ldots, c_l\}$ is a set of symbols separated into *relations*, $R_1, \ldots, R_n$, *functions*, $f_1, \ldots, f_m$, and *constants*, $c_0, \ldots, c_l$. For example, $\mathcal{P} = \{<, +, -, 0, 1\}$ and $\mathcal{R} = \{<, +, -, \cdot, 0, 1\}$ are languages with (binary) relation $<$, (binary) functions $+$, $-$ and $\cdot$, and constants 0 and 1.

Given a language $\mathcal{L}$ and a set of variables $\{x_1, x_2, \ldots, v_1, v_2, \ldots\}$, the *terms* of the language are inductively defined. All the variables and all the constants are terms, and if $t_1, \ldots, t_n$ are terms and f is an n-ary function, $f(t_1, \ldots, t_n)$ is also a term. For instance, if a, b and c are positive integer constants and x_1 and x_2 are variables, $ax_1 - bx_2 + c$ is a term of $\mathcal{P}$ and $ax_1^2 + bx_1x_2 + c$ is a term of $\mathcal{R}$[4].

An *atomic formula* of the language is of the form $t_1 = t_2$ or $R(t_1, \ldots, t_n)$, where R is a n-ary relation and t_i, $i = 1, \ldots, n$ are terms. For example, $ax_1 - bx_2 + c < 0$ is an atomic formula of $\mathcal{P}$ and $ax_1^2 + bx_1x_2 + c = 0$ is an atomic formula of $\mathcal{R}$. *First order formulas* (or simply formulas) are recursively defined from atomic formulas:

1. atomic formulas are formulas,
2. if ϕ, ψ are formulas, then so are $\phi \wedge \psi$ and $\neg\phi$[5],
3. if ϕ is a formula and x is a variable, then $\exists x \mid \phi$ is also a formula[6].

[4] Integer constants are generated inductively by repeatedly adding the constant 1 to itself.
[5] Disjunction, $\phi \vee \psi$, is interpreted as $\neg(\neg\phi \wedge \neg\psi)$.
[6] Universal quantification, $\forall x \mid \phi$, is interpreted as $\neg(\exists x \mid \neg\phi)$.

Formulas defined in a language $\mathcal{L}$ are called $\mathcal{L}$-formulas. For example, $Mx \leq \beta$ is a $\mathcal{P}$ formula, where $M \in \mathbb{Q}^{m \times n}$ and $\beta \in \mathbb{Q}^m$ are constants, and $x = (x_1, \ldots, x_n)$ are variables. This becomes clear if we let $m_{ij} \in \mathbb{Q}$ be the i,j element of M and $\beta_i \in \mathbb{Q}$ be the i element of β and write $Mx \leq \beta$ as

$$\bigwedge_{i=1}^{m} [q_i (m_{i1}x_1 + \ldots + m_{in}x_n - \beta_i) < 0] \vee [q_i (m_{i1}x_1 + \ldots + m_{in}x_n - \beta_i) = 0]$$

where q_i is a positive common denominator of m_{ij}, $j = 1, \ldots, n$ and β_i. With a similar interpretation the following expression is also a $\mathcal{P}$ formula

$$\exists u \, \forall d \mid (Mx \leq \beta) \wedge (MAx + Mbu + Mcd \leq \beta) \tag{1}$$

where A, b, c, M and β are constant matrices with rational coefficients and x, u and d are variables.

The occurrence of a variable in a formula is *free* if it is not within the scope of a quantifier; otherwise it is *bound*. For example, x is free, and u and d are bound in (1). We often write $\phi(x_1, \ldots, x_n)$ to indicate that $x_1, \ldots, x_n$ are the free variables of formula ϕ. A *sentence* is a formula with no free variables.

A *model* of a language $\mathcal{L}$ consists of a non-empty set S and a semantic interpretation of the relations, functions and constants of $\mathcal{L}$. For instance, $(\mathbb{R}, <, +, -, 0, 1)$ and $(\mathbb{R}, <, +, -, \cdot, 0, 1)$ with the usual interpretation for the symbols are models of $\mathcal{P}$ and $\mathcal{R}$ respectively. Every sentence of the language is either true or false for a given model. Every formula, $\phi(x_1, \ldots, x_n)$, of the language defines a subset of S^n, namely the set of the valuations of $x_1, \ldots, x_n$ for which the formula is true. Conversely, we say that a set $Y \subseteq S^n$ is *definable* in $\mathcal{L}$ if there exists a formula in $\phi(x_1, \ldots, x_n)$ in $\mathcal{L}$ such that

$$Y = \{(a_1, \ldots, a_n) \in S^n \mid \phi(a_1, \ldots, a_n)\}$$

Two formulas $\phi(x_1, \ldots, x_n)$ and $\psi(x_1, \ldots, x_n)$ are equivalent in a model, denoted by $\phi \equiv \psi$, if for every valuation $(a_1, \ldots, a_n)$ of $(x_1, \ldots, x_n)$, $\phi(a_1, \ldots, a_n)$ is true if and only if $\psi(a_1, \ldots, a_n)$ is true. Equivalent formulas define the same set.

Every model defines a *theory*, as the set of all sentences which hold in the model. For example, we denote by Lin($\mathbb{R}$) the theory defined by the formulas of $\mathcal{P}$ which are true over $(\mathbb{R}, <, +, -, 0, 1)$; in other words, Lin($\mathbb{R}$) is the theory of linear constraints. We denote by OF($\mathbb{R}$) the theory defined by the formulas of $\mathcal{R}$ which are true over $(\mathbb{R}, <, +, -, \cdot, 0, 1)$; in other words, OF($\mathbb{R}$) is the theory of the real numbers as an ordered field.

3.2 Quantifier Elimination and Semi-decidability

The terminology introduced above provides a framework for defining sets of states, by using formulas in an appropriate theory. It also provides a method for performing intersection and complementation of sets, by taking conjunction and negation of the corresponding formulas. One would also like to be able to determine whether a set definable in the model is empty or not.

For some theories, it is possible to determine the sentences that belong to the theory. The Tarski-Seidenberg decision procedure provides a way of doing this for OF($\mathbb{R}$). It can be shown [26, 24] that OF($\mathbb{R}$) is decidable, in other words, there exists a computational procedure that after a finite number of steps determines whether a $\mathcal{R}$-sentence belongs to OF($\mathbb{R}$) or not. The decision procedure is based on quantifier elimination. An algorithm is provided that converts a formula $\phi(x_1, \ldots, x_n)$ to an equivalent quantifier free formula $\psi(x_1, \ldots, x_n)$. Notice that this provides a method for testing emptiness. A set $Y = \{(x_1, \ldots, x_n) | \phi(x_1, \ldots, x_n)\}$ is empty if and only if the sentence $\exists x_1 \ldots \exists x_n | \phi(x_1, \ldots, x_n)$ is equivalent to false.

To relate this to the problem at hand, we restrict our attention to CIP which are "definable" in an appropriate theory.

Definition 6 (Definable CIP) *A CIP, $(H, \Box F)$, is definable in a theory if $\mathbf{X} = \mathbb{R}^n$, $\mathbf{U} \subseteq \mathbb{R}^{n_u}$, $\mathbf{D} \subseteq \mathbb{R}^{n_d}$ and the sets* $\mathbf{U}$, $\mathbf{D}$, Init, *$f(x, u, d)$ for all $x \in \mathbf{X}$, $u \in \mathbf{U}$, $d \in \mathbf{D}$, and F are definable in the theory.*

If $(H, \Box F)$ and W^l are definable in OF($\mathbb{R}$), then

$$\psi^l(x) \equiv \exists u \, \forall d \, \forall x' \mid [x \in W^l] \wedge [u \in \mathbf{U}] \wedge [(d \notin \mathbf{D}) \vee (x' \notin f(x, u, d)) \vee (x' \in W^l)] \tag{2}$$

is a first order formula in the corresponding language. Therefore, each step of the controlled invariance algorithm involves eliminating the quantifiers in (2) to obtain a quantifier free formula defining W^{l+1}, complementing the resulting set W^{l+1}, intersecting $(W^{l+1})^c$ with W^l and testing the intersection for emptiness. The fact that OF($\mathbb{R}$) is decidable immediately leads to the following.

Theorem 3 *For the class of DTS definable in* OF($\mathbb{R}$) *the CIP is semi-decidable.*

Moreover, if $(H, \Box F)$ is definable in OF($\mathbb{R}$) and W is a controlled invariant set also definable in OF($\mathbb{R}$), then the set

$$\{(x,u) \mid \forall d \in \mathbf{D}\ \forall x' \in f(x,u,d),\ x' \in W\}$$

describing the least restrictive controller that renders W invariant is also definable in OF($\mathbb{R}$). Quantifier elimination can be performed in this formula, to obtain an explicit expression for the least restrictive controller. Finally, the question $W \cap \mathrm{Init}^c = \emptyset$ can be decided. Therefore, if the algorithm happens to terminate in a finite number of steps, the CIP problem can be completely solved.

The class of CIP definable in OF($\mathbb{R}$) is fairly broad. The class contains DTS with polynomial constraints on u, d and Init and reset relations encoded by constraints on the possible next states encoded by polynomials in x, u and d. Strictly speaking the problem remains semi-decidable even if we add polynomial state dependent input constraints, i.e. at each state, x, allow values of u and d that satisfy polynomial constraints in x, u and d. This includes for example the closed loop system obtained by coupling the least restrictive controller with the plant.

Although different methods have been proposed for performing quantifier elimination [26, 24, 1] in OF($\mathbb{R}$), and the process can be automated using symbolic tools [11], the quantifier elimination procedure is in general hard, both in theory and in practice. For the theory Lin($\mathbb{R}$) a somewhat more efficient implementation can be derived using techniques from linear algebra and linear programming. The next section shows how quantifier elimination in the theory Lin($\mathbb{R}$) can be performed more efficiently for the formula (2) used in the controlled invariance algorithm.

4 CIP for Discrete Time Linear Systems

A *linear CIP* (LCIP) consists of

- is a Linear DTS (LDTS), that is a DTS with $\mathbf{X} = \mathbb{R}^n$, $\mathbf{U} = \{u \in \mathbb{R}^{n_u} \mid Eu \leq \eta\} \subseteq \mathbb{R}^{n_u}$, $\mathbf{D} = \{d \in \mathbb{R}^{n_d} \mid Gd \leq \gamma\} \subseteq \mathbb{R}^{n_d}$, Init $= \{x \in \mathbf{X} \mid Jx \leq \theta\}$ and a reset relation given by $f(x,u,d) = \{Ax + Bu + Cd\}$, where $A \in \mathbb{Q}^{n \times n}$, $B \in \mathbb{Q}^{n \times n_u}$, $C \in \mathbb{Q}^{n \times n_d}$, $E \in \mathbb{Q}^{m_u \times n_u}$, $G \in \mathbb{Q}^{m_d \times n_d}$, $\eta \in \mathbb{Q}^{m_u}$, $\gamma \in \mathbb{Q}^{m_d}$, $J \in \mathbb{Q}^{n \times m_i}$ and $\theta \in \mathbb{Q}^{m_i}$ with m_u, m_d and m_i being the number of constraints on the control, disturbance and initial conditions respectively; and,
- a set $F = \{x \in \mathbb{R}^n \mid Mx \leq \beta\}$ where $M \in \mathbb{Q}^{m \times n}$, $\beta \in \mathbb{Q}^m$ and m is the number of constraints on the state.

Notice that LDTS are non-blocking and deterministic, in the sense that for every state x and every input (u,d) there exists a unique next state. Since the sets F, $\mathbf{U}$ and $\mathbf{D}$ are all convex polygons, and the dynamics f are given by a linear map, the LCIP is definable in the theory Lin($\mathbb{R}$), and therefore, according to the discussion in Section 3, is semi-decidable. We assume that the sets F and $\mathbf{U}$ can be either bounded or unbounded, but $\mathbf{D}$ is bounded[7].

For the LCIP it turns out that, after the l-th iteration, the set W^l can be described as $\{x \in \mathbb{R}^{m^l \times n} \mid M^l x \leq \beta^l\}$, that is, W^l remains a convex polygon. Obviously, $m^0 = m$, $M^0 = M$ and $\beta^0 = \beta$. Letting $\hat{A}^l = M^l A$, $\hat{B}^l = M^l B$ and $\hat{C}^l = M^l C$, equation (2) becomes

$$\begin{aligned} \psi^l(x) &\equiv \exists u\, \forall d \mid [M^l x \leq \beta^l] \wedge [Eu \leq \eta] \wedge [(Gd > \gamma) \vee (\hat{A}^l x + \hat{B}^l u + \hat{C}^l d \leq \beta^l)] \\ &\equiv [M^l x \leq \beta^l] \wedge [\exists u \mid (Eu \leq \eta) \wedge (\forall d \mid (Gd > \gamma) \vee (\hat{A}^l x + \hat{B}^l u + \hat{C}^l d \leq \beta^l))] \end{aligned}$$

Thus, in each step of the algorithm, we need to be able to eliminate variables u and d from the inner formulae, intersect the new constraints with the old ones and check if the new set is empty. Notice that not all of the new constraints generated by quantifier elimination may be necessary to define the set W^{l+1}. Also, some of the old constraints may become redundant after adding the new ones. Hence we need to check the redundancy of the constraints when doing the intersection.

[7] The theoretical discussion can be extended to unbounded $\mathbf{D}$ sets, but the computational implementation is somewhat more involved.

4.1 Quantifier Elimination

We first perform quantifier elimination on d over the formula

$$\phi^l(x,u) \equiv \forall d \mid (Gd > \gamma) \vee (\hat{A}^l x + \hat{B}^l u + \hat{C}^l d \leq \beta^l)\,.$$

Let $\hat{a}_i^T$, $\hat{b}_i^T$ and $\hat{c}_i^T$ be the i-th row of $\hat{A}^l$, $\hat{B}^l$ and $\hat{C}^l$, respectively. Then, parsing ϕ^l leads to

$$\phi^l(x,u) \equiv \forall d \mid \bigwedge_{i=1}^{m^l} (Gd > \gamma) \vee (\hat{a}_i^T x + \hat{b}_i^T u + \hat{c}_i^T d \leq \beta_i^l) \equiv \forall d \mid \bigwedge_{i=1}^{m^l} (Gd > \gamma) \vee (\hat{c}_i^T d \leq \beta_i^l - \hat{a}_i^T x - \hat{b}_i^T u)$$

Consider $\delta : \mathbb{R}^{m \times n_d} \rightarrow \mathbb{R}^m$ defined by $\delta_i(\hat{C}^l) = \max_{d:Gd \leq \gamma} (\hat{c}_i^T d)$ for $i = 1, \ldots, m$.

Proposition 3 *$\phi^l(x,u)$ is equivalent to $\varphi^l(x,u) \equiv \hat{A}^l x + \hat{B}^l u \leq \beta^l - \delta(\hat{C}^l)$*

Proof: If $\phi^l(x,u)$ is false then $\exists d^* \mid (Gd^* \leq \gamma) \wedge (\hat{c}_i^T d^* > \beta_i^l - \hat{a}_i^T x - \hat{b}_i^T u)$ for some i. Since $\delta_i(\hat{C}^l) \geq \hat{c}_i^T d^*$, we conclude that $\neg\phi^l \Rightarrow \neg\varphi^l$, hence $\varphi^l \Rightarrow \phi^l$. Similarly, if $\varphi^l(x,u)$ is false, $\exists d^* \mid (Gd^* \leq \gamma) \wedge (\hat{c}_i^T d^* > \beta_i^l - \hat{a}_i^T x - \hat{b}_i^T u)$ for some i. Thus ϕ^l is false and $\phi^l \Rightarrow \varphi^l$. ■

The elimination of the $\forall$ quantifier can be done by solving a collections of linear programming problems. Since we have assumed that $\mathbf{D}$ is bounded, such an optimization problem is guaranteed to have a solution, and hence $\delta(\cdot)$ is well defined. Since $\delta(\cdot)$ is applied to each row of $\hat{C}^l$, in the sequel we will use $\delta_i(\hat{C}^l)$ and $\delta(\hat{c}_i^T)$ interchangeably. Notice that, strictly speaking, $\delta(\cdot)$ is not part $\mathrm{Lin}(\mathbb{R})$, but we use it as a shorthand for the constant obtained by solving the linear programs.

Next, we perform quantifier elimination on u over the formula

$$\phi^l(x) = \exists u \mid (Eu \leq \eta) \wedge (\hat{A}^l x + \hat{B}^l u \leq \beta^l - \delta(\hat{C}^l)) \equiv \exists u \mid \begin{pmatrix} \hat{A}^l & \hat{B}^l \\ 0 & E \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} \leq \begin{pmatrix} \beta^l - \delta(\hat{C}^l) \\ \eta \end{pmatrix} \tag{3}$$

We will discuss two methods to eliminate u: the first is known as *Fourier Elimination*, and the second, attributed to Cernikov [6], is an application of Farkas Lemma on duality [7].

For the first method, assume we want to eliminate u_1 first. Let e_i be the i-th unit vector in $\mathbb{R}^{m+m_u}$,

$$H^l = \begin{pmatrix} \hat{B}^l \\ E \end{pmatrix} \quad \text{and} \quad \xi^l(x) = \begin{pmatrix} \beta^l - \delta(\hat{C}^l) - \hat{A}^l x \\ \eta \end{pmatrix}.$$

Thus $\phi^l(x) \equiv \exists u \mid H^l u \leq \xi^l(x)$. Also define $P^l = \{p \mid H_{p1}^l > 0\}$, $Q^l = \{q \mid H_{q1}^l < 0\}$ and $R^l = \{r \mid H_{r1}^l = 0\}$, where H_{ij}^l refers to the i,j element of the matrix H^l. Then $\phi^l(x)$ is equivalent to

$$\exists u \mid \bigwedge_{p \in P^l} \bigwedge_{q \in Q^l} \left[\frac{1}{H_{q1}^l} (\xi_q^l(x) - \sum_{j=2}^{m} H_{qj}^l u_j) \leq u_1 \leq \frac{1}{H_{p1}^l} (\xi_p^l(x) - \sum_{j=2}^{m} H_{pj}^l u_j) \right] \wedge \bigwedge_{r \in R^l} \left[0 \leq (\xi_r^l(x) - \sum_{j=2}^{m} H_{rj}^l u_j) \right]$$

Therefore, after the elimination of u_1 we obtain

$$\exists u \mid \bigwedge_{\substack{p \in P^l \\ q \in R^l}} \bigwedge_{q \in Q^l} (H_{p1}^l - H_{q1}^l) \begin{pmatrix} \hat{e}_q^T \\ \hat{e}_p^T \end{pmatrix} \begin{pmatrix} \hat{A}^l x \\ 0 \end{pmatrix} \leq (H_{p1}^l - H_{q1}^l) \begin{pmatrix} \hat{e}_q^T \\ \hat{e}_p^T \end{pmatrix} \begin{pmatrix} \beta^l - \delta(\hat{C}^l) \\ \eta \end{pmatrix} - (H_{p1}^l - H_{q1}^l) \begin{pmatrix} \sum_{j=2}^m H_{qj}^l u_j \\ \sum_{j=2}^m H_{pj}^l u_j \end{pmatrix} \tag{4}$$

Therefore, the elimination of the $\exists$ quantifier is performed by taking nonnegative linear combinations of all pairs of constraints so as to cancel the quantified variable. Note that if all the coefficients of the quantified variable are positive (negative), then ϕ^l is true, and we need not eliminate the remaining variables. Otherwise, after u_1 has been eliminated, we apply the same procedure to the constraints in (4), so as to eliminate $u_2, \ldots, u_{n_u}$. Since the procedure is based on nonnegative row operations, it is clear that

$$\phi^l(x) \equiv \Lambda^l \begin{pmatrix} \hat{A}^l x \\ 0 \end{pmatrix} \leq \Lambda^l \begin{pmatrix} \beta^l - \delta(\hat{C}^l) \\ \eta \end{pmatrix} \equiv \tilde{M}^l x \leq \tilde{\beta}^l\,. \tag{5}$$

where $\Lambda^l \in \mathbb{Q}^{\tilde{m}^l \times (m+m_u)}$ is a matrix with nonnegative entries such that $\Lambda^l H^l = 0$, $\tilde{m}^l$ is the number of constraints obtained through quantifier elimination, $\tilde{M}^l \in \mathbb{Q}^{\tilde{m}^l \times n}$ and $\tilde{\beta}^l \in \mathbb{Q}^{\tilde{m}^l}$.

Although *Fourier Elimination* is attractive because of its simplicity, it is quite inefficient. In general, it generates many new constraints, and in the worst case the method is exponential. This difficulty can be partially remedied since many of the inequalities are likely to be redundant [7]. An alternative method [6] gives the same result, but computes Λ^l in a different way. In fact, the rows of Λ^l are the extreme points of the set $\{\lambda^l \in \mathbb{R}^{m+m_u} \mid \lambda^{l^T} H^l = 0 \wedge \lambda^l \geq 0\}$. By extreme points we mean points in the boundary of the first orthant of $\mathbb{R}^{m+m_u}$ with $m + m_u - rank(H^l) - 1$ components set to zero.

4.2 Intersection, Emptiness and Redundancy

The quantifier elimination presented above allows us to compute the set of states that can be forced by u to transition into W^l as $\left\{x \mid \tilde{M}x \leq \tilde{\beta}\right\}$. To obtain W^{l+1} this set must be intersected with W^l itself. Since both sets are convex the intersection can be carried out by simply appending $\tilde{M}$ and $\tilde{\beta}$ to M^l and β^l respectively. This however is likely to lead to a much larger description than necessary, since many of the new resulting constraints may redundant (implied by other constraints). We propose the following algorithms eliminating the redundant constraints and checking the resulting set for emptiness.

Algorithm 2 (Redundancy Algorithm)

initialization $M' = \begin{pmatrix} \tilde{M}^l \\ M^l \end{pmatrix}$, $\beta' = \begin{pmatrix} \tilde{\beta}^l \\ \beta^l \end{pmatrix}$, $M^{l+1} = []$, $\beta^{l+1} = []$, $c \in \mathbb{Q}^n$ arbitrary.
$m^* = \max\{c^T x \mid M'x \leq \beta\}$
if problem infeasible
  $\hat{W} = \emptyset$
  terminate controlled invariance algorithm
else
  for $i = 1$ **to** $\tilde{m}^l + m$ **do**
    $m_i'^T = i$-th row of M'
    $\beta_i' = i$-th row of β'
    remove m_i' from M' and β_i' from β'
    $m^* = \max\{m_i'^T x \mid M'x \leq \beta'\}$
    if $m^* > \beta_i'$
      add $m_i'^T$ to M^{l+1} and β_i' to β^{l+1}
      add $m_i'^T$ to M' and β_i' to β'
    end if
  end for
end if
if $M^{l+1} = M^l$ and $\beta^{l+1} = \beta$
  $\hat{W} = W^l$
  terminate controlled invariance algorithm
end if

In the above [] denotes an empty matrix. The idea behind the algorithm is that $W^l \cap \tilde{W}^l = \emptyset$ if and only if the optimization problem maximize $c^T x$ subject to $M'x \leq \beta$ is infeasible for all $c \in \mathbb{Q}^n$. If, in particular, the problem maximize $m_i'^T x$ subject to $M'x \leq \beta$ is feasible, and the constraint $m_i'^T x \leq \beta_i'$ is not redundant, then the optimal value is β_i'. Moreover, by removing the non-redundant constraint $m_i'^T x \leq \beta_i'$ from the optimization problem, the new optimal value m^* satisfies $m^* > \beta_i'$.

The controlled invariance algorithm terminates if the redundancy algorithm concludes that either $W^l \cap \tilde{W}^l = \emptyset$ (in which case $\hat{W} = \emptyset$), or all the new constraints are redundant (in which case $W^l = W^{l+1} = \hat{W}$)[8]. Otherwise, upon termination of the redundancy algorithm, the process is repeated for W^{l+1}. An obvious optimization of the code involves terminating both algorithms if, after all new constraints in $\tilde{M}x \leq \tilde{b}$ have been tested, M^{l+1} and β^{l+1} are still empty.

[8]Note that any redundant constraints in the original description of F will be eliminated the first time the redundancy algorithm is invoked by the controlled invariance algorithm.

The proposed implementation shows that for all l the set W^l is a convex polygon as claimed. Summarizing:

Theorem 4 *The LCIP problem is semi-decidable.*

In the next section we study situations where the algorithm is guaranteed to terminate in a finite number of steps. In section 5, we will provide and example which actually converges after an infinite number of iterations.

4.3 Decidable Special Cases

We first summarize some of the observations made so far about situations where the algorithm terminates in a finite number of steps.

Proposition 4 *For an LCIP with* $\mathbf{U} = \mathbb{R}^{n_u}$, *if either one of the columns of* MB *does not change sign, or if* $rank(MB) = \min\{m, n\}$, *the algorithm terminates in a finite number of steps.*

Proof: If one of the columns of MB does not change sign, there is no $\lambda \geq 0$ such that $\lambda^T MB = 0$. Thus $\hat{W} = F$, $g(x) = \mathbf{U}$ for all $x \in \mathbf{X}$, and the algorithm terminates in the first iteration. If $rank(MB) = m$, the only solution of $\lambda^T MB = 0$ is $\lambda = 0$, thus we obtain the same result as in the previous case. Finally, if $rank(MB) = n$, then $\lambda^T MB = 0 \Rightarrow \lambda^T MA = 0$. Thus, if $\lambda^T(\beta - \delta(MC)) \geq 0$, we have $\hat{W} = F$ and $\hat{g}(x) = \mathbf{U}$ for all $x \in \mathbf{X}$; otherwise, we have $\hat{W} = \emptyset$ and $\hat{g}(x) = \mathbf{U}$ for all $x \in \mathbf{X}$. Again, the algorithm terminates in the first iteration. ∎

Next, we limit our attention to the case where $F = [\alpha_1, \beta_1] \times \ldots \times [\alpha_n, \beta_n] \subset \mathbb{R}^n$ with $\alpha_i \leq \beta_i$ and $[\alpha_i, \beta_i] \subset \mathbb{R}, i = 1 \ldots n$, $u \in \mathbb{R}$, and $d \in [d_1, d_2] \subset \mathbb{R}$. To remind ourselves of the fact that u and d are scalar, we use b and c instead of B and C. We also assume that (A, b) is in canonically controllable form, that is

$$x[k+1] = \begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & & & & & 1 \\ a_{n1} & a_{n2} & & \cdots & & a_{nn} \end{pmatrix} x[k] + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} u[k] + \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} d[k] \tag{6}$$

In this case $\psi^1(x)$ is equivalent to

$$\exists u \mid \bigwedge_{j=1}^{n} (\alpha_j \leq x_j \leq \beta_j) \wedge \bigwedge_{j=2}^{n} (\alpha_{j-1} - \delta(-c_{j-1}) \leq x_j \leq \beta_{j-1} - \delta(c_{j-1})) \wedge$$
$$\left(\alpha_n - \sum_{j=1}^{n} a_{nj} x_j - \delta(-c_n) \leq u \leq \beta_n - \sum_{j=1}^{n} a_{nj} x_j - \delta(c_n) \right)$$

From the last expression, it is clear that given $x_1 \in [\alpha_1, \beta_1]$, $x_j, j = 2 \ldots n$ exists if and only if

$$\alpha_j^1 = \max(\alpha_j, \alpha_{j-1} - \delta(-c_{j-1})) \quad \leq \quad \min(\beta_j, \beta_{j-1} - \delta(c_{j-1})) = \beta_j^1, \qquad j = 2 \ldots n$$

and u exists if and only if

$$\alpha_n - \delta(-c_n) \leq \beta_n - \delta(c_n)$$

Thus, after one iteration of the algorithm, variables $x_j, j = 2 \ldots n$ get restricted, which means that we have to do another iteration. It is straightforward to see that in the l-th iteration ($1 \leq l \leq n$) W^l is defined by:

$$W^l \quad = \quad [\alpha_1^0, \beta_1^0] \times [\alpha_2^1, \beta_2^1] \times \ldots \times [\alpha_l^{l-1}, \beta_l^{l-1}] \times [\alpha_{l+1}^l, \beta_{l+1}^l] \times [\alpha_{l+2}^l, \beta_{l+2}^l] \times \ldots \times [\alpha_n^l, \beta_n^l]$$

where

$$\alpha_j^0 = \alpha_j, \qquad \alpha_j^l = \max(\alpha_j^{l-1}, \alpha_{j-1}^{l-1} - \delta(c_{j-1})) \qquad l+1 \leq j \leq n$$
$$\beta_j^0 = \beta_j, \qquad \beta_j^l = \min\,(\beta_j^{l-1}, \beta_{j-1}^{l-1} - \delta(c_{j-1})) \qquad l+1 \leq j \leq n$$

This means that after n iterations, the maximal controlled invariant set remains unchanged, and the least restrictive controller is given by the last constraint of equation (7), but with α_n, β_n replaced by $\alpha_n^{n-1}, \beta_n^{n-1}$. This result can be summarized as follows:

Lemma 3 *Given system (6) with* $F = [\alpha_1, \beta_1] \times \ldots \times [\alpha_n, \beta_n] \subset \mathbb{R}^n$, $\mathbf{U} = \mathbb{R}$ *and* $\mathbf{D} = [d_1, d_2] \subset \mathbb{R}$, *the solution to the CIP problem, obtained after at most n iterations of the algorithm, is given by:*

$$\hat{W} = \begin{cases} \left\{ (x \mid \bigwedge_{j=1}^{n} \alpha_j^{j-1} \leq x_j \leq \beta_j^{j-1} \right\} & \text{if } \alpha_j^{j-1} \leq \beta_j^{j-1}, j = 2 \ldots n \wedge |c_n|(d_2 - d_1) \leq \beta_n^{n-1} - \alpha_n^{n-1} \\ \emptyset & \text{otherwise} \end{cases}$$

$$\hat{g}(x) = \begin{cases} \left\{ u \mid \alpha_n^{n-1} - \sum_{j=1}^{n} a_{nj} x_j - \delta(-c_n) \leq u \leq \beta_n^{n-1} - \sum_{j=1}^{n} a_{nj} x_j - \delta(c_n) \right\} & \text{if } x \in \hat{W} \\ \mathbf{U} & \text{otherwise} \end{cases}$$

Theorem 5 *For systems of the form (6) with* $F = [\alpha, \beta] = [\alpha_1, \beta_1] \times \ldots \times [\alpha_n, \beta_n] \subset \mathbb{R}^n$, $\mathbf{U} = \mathbb{R}$ *and* $\mathbf{D} = [d_1, d_2] \subset \mathbb{R}$, *the LCIP problem is decidable.*

Note that if at the first iteration, we compute

$$\begin{aligned} \alpha_1^1 &= \alpha_1, \quad & \alpha_j^1 &= \max(\alpha_j, \alpha_{j-1}^1 - \delta(c_{j-1})) \quad & 2 \leq j \leq n \\ \beta_1^1 &= \beta_1, \quad & \beta_j^1 &= \min(\beta_j, \beta_{j-1}^1 - \delta(c_{j-1})) \quad & 2 \leq j \leq n \end{aligned}$$

then the problem can be solved in one iteration.

The above conditions for decidability are somewhat demanding. If, for example, u is bounded, that is, $u \in \mathbf{U} = [u_1, u_2] \subset \mathbb{R}$, then the new constraints added to x during each iteration may change the bounds on x to a non-rectangular polyhedron. For example, in the first iteration, the following constraints are added to x:

$$\left(\alpha_n - \sum_{j=1}^{n} a_{nj} x_j - \delta(-c_n) \leq u_2 \right) \bigwedge \left(u_1 \leq \beta_n - \sum_{j=1}^{n} a_{nj} x_j - \delta(c_n) \right)$$

In this case, the CIP problem is no longer decidable, and the system falls into the more general class of systems described in section 4. We conjecture that the LCIP is decidable in a much more general setting, using a completely different algorithm that exploits the stabilizability of the pairs (A, B) and (A, C) and the observability of the pair (A, M).

5 Experimental Results

The algorithm proposed above was implemented in MATLAB. In this section, we present three examples that were solved using this implementation. The first two examples are also worked out analytically to validate the MATLAB program.

5.1 Example 1

The linear system is defined by

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, C = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, M = \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{pmatrix}, \beta = \begin{pmatrix} 80 \\ 40 \\ 80 \\ 70 \end{pmatrix} \tag{7}$$

with $u[k] \in \mathbf{U} = \mathbb{R}$ and $d[k] \in \mathbf{D} = [-1, 1]$.

1. Initialization $M^0 = M$, $\beta^0 = \beta$.
2. Iteration 1
   (a) Computing $\hat{A}$, $\hat{b}$, $\hat{c}$

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -1 & -2 \\ -1 & 0 \\ 1 & 0 \end{pmatrix} \quad \hat{b} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} \quad \hat{c} = \begin{pmatrix} 2 \\ -2 \\ 0 \\ 0 \end{pmatrix}$$

(b) Computing possible new constraints. Here $P^1 = \{1,4\}$, $Q^1 = \{2,3\}$ and $R^1 = \emptyset$

$$\begin{aligned}
\begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} -1 & -2 \\ 1 & 2 \end{pmatrix}x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 40 \\ 80 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 2 \\ 2 \end{pmatrix} \\
&\Rightarrow 0 \leq 116 \Rightarrow \text{Redundant} \\
\begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix}x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 80 \\ 80 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 2 \end{pmatrix} \\
&\Rightarrow 2x_2 \leq 158 \Rightarrow \text{Redundant} \\
\begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} -1 & -2 \\ 1 & 0 \end{pmatrix}x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 40 \\ 70 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 2 \\ 0 \end{pmatrix} \\
&\Rightarrow -2x_2 \leq 108 \Rightarrow \text{Not Redundant} \\
\begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} -1 & -2 \\ 1 & 2 \end{pmatrix}x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 40 \\ 80 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 2 \\ 2 \end{pmatrix} \\
&\Rightarrow 0 \leq 150 \Rightarrow \text{Redundant}
\end{aligned}$$

(c) Computing new M and β

$$M^1 = \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 0 & -2 \end{pmatrix} \qquad \beta^1 = \begin{pmatrix} 80 \\ 40 \\ 80 \\ 70 \\ 108 \end{pmatrix}$$

3. Iteration 2

(a) Computing $\hat{A}$, $\hat{b}$, $\hat{c}$

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -1 & -2 \\ -1 & 0 \\ 1 & 0 \\ -2 & -2 \end{pmatrix} \qquad \hat{b} = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ -2 \end{pmatrix} \qquad \hat{c} = \begin{pmatrix} 2 \\ -2 \\ 0 \\ 0 \\ -2 \end{pmatrix}$$

(b) Computing possible new constraints. Here $P^2 = \{1,4\}$, $Q^2 = \{5\}$ and $R^2 = \emptyset$

$$\begin{aligned}
\begin{pmatrix} 1 & 2 \end{pmatrix}\begin{pmatrix} -2 & -2 \\ 1 & 2 \end{pmatrix}x &\leq \begin{pmatrix} 1 & 2 \end{pmatrix}\begin{pmatrix} 108 \\ 80 \end{pmatrix} - \begin{pmatrix} 1 & 2 \end{pmatrix}\begin{pmatrix} 2 \\ 2 \end{pmatrix} \\
&\Rightarrow 2x_2 \leq 266 \Rightarrow \text{Redundant} \\
\begin{pmatrix} 1 & 2 \end{pmatrix}\begin{pmatrix} -2 & 2 \\ 1 & 2 \end{pmatrix}x &\leq \begin{pmatrix} 1 & 0 \end{pmatrix}\begin{pmatrix} 108 \\ 70 \end{pmatrix} - \begin{pmatrix} 1 & 2 \end{pmatrix}\begin{pmatrix} 2 \\ 0 \end{pmatrix} \\
&\Rightarrow -2x_2 \leq 246 \Rightarrow \text{Redundant}
\end{aligned}$$

(c) Therefore $\hat{W}$ and $\hat{g}(x)$ converge to

$$\hat{W} = \left\{ x \mid \begin{pmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 0 & -2 \end{pmatrix} x \leq \begin{pmatrix} 80 \\ 40 \\ 80 \\ 70 \\ 108 \end{pmatrix} \right\}$$

$$\hat{g}(x) = \begin{cases} \{u \in \mathbf{U} \mid u \geq \max(-38 - x_1 - 2x_2, -80 - x_1, -52 - x_1 - x_2) \\ \qquad u \leq \min(78 - x_1 - 2x_2, 70 - x_1)\} & \text{if } x \in \hat{W} \\ \mathbf{U} & \text{otherwise} \end{cases}$$

Figure 1 shows the plot obtained by MATLAB.

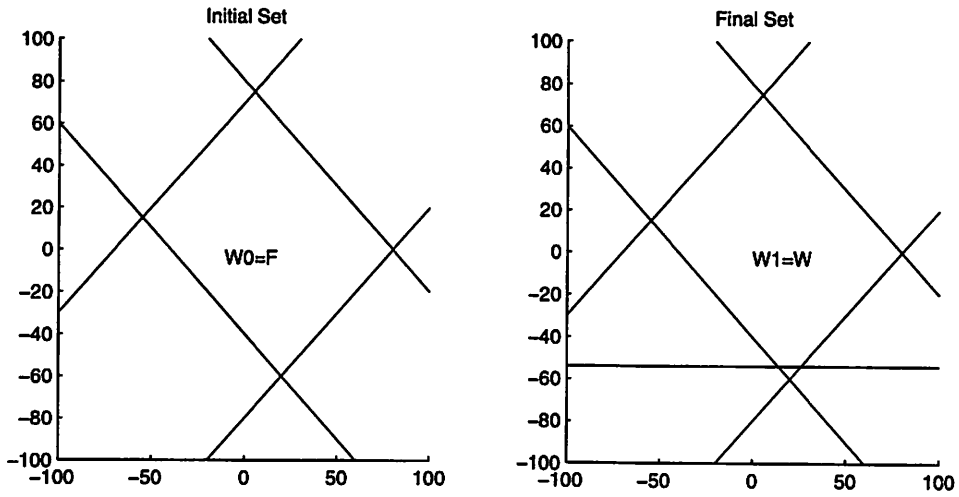

Figure 1: Iterations of the algorithm for Example 1

5.2 Example 2

In this example, we use the system defined in example 1, but with

$$M = \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & 1 \end{pmatrix} \qquad \beta = \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \end{pmatrix}$$

1. Initialization $M^0 = M$, $\beta^0 = \beta$.
2. Iteration 1
   (a) Computing $\hat{A}$, $\hat{b}$, $\hat{c}$

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -3 & -4 \\ -1 & 0 \\ 1 & -2 \end{pmatrix} \qquad \hat{b} = \begin{pmatrix} 1 \\ -3 \\ -1 \\ 1 \end{pmatrix} \qquad \hat{c} = \begin{pmatrix} 2 \\ -4 \\ 0 \\ -2 \end{pmatrix}.$$

   (b) Computing possible new constraints. Here $P^1 = \{1,4\}$, $Q^1 = \{2,3\}$ and $R^1 = \emptyset$

$$\begin{aligned}
\begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} -3 & -4 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} -50 \\ 100 \end{pmatrix} - \begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} 4 \\ 2 \end{pmatrix} \\
&\Rightarrow \quad 2x_2 \leq 240 \Rightarrow \text{Redundant} \\
\begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} -1 & 0 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 100 \\ 100 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 2 \end{pmatrix} \\
&\Rightarrow \quad 2x_2 \leq 198 \Rightarrow \text{Redundant} \\
\begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} -3 & -4 \\ 1 & -2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} -50 \\ -50 \end{pmatrix} - \begin{pmatrix} 1 & 3 \end{pmatrix}\begin{pmatrix} 4 \\ 2 \end{pmatrix} \\
&\Rightarrow \quad -10x_2 \leq -210 \Rightarrow \text{Not Redundant} \\
\begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} -1 & 0 \\ 1 & -2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 100 \\ -50 \end{pmatrix} - \begin{pmatrix} 1 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 2 \end{pmatrix} \\
&\Rightarrow \quad -2x_2 \leq 48 \Rightarrow \text{Redundant}
\end{aligned}$$

(c) Computing new M and β

$$M^1 = \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & -1 \\ 0 & -10 \end{pmatrix} \qquad \beta^1 = \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \\ -210 \end{pmatrix}$$

3. Iteration 2

(a) Computing $\hat{A}$, $\hat{b}$, $\hat{c}$

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -3 & -4 \\ -1 & 0 \\ 1 & -2 \\ -10 & -10 \end{pmatrix} \qquad \hat{b} = \begin{pmatrix} 1 \\ -3 \\ -1 \\ 1 \\ -10 \end{pmatrix} \qquad \hat{c} = \begin{pmatrix} 2 \\ -4 \\ 0 \\ -2 \\ -10 \end{pmatrix}$$

(b) Computing possible new constraints. Here $P^2 = \{1, 4\}$, $Q^2 = \{5\}$ and $R^2 = \emptyset$

$$\begin{aligned} \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} -10 & -10 \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} -210 \\ 100 \end{pmatrix} - \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} 10 \\ 2 \end{pmatrix} \\ &\Rightarrow \quad 10x_2 \leq 760 \Rightarrow \text{Redundant} \\ \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} -10 & -10 \\ 1 & -2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} -210 \\ -50 \end{pmatrix} - \begin{pmatrix} 1 & 10 \end{pmatrix} \begin{pmatrix} 10 \\ 2 \end{pmatrix} \\ &\Rightarrow \quad -30x_2 \leq -740 \Rightarrow \text{Not redundant} \end{aligned}$$

(c) Computing new M and β

$$M^2 = \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & -1 \\ 0 & -10 \\ 0 & -30 \end{pmatrix} \qquad \beta^2 = \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \\ -210 \\ -740 \end{pmatrix}$$

4. Iteration l: note that when adding the new non-redundant constraint, the one added in the previous iteration becomes redundant, but the algorithm is not checking for that.

(a) Updated M and β from iteration $l-1$.

$$M^{l-1} = \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & -1 \\ 0 & -10 \\ 0 & -30 \\ \vdots & \vdots \\ 0 & m_l \end{pmatrix} \qquad \beta^{l-1} = \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \\ -210 \\ -740 \\ \vdots \\ \beta_l \end{pmatrix}.$$

(b) Computing $\hat{A}$, $\hat{b}$, $\hat{c}$

$$\hat{A} = \begin{pmatrix} 1 & 2 \\ -3 & -4 \\ -1 & 0 \\ 1 & -2 \\ -10 & -10 \\ -30 & -30 \\ \vdots & \vdots \\ m_l & m_l \end{pmatrix} \qquad \hat{b} = \begin{pmatrix} 1 \\ -3 \\ -1 \\ 1 \\ -10 \\ -30 \\ \vdots \\ m_l \end{pmatrix} \qquad \hat{c} = \begin{pmatrix} 2 \\ -4 \\ 0 \\ -2 \\ -210 \\ -740 \\ \vdots \\ m_l \end{pmatrix}$$

(c) Computing possible new constraints. Here $P^l = \{1, 4\}$, $Q^l = \{l + 3\}$ and $R^l = \emptyset$

$$\begin{aligned}
\begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} m_l & m_l \\ 1 & 2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} \beta_l \\ 100 \end{pmatrix} - \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} -m_l \\ 2 \end{pmatrix} \\
&\Rightarrow \quad -m_l x_2 \leq \beta_l - 97 m_l \\
\begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} m_l & m_l \\ 1 & -2 \end{pmatrix} x &\leq \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} \beta_l \\ -50 \end{pmatrix} - \begin{pmatrix} 1 & -m_l \end{pmatrix} \begin{pmatrix} -m_l \\ 2 \end{pmatrix} \\
&\Rightarrow \quad 3 m_l x_2 \leq \beta_l + 53 m_l
\end{aligned}$$

In order to check for both redundancy and convergence of the constraint on x_2, we consider the second constraint first. We have $m_{l+1} = 3m_l, m_1 = -10$, and $\beta_{l+1} = \beta_l + 53m_l, \beta_1 = -210$. Thus $m_l = m_1 3^{l-1}$, and $\beta_l = \beta_1 + 53m_1(3^{l-1} - 1)/2$. Thus the second constraint becomes $x_2 \geq 26.5 - 5.5/3^l$. This means that if the first constraint is redundant $\forall l$, then the second constraint is always not redundant and converges after an infinite number of steps to $x_2 \geq 26.5$. Thus the only thing we need to check is the redundancy of the first constraint, which is $x_2 \leq 70.5 + 5.53^{l-1}$. Since the top vertex of W has $x_2 = 62.5$, the first constraint is redundant $\forall l$, and converges to $x_2 \leq 70.5$.

(d) Therefore after an infinite number of iterations, $\hat{W}$ and $\hat{g}(x)$ converge to

$$\hat{W} = \left\{ x \,|\, \begin{pmatrix} 1 & 1 \\ -1 & -3 \\ 1 & -1 \\ -3 & -1 \\ 0 & -2 \end{pmatrix} x \leq \begin{pmatrix} 100 \\ -50 \\ 100 \\ -50 \\ -53 \end{pmatrix} \right\}$$

$$\hat{g}(x) = \begin{cases} \{u \in \mathbf{U} \,|\, u \geq \max(18 - x_1 - 4x_2/3, -100 - x_1, -55/2 - x_1 - x_2) \\ \qquad u \leq \min(98 - x - 1 - 2x_2, -52 - x_1 + 2x_2)\} & \text{if } x \in \hat{W} \\ \mathbf{U} & \text{otherwise} \end{cases}$$

Figure 2 shows the plot obtained by MATLAB for the first three iterations.

5.3 Example 3

Finally, we consider a multi-input multi-disturbance example. The system is defined by:

$$A = \begin{pmatrix} -1 & -8 & -1 \\ 1 & -4 & -1 \\ -5 & -3 & -1 \end{pmatrix}, B = \begin{pmatrix} 2 & 1 \\ 4 & 1 \\ 1 & -1 \end{pmatrix}, C = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 1 & 7 \\ 1 & 2 & 1 \end{pmatrix}, M = \begin{pmatrix} 3 & 1 & 0 \\ -1 & 3 & 0 \\ 1 & -1 & 0 \\ -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & -1 \end{pmatrix}, \beta = \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix} \tag{8}$$

The bounds on the control, $\mathbf{U}$, are defined by $\mathbf{U} = \{u \in \mathbb{R}^2 \mid Eu \leq \eta\}$ where E and η are

$$E = \begin{pmatrix} 1 & 1 \\ 1 & -1 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} \quad \eta = \begin{pmatrix} 1000 \\ 1000 \\ 1000 \\ 1000 \end{pmatrix} \tag{9}$$

The bounds on the disturbance, $\mathbf{D}$, are defined by $\mathbf{D} = \{d \in \mathbb{R}^3 \mid Gd \leq \gamma\}$ where G and γ are

$$G = \begin{pmatrix} 1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{pmatrix} \quad \gamma = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \tag{10}$$

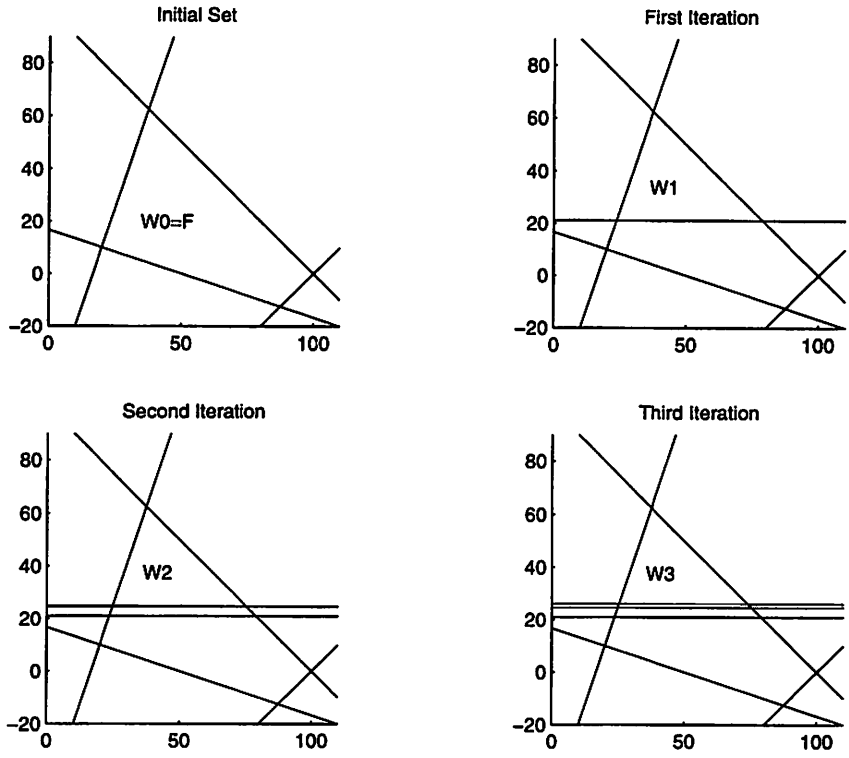

Figure 2: Iterations of the algorithm for Example 2

Using Matlab, this example converges in two iterations. Each iteration has several stages. Information about each stage is shown in Table 1. First, $\delta(\hat{C})$ is determined by solving several LP problems. Next, the constraints on x and u are grouped together for quantifier elimination. After quantifier elimination is performed, several constraints on x remain that must be checked for redundancy. After eliminating the redundant constraints, the remaining constraints are added to the previous set of constraints and the process is repeated. After two iterations, the constraints on x converge, and we find the maximal controlled invariant set and the least restrictive controller.

Table 1: Results of Example 3

Iteration	1	2
Number of LP problems for QE on d	6	10
Number of constraints on (x,u) before QE on u	10	14
Number of new constraints on x after QE on u	281	614
Number of new non-redundant constraints on x	4	0
Total number of constraints on x after iteration	10	10

$\hat{W}$ and $\hat{g}(x)$ are found to converge to

$$\hat{W} = \left\{ x \mid \begin{pmatrix} 3 & 1 & 0 \\ -1 & 3 & 0 \\ 1 & -1 & 0 \\ -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & -1 \\ -720 & -1360 & -160 \\ 360 & 680 & 80 \\ 720 & 1360 & 160 \\ 720 & 1360 & 160 \end{pmatrix} x \leq \begin{pmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 100 \\ 25720 \\ 29220 \\ 26340 \\ 23760 \end{pmatrix} \right\}$$

$$\hat{g}(x) = \begin{cases} \left\{ u \mid \begin{pmatrix} 10 & 4 \\ 10 & 2 \\ -2 & 0 \\ -6 & -2 \\ -1 & -2 \\ -1 & 1 \\ -7040 & 1920 \\ 3520 & 960 \\ 7040 & 1920 \\ 7040 & 1920 \\ 1 & 1 \\ 1- & -1 \\ 0 & 1 \\ 0 & -1 \end{pmatrix} u \leq \begin{pmatrix} 75 \\ 75 \\ 93 \\ 84 \\ 98 \\ 96 \\ 7160 \\ 20540 \\ 8980 \\ 6400 \\ 1000 \\ 1000 \\ 1000 \\ 1000 \end{pmatrix} + \begin{pmatrix} 2 & 28 & 4 \\ -4 & 4 & 2 \\ 2 & 4 & 0 \\ 0 & -12 & -2 \\ 4 & -5 & 0 \\ -5 & -3 & -1 \\ -160 & -11680 & -2240 \\ 80 & 5840 & 1120 \\ 160 & 11680 & 2240 \\ 160 & 11680 & 2240 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x \right\} & \text{if } x \in \hat{W} \\ \mathbf{U} & \text{otherwise} \end{cases}$$

6 Conclusions and Future Work

We showed that the problem of computing the maximal controlled invariant set and the least restrictive controller for discrete time systems is well posed and proposed a general algorithm for carrying out the computation. We then specialized the algorithm to discrete time linear systems with convex polygonal constraints, and showed how it can be implemented using linear programming. The decidability of the problem was also analyzed, and some simple, but interesting cases were found to be decidable.

We are currently working on sufficient conditions under which the problem is decidable. So far, it seems that the decidability property is not only dependent on the system itself, but also on the initial set, as shown by Example 2. Another topic of further research, is the application of these algorithm to hybrid discrete time systems, where some states and inputs take values in finite sets, while others in subsets of Euclidean space. It is easy to show how this class of systems is a special case of the more general class of DTS. Therefore, all the conclusions of Sections 2 and 3 directly extend to them. Unfortunately the implementation of the controlled invariance algorithm is more complicated, even in the case where the continuous state evolves according to a linear difference equation.

References

[1] D.S. Arnon, G.E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: the basic algorithm. *SIAM Journal on Computing*, 13(4):865–877, 1984.

[2] T. Başar and G. J. Olsder. *Dynamic Non-cooperative Game Theory.* Academic Press, second edition, 1995.

[3] A. Bensoussan and J.L. Menaldi. Hybrid control and dynamic programming. *Dynamics of Continuous, Discrete and Impulsive Systems*, (3):395–442, 1997.

[4] Michael S. Branicky, Vivek S. Borkar, and Sanjoy K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[5] P.E. Caines and Y.J. Wei. Hierarchical hybrid control systems: A lattice theoretic formulation. *IEEE Transactions on Automatic Control*, 43(4):501–508, April 1998.

[6] R.N. Cernikov. The solution to linear programming problems by elimination of unknowns. *Soviet Mathematics Doklady*, 2:1099–1103, 1961.

[7] Vijay Chandru. Variable elimination in linear constraints. *The Computer Journal*, 36(5):463–472, 1993.

[8] C.C. Chang and H.J. Keisler. *Model Theory.* North-Holland, third edition, 1990.

[9] D. Van Dalen. *Logic and Structure.* Springer-Verlag, 3rd edition, 1994.

[10] T. Dang and O. Maler. Reachability analysis via face lifting. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control*, number 1386 in LNCS, pages 96–109. Springer Verlag, 1998.

[11] A. Dolzman and T. Strum. REDLOG: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, 1997.

[12] G. Grammel. Maximum principle for a hybrid system via singular pertrubations. *SIAM Journal of Control and Optimization*, 37(4):1162–1175, 1999.

[13] M.R. Greenstreet and I. Mitchell. Integrating projections. In S. Sastry and T.A. Henzinger, editors, *Hybrid Systems: Computation and Control*, number 1386 in LNCS, pages 159–174. Springer Verlag, 1998.

[14] Michael Heymann, Feng Lin, and George Meyer. Control synthesis for a class of hybrid systems subject to configuration-based safety constraints. In *Hybrid and Real Time Systems*, number 1201 in LNCS, pages 376–391. Springer Verlag, 1997.

[15] Joseph Lewin. *Differential Games.* Springer-Verlag, 1994.

[16] John Lygeros, Claire Tomlin, and Shankar Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, pages 349–370, March 1999.

[17] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Theoretical Aspects of Computer Science*, number 900 in LNCS, pages 229–242. Springer Verlag, 1995.

[18] A. Nerode and W. Kohn. Multiple agent hybrid control architecture. In Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors, *Hybrid Systems*, number 736 in LNCS, pages 297–316. Springer Verlag, New York, 1993.

[19] George J. Pappas. *Hybrid Systems: Computation and Abstraction.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, 1998.

[20] George J. Pappas, Gerardo Lafferriere, and Shankar Sastry. Hierarchically consistent control systems. In *IEEE Conference on Decision and Control*, pages 4336–4341, Tampa, FL, December 1998.

[21] Benedetto Piccoli. Necessary conditions for hybrid optimization. In *IEEE Conference on Decision and Control*, pages 410–415, Phoenix, Arizona, U.S.A., December 7-10 1999.

[22] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, Vol.77(1):81–98, 1989.

[23] H. L. Royden. *Real Analysis.* MacMillan, third edition, 1988.

[24] A. Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, 60:387–374, 1954.

[25] Hector J. Sussmann. A maximum principle for hybrid optimal control problems. In *IEEE Conference on Decision and Control*, pages 425–430, Phoenix, Arizona, U.S.A., December 7-10 1999.

[26] A. Tarski. *A decision method for elementary algebra and geometry.* University of California Press, second edition, 1951.

[27] W. Thomas. On the synthesis of strategies in infinite games. In Ernst W. Mayr and Claude Puech, editors, *Proceedings of STACS 95, Volume 900 of LNCS*, pages 1–13. Springer Verlag, Munich, 1995.

[28] Claire Tomlin, John Lygeros, and Shankar Sastry. Computing controllers for nonlinear hybrid systems. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, number 1569 in LNCS, pages 238–255. Springer Verlag, 1999.

[29] Howard Wong-Toi. The synthesis of controllers for linear hybrid automata. In *IEEE Conference on Decision and Control*, pages 4607–4613, San Diego, California, USA, December 10-12 1997.