A simple hierarchy for computing controlled invariant sets[*]

Tzanis Anevlavis[†]
University of California, Los Angeles
Los Angeles, CA, USA
janis10@ucla.edu

Paulo Tabuada
University of California, Los Angeles
Los Angeles, CA, USA
tabuada@ucla.edu

ABSTRACT

In this paper we revisit the problem of computing controlled invariant sets for controllable discrete-time linear systems and present a novel hierarchy for their computation. The key insight is to lift the problem to a higher dimensional space where the maximal controlled invariant set can be computed exactly and in closed-form for the lifted system. By projecting this set into the original space we obtain a controlled invariant set that is a subset of the maximal controlled invariant set for the original system. Building upon this insight we describe in this paper a hierarchy of spaces where the original problem can be lifted into so as to obtain a sequence of increasing controlled invariant sets. The algorithm that results from the proposed hierarchy does not rely on iterative computations. We illustrate the performance of the proposed method on a variety of scenarios exemplifying its appeal.

CCS CONCEPTS

• **Theory of computation** → **Logic and verification**; **Verification by model checking**; **Modal and temporal logics**.

KEYWORDS

Controller Synthesis; Safety; Controlled Invariance.

ACM Reference Format:
Tzanis Anevlavis and Paulo Tabuada. 2020. A simple hierarchy for computing controlled invariant sets. In *23rd ACM International Conference on Hybrid Systems: Computation and Control (HSCC '20), April 22–24, 2020, Sydney, NSW, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3365365.3382205

1 INTRODUCTION

Controlled invariance is a prevalent concept throughout the history of control systems. Principally, a set is controlled invariant if trajectories that start in it, can be forced to remain therein by using admissible control inputs. Controlled invariance is the technical problem to be solved when synthesizing a controller enforcing safety properties, in the sense of ensuring that something bad never happens. If a trajectory is to remain forever in a set of safe states, then it must start in a Controlled Invariant Set (CIS) within the set of safe states. Therefore, such sets are an important tool in several control design problems, e.g., they act as safe sets in constrained control [8, 13, 18], are used to design stabilizing control policies [6, 18], guarantee feasibility of optimization problems in model predictive control [21], and more recently, they assumed a central role in synthesizing controllers enforcing safety properties expressed in temporal logic [30, 31, 35].

Consequently, a substantial effort has been devoted over the past decades to computing CISs for continuous-time, discrete-time and hybrid systems. Beginning with the pioneering work of [4] on the computation of the *Maximal* Controlled Invariant Set (MCIS), many other contributions followed and are documented in [5, 7].

However, computing CISs is still extremely challenging. Although an iterative procedure was proposed in [4, 12] (see Section 2 for details), its termination is not guaranteed for a number of important classes of problems. Moreover, the sets obtained become more complex with each iteration, thus making this method intractable for high dimensional systems.

To alleviate these impediments, alternative approaches, relying on optimization and approximation techniques, have been investigated. Unavoidably, these techniques suffer from the efficiency-accuracy tradeoff, and the curse of dimensionality. Some of the state-of-the-art methods propose inner and outer approximations of the MCIS, by solving Semi-Definite Programs (SDPs) [20, 28], or Linear Programs (LPs) [33]. Naturally, outer approximations are not invariant, but even inner approximations are not guaranteed to be. Other attempts utilize sampling-based approximations [14], or solve SDPs [17] to keep the system within a set over some specified time horizon, thus only providing invariance on finite time intervals. Another line of work is based on the iterative procedure of [4, 12], such as [11, 29, 32] for a class of control systems with bounded disturbances, and [10, 23, 34] for a class of switched systems.

In some cases [7, 19], it is possible to guarantee termination of the classical algorithm. An important case ensuring this, is the class of controllable discrete-time linear systems, with the states and controls lying in finite unions of hyper-rectangles, see [31, 35, 36], and in the case of bounded perturbations, with states and controls in polytopes [32].

In the same spirit, two methods [1, 22] compute *exact* CISs and also guarantee finite termination. In [22], an efficient method was presented using SDPs that computes an ellipsoidal CIS for a class of hybrid systems. In turn, [1] proposed a novel method that computes CISs by lifting the problem to a higher dimensional space, in which invariant hyper-rectangles are computed in closed-form. Thus, the MCIS for the lifted system is the union of such hyper-rectangles. By

[*]This work was supported in part by the NSF grant 1645824 and the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.
[†]Corresponding author.

HSCC '20, April 22–24, 2020, Sydney, NSW, Australia
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7018-9/20/04...$15.00
https://doi.org/10.1145/3365365.3382205

projecting this MCIS back to the original space, a CIS is obtained for the original system.

In this paper, we extend the work in [1] by drawing inspiration from the structure of CISs in automata. It is well-known that given an automaton, a subset of its state-space is invariant if and only if it contains a loop [25, 26]. This observation lets us interpret the algorithm in [1] as the construction of loops of length one, i.e., self-loops. Motivated by this reasoning, we consider in this paper the more general problem of constructing loops of length greater than one.

More specifically, we propose a *hierarchy* based on an upper bound for the length of loops. As a consequence, by enlarging the upper bound we compute potentially larger controlled invariant sets. Since the union of CISs is a CIS, it follows that we can potentially enlarge the CIS based on a loop of length L, by taking its union with a CIS based on a loop of length $L + 1$. This is illustrated in Section 2.

Similarly to [1], the algorithm proposed in this paper computes the MCIS in a higher dimensional space in closed-form, hence, it does not rely on iterative computations. The computed MCIS is projected back to the original space, obtaining thus a CIS for the original problem. Although the proposed algorithm does not compute the MCIS, it is nonetheless complete in the sense that if the MCIS is non-empty, it computes a non-empty CIS.

The paper is organized as follows, in Section 2 we present a motivating example illustrating the ability of the proposed method to compute incrementally larger CISs. In Section 3, the problem is mathematically setup, along with the essential definitions. Next, Section 4 recalls the idea of computing a CIS by lifting the problem into higher dimensional spaces, and provides the main technical results. Finally, in Section 5 we provide a number of examples evaluating the trade-off between size of the computed sets and performance, prior to concluding our remarks in Section 6.

2 MOTIVATING EXAMPLE

Before getting into the details of our work, we present a pedagogical example that motivates the usefulness, and illustrates the improved performance of the proposed algorithm over the one in [1].

Example 1. *For our motivating example we consider the following system in $\mathbb{R}^2$ taken from [1]:*

$$x^+ = \begin{bmatrix} 1.5 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} u \quad .$$

Let D be a subset of $\mathbb{R}^2$ (union of all sets in Fig. 1a), and $u \in [-1, 1]$. We aim to compute a subset of D in which the state $x \in \mathbb{R}^2$ can be forced to remain with a suitable choice of u, i.e., a CIS of D.

In Figure 1a, the union of all sets is D, and the union of all sets except the blue one is the MCIS of D. As explained in the introduction, we lift the problem into a higher dimensional space where the safe set D is represented as a union of hyper-rectangles. Recall that the algorithm in [1] computes invariant hyper-rectangles. Projecting these hyper-rectangles into the original space, the CIS obtained for our example is the white set in Figures 1a-1d. Now if instead we allow the state to move between these hyper-rectangles in loops, i.e., always returning back to a hyper-rectangle in the loop, we may obtain larger CISs. These CISs are depicted in Figures 1a to 1d as

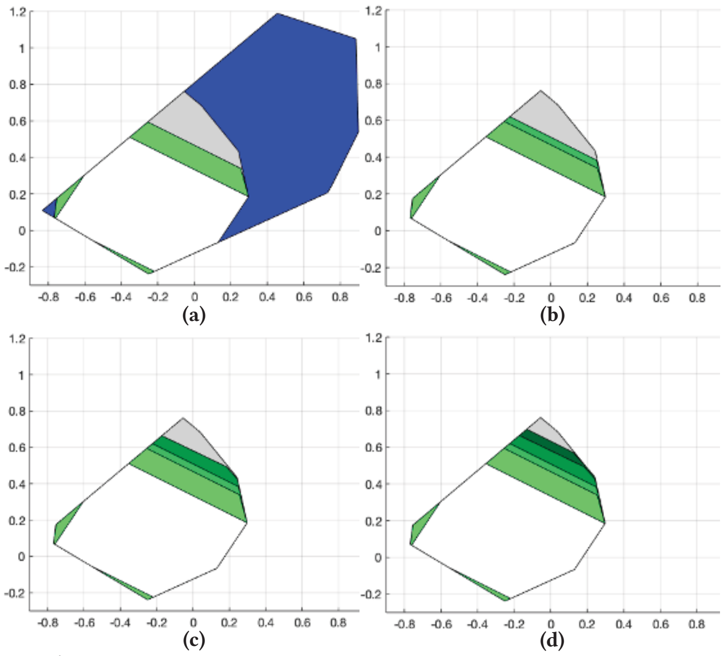


Figure 1:
(1a-d): Sets of Example 1.
Legend: Original set D (blue); Exact MCIS of D (light gray); Controlled invariant set from [1] (white); Controlled invariant sets from Algorithm 1 in different shades of green for loops of length 2 to 5.

the incremental union of the white set with the stripes in different shades of green for loops of length 2, 3, 4, and 5 respectively. For easier comparison, only the relevant CISs appear in Figures 1b-1d.

3 PROBLEM FORMULATION

In this section, the problem of computing a controlled invariant subset $\mathcal{D}$ of a compact polyhedron $D \subset \mathbb{R}^n$ for a discrete-time linear system Σ is formalized. We begin with the necessary definitions.

Definition 1 (Polyhedron and polytope). *A* polyhedron *D in $\mathbb{R}^n$ is a set of the form:*

$$D = \left\{ x \in \mathbb{R}^n \;\middle|\; Gx \leq f \right\}, \tag{1}$$

where $G \in \mathbb{R}^{k \times n}$, and $f \in \mathbb{R}^k$. Set D can be seen as an intersection of finitely many closed halfspaces of the form $\{x \in \mathbb{R}^n | g_j^T x \leq f_j\}$, where g_j^T is the j-th row of G and f_j the j-th element of f. A polyhedron that is both closed and bounded is called a polytope.

Definition 2 (Discrete-time linear system). *A* discrete-time linear system *Σ is a linear difference equation of the form:*

$$x^+ = Ax + Bu \quad , \tag{2}$$

where $x \in \mathbb{R}^n$, $u \in \mathbb{R}$, $A \in \mathbb{R}^{n \times n}$, and $B \in \mathbb{R}^n$. The state of Σ is denoted by x, and the input *of Σ by u. A linear system Σ is defined by the pair of matrices (A, B).*

Assumption 1. *In this work, we make the following* assumptions *as part of the problem setup:*

(1) The set $D \subset \mathbb{R}^n$ is a compact polyhedron, i.e., a polytope[1].
(2) The system Σ is controllable[2].

Definition 3 (Controlled invariant subset). *Consider a polytope D as in (1), and a discrete-time linear system Σ as in (2). A set $\mathcal{D} \subseteq D$ is a* Controlled Invariant Subset *(CIS) of D for Σ if:*

$$x \in \mathcal{D} \Rightarrow \exists u \in \mathbb{R} : Ax + Bu \in \mathcal{D}.$$

We call D the safe set.

Notice, that it follows from the implication in Definition 3 that for any initial state in $\mathcal{D}$ there is a control policy that forces the resulting trajectories to remain in $\mathcal{D}$ forever [4]. Now we are ready to formalize the problem we address.

Problem 1. *Given a safe set D as in (1), and a discrete-time linear system Σ as in (2), compute a controlled invariant subset of D for Σ.*

There exists a classical algorithm that solves Problem 1. In order to recall it, we need to first define the *controlled predecessor* of a set $X \subset \mathbb{R}^n$ for the system Σ, denoted by $\mathrm{Pre}_\Sigma(X)$:

$$\mathrm{Pre}_\Sigma(X) = \left\{x \in \mathbb{R}^n \middle| \exists u \in \mathbb{R} : x^+ = Ax + Bu \in X\right\}. \tag{3}$$

Intuitively, $\mathrm{Pre}_\Sigma(X)$ is the set of all states $x \in \mathbb{R}^n$ for which there exists a control input u forcing them into X in one time-step. Considering now our safe set D, it follows that $\mathrm{Pre}_\Sigma(D)$ is given by:

$$\mathrm{Pre}_\Sigma(D) = \pi_{\mathbb{R}^n}\left(W(D)\right), \tag{4}$$

where the auxiliary set $W(D)$ is:

$$W(D) = \left\{(x,u) \in \mathbb{R}^n \times \mathbb{R} \middle| \begin{bmatrix} GA & GB \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \le f \right\}, \tag{5}$$

i.e., the set of pairs $(x,u) \in \mathbb{R}^n \times \mathbb{R}$ such that $Ax + Bu \in D$, and $\pi_{\mathbb{R}^n}$ is the natural projection from $\mathbb{R}^n \times \mathbb{R}$ to $\mathbb{R}^n$.

The classical algorithm that solves Problem 1 consists of the following iterative procedure proposed in [4, 12]:

$$\begin{cases} S_0 = D \\ S_{i+1} = S_i \cap \mathrm{Pre}_\Sigma(S_i) \end{cases}, \tag{6}$$

with the terminating condition $S_{i+1} = S_i$. Procedure (6) computes the MCIS of D for Σ, and theoretically works for any discrete-time system and set, not just for linear systems and polyhedra. Even though the proposed algorithm in this paper does not rely on iterative computations, we will use the above procedure to establish some of its properties.

In order to provide a clean and streamlined mathematical description of the proposed results, we work with the Brunovsky normal form [9] of (2):

$$x^+ = \begin{bmatrix} x_1^+ \\ \vdots \\ x_{n-1}^+ \\ x_n^+ \end{bmatrix} = \begin{bmatrix} x_2 \\ \vdots \\ x_n \\ u \end{bmatrix} = A_c x + B_c u. \tag{7}$$

Any controllable linear system (2) can be transformed in the above form by a suitable change of coordinates and feedback [2]. Hence, without loss of generality, we deal with systems of the form (7).

[1]See proof of Section 4.5.
[2]This assumption is only used to obtain the Brunovsky normal form in (7). For details refer to [2, Ch. 3]

Remark 1. *In the remainder of the paper, we assume unconstrained inputs, i.e., $u \in \mathbb{R}$. Our results hold in the exact same manner in the presence of input constraints, i.e. $u \in \mathcal{U} \subset \mathbb{R}$. In this case, we extend the original state space by one dimension, thus obtaining the state $y = (x,u)$, and introduce a new unconstrained input $v \in \mathbb{R}$ governing the evolution of the state u according to $u^+ = v$. The input constraints are now part of the safe set for the extended system:*

$$y^+ = \begin{bmatrix} x^+ \\ u^+ \end{bmatrix} = \begin{bmatrix} x_1^+ \\ \vdots \\ x_n^+ \\ u^+ \end{bmatrix} = \begin{bmatrix} x_2 \\ \vdots \\ u \\ v \end{bmatrix}, \tag{8}$$

and the safe set for the extended system is $D \times \mathcal{U}$, where D is the safe set for the original system.

Finally, we introduce the notion of transition system that we will use to describe loops of various lengths.

Definition 4 (Finite transition system). *A* Finite Transition System *(FTS) $\mathcal{A}$ is a tuple (Q,δ), where Q is a finite set of states, and $\delta : Q \to Q$ is a state-transition function.*

In this work we consider only deterministic FTSs, therefore each state q_i has a unique successor state q_j, and we write $\delta(q_i) = q_j$.

4 CONTROLLED INVARIANCE IN HIGHER DIMENSIONAL SPACES

4.1 Controlled invariance in two moves

A novel algorithm for computing a CIS was recently presented in [1], and relies on an idea that works in two moves: 1) it lifts the safe set to a higher dimensional space, where it can be represented as a union of hyper-rectangles, and the MCIS is computed in closed-form; 2) it projects this MCIS back to the original domain, obtaining thus a CIS for the original problem.

We briefly review the lifting procedure of [1]. In order to represent the safe set $D = \{x \in \mathbb{R}^n \mid Gx \le f\}$ as a union of hyper-rectangles, a new variable $\lambda \in \mathbb{R}^{kn}$ is introduced. Then, $D \subset \mathbb{R}^n$ is lifted to the set $D^\ell \subset \mathbb{R}^{n+kn}$ by the following construction:

$$\begin{cases} g_{ji} x_i \le \lambda_i^j, i = 1, \cdots, n \\ \sum_{i=1}^n \lambda_i^j \le f_j \end{cases}, j = 1, \ldots, k, \tag{9}$$

where g_{ji} is the entry of G in its j-th row and i-th column, and $\lambda = \left(\lambda^1, \ldots, \lambda^k\right) \in \mathbb{R}^{kn}$, with each $\lambda^j \in \mathbb{R}^n$, $j = 1, \ldots, k$. The *lifted* set D^ℓ is:

$$D^\ell = \left\{(x,\lambda) \in \mathbb{R}^{n+kn} \;\middle|\; \bar{G}x \le \lambda,\ H\lambda \le f\right\}, \tag{10}$$

where $\bar{G} = [diag(g_1) \cdots diag(g_k)]^{\mathrm{T}}$, with $diag(g_j)$ a diagonal matrix with the elements of g_j along its diagonal, with g_j being the j-th row of G, and $H \in \mathbb{R}^{k \times kn}$ is such that $H\lambda \le f \Leftrightarrow \sum_i \lambda_i^j \le f_j$, $j = 1, \ldots, k$. Once λ is fixed, (9) defines a hyper-rectangle by restricting $x_i \in \mathbb{R}$ to an interval. Hence, one can see D^ℓ as the union of all the hyper-rectangles defined by the λ-variables satisfying the second equation in (9). In addition, the system Σ (7) is lifted to Σ^ℓ:

$$\begin{bmatrix} x^+ \\ \lambda^+ \end{bmatrix} = \begin{bmatrix} A_c & 0 \\ 0 & \mathbb{I} \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} + \begin{bmatrix} B_c \\ 0 \end{bmatrix} u. \tag{11}$$

Figure 2 illustrates the aforementioned ideas: the original set, an octagon, is lifted to a higher dimensional space and represented by a union of different hyper-rectangles parameterized by λ. For a fixed λ we have an instantiation of the hyper-rectangles (transparent). By projecting all these instantiations back to the original domain, one recovers the octagon.

Notice that in (11) equation $\lambda^+ = \lambda$ forces λ to remain constant so that the hyper-rectangles are preserved by the lifted system. This requirement allows for the computation of the MCIS of D^ℓ in the lifted space as a union of invariant hyper-rectangles. This MCIS is given in closed-form as:

$$S_{\text{MCIS}} = \left\{ \begin{array}{l} x \in \mathbb{R}^n, \\ \lambda \in \mathbb{R}^{kn} \end{array} \middle| \begin{array}{c} G_{\text{MCIS}} x \leq \lambda \\ H\lambda \leq f \\ \Gamma_n \lambda \leq 0 \end{array} \right\}, \tag{12}$$

where the exact expressions for the matrices G_{MCIS}, H, and Γ_n can be found in [1, Th. 3.1].

4.2 Proposed algorithm

In this paper, we relax the requirement of invariant hyper-rectangles imposed in [1]. Intuitively, we allow the state of the lifted system to move between hyper-rectangles, visiting each of them at least every L steps. This is formalized by an FTS $\mathcal{A}$ with L states. Each state of $\mathcal{A}$ corresponds to a collection of hyper-rectangles, and different states of $\mathcal{A}$ can be associated with the same collection. The FTSs we consider consist of a *single* loop of length L and have L states:

$$\mathcal{A} = (Q, \delta) \text{ such that } |Q| = L,\ \delta^L(q) = q,\ \forall q \in Q, \tag{13}$$

where δ^i denotes the i-fold composition of δ with itself. The proposed method is realized by the following steps:

(1) Given the system Σ, and the safe set D, select an FTS $\mathcal{A}$ satisfying (13). Composing Σ and $\mathcal{A}$ we obtain $\Sigma' = \Sigma \times \mathcal{A}$:

$$\begin{bmatrix} x^+ \\ q^+ \end{bmatrix} = \begin{bmatrix} A_c x + B_c u \\ \delta(q) \end{bmatrix}, \tag{14}$$

with state $(x, q) \in \mathbb{R}^n \times Q$. Similarly, we compute the new set D' as the product $D' = D \times Q$. We interpret the set D' as L copies of D, one for each possible value of q. Let us denote each such copy by $D_q = D \times \{q\}$, $q \in Q$.

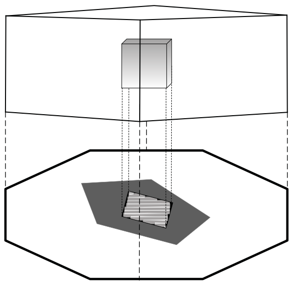


Figure 2: **Illustration of domain lifting. Octagon is the safe set D. Transparent hyper-rectangle is an instantiation of the hyper-rectangles whose union equals the lifted set D^ℓ. Gradient cube is the MCIS of D^ℓ. Light grey rectangle is the CIS obtained after projection. Dark grey polygon is the MCIS.**

(2) Lift system Σ' from $\mathbb{R}^n \times Q$ to $\mathbb{R}^n \times Q \times \left(\mathbb{R}^{kn}\right)^Q$. The lifted system Σ'^ℓ is:

$$\begin{bmatrix} x^+ \\ q^+ \\ \lambda^+(q) \end{bmatrix} = \begin{bmatrix} A_c x + B_c u \\ \delta(q) \\ \lambda \circ \delta(q) \end{bmatrix}. \tag{15}$$

The lift of D' is performed using construction (9), and D' is lifted to D'^ℓ:

$$D'^\ell = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \bar{G}x \leq \lambda(q), \\ H\lambda(q) \leq f \end{array} \right\}. \tag{16}$$

For the rest of the paper, towards ease of notation, we write λ_q to denote $\lambda(q)$, and also denote $\lambda^+(q) = \lambda \circ \delta(q)$ by λ_{q^+}. This completes the lift process.

It is important to notice that λ lives in the space of *functions* from Q to $\mathbb{R}^{kn}$. For a fixed q, we have a collection of hyper-rectangles corresponding to copy D_q, and for a fixed value of λ_q, an instantiation of these hyper-rectangles. To portray it, the case of the state moving between two hyper-rectangles is shown in Figure 3.

Given system Σ'^ℓ and set D'^ℓ, in principle one could use iterative procedure (6) to obtain the MCIS. Albeit termination of (6) is not guaranteed in general, we prove that for system (15) and set (16) it terminates in a finite number of steps, and provide the closed-form expression for the resulting MCIS. Therefore, the proposed algorithm computes the MCIS in the lifted space $\mathbb{R}^n \times Q \times \left(\mathbb{R}^{kn}\right)^Q$ without any iterative computations. Finally, by projecting the computed MCIS back to $\mathbb{R}^n$ a CIS is obtained for the original system. The ideas of this subsection are summarized in Algorithm 1, where $MCIS_{S_0, \Sigma'^\ell}$ denotes the MCIS of a set S_0 for a system Σ'^ℓ.

4.3 Exact computation of CIS in the lifted space

This section contains the main theorem of this work. It is a generalization of the result in [1], and was inspired by similar results in [31, 35, 36], in which finite termination is guaranteed. The latter

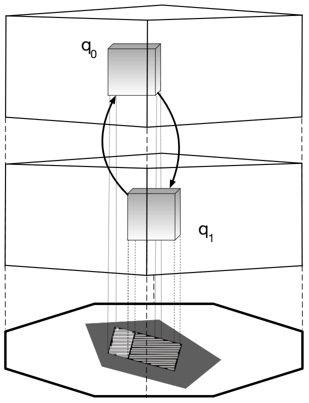


Figure 3: **The state in the lifted space moves between two instantiations of (transparent) hyper-rectangles corresponding to states q_0, q_1 of $\mathcal{A}$. For each of them, we compute a (gradient) cube. Their union is the MCIS of the lifted set. By projecting it, a larger CIS is obtained in the original space.**

```
Algorithm 1: Computation of a controlled invariant subset of a set $D$.
  Data: A set $D = \{x \in \mathbb{R}^n | Gx \le f\}$ as in (1), a controllable
        system $\Sigma$ defined by the pair $(A_c, B_c)$ as in (7), and an
        FTS $\mathcal{A} = (Q, \delta)$ as in (13).
  Result: A controlled invariant subset of $D$.
  Input: $G, f, A_c, B_c, Q, \delta$
  Define:
    $\Sigma'^{\ell} : \begin{bmatrix} x^+ \\ q^+ \\ \lambda^+(q) \end{bmatrix} = \begin{bmatrix} A_c x + B_c u \\ \delta(q) \\ \lambda \circ \delta(q) \end{bmatrix}$
    $D'^{\ell} = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \bar{G}x \le \lambda(q), \\ H\lambda(q) \le f \end{array} \right\}.$
  Compute:
    $S_0 \leftarrow D'^{\ell}$
    $MCIS_{S_0, \Sigma'^{\ell}} \leftarrow S_{\mathcal{A}}$ as in (18)
    $\mathcal{D}_{\mathcal{A}} \leftarrow \pi_{\mathbb{R}^n}(S_{\mathcal{A}})$
  Return $\mathcal{D}_{\mathcal{A}}$
```

results rely on controllability as one of their assumptions, however in the lifted space where procedure (6) is applied, the system (15) is not controllable.

In this theorem we make three claims: 1) finite termination, 2) correctness, and 3) completeness of Algorithm 1. While 1) is proven in this subsection, we prove 2) and 3) in subsequent theorems.

Theorem 4.1. *Consider a safe set D as in (1), and a discrete-time linear system Σ as in (7), for which Assumption 1 holds, and select an FTS $\mathcal{A}$ as in (13). A CIS $\mathcal{D}_{\mathcal{A}} \subseteq D$ for Σ is provided by:*

$$\mathcal{D}_{\mathcal{A}} = \pi_{\mathbb{R}^n}\left(S_{\mathcal{A}}\right), \tag{17}$$

where $S_{\mathcal{A}}$ is the MCIS of D'^{ℓ} for system Σ'^{ℓ}, and whose closed-form expression is given in (18).

Proof. To show finite termination, we apply iterative procedure (6) with $S_0 = D'^{\ell}$, for system Σ'^{ℓ} (15). In each iteration $i \in \mathbb{N}$, we compute $S_{i+1} = S_i \cap \mathrm{Pre}_{\Sigma'^{\ell}}(S_i)$. Computation of $\mathrm{Pre}_{\Sigma'^{\ell}}(S_i)$ requires the auxiliary set $W(S_i)$ as in (5). During the first n iterations, the constraints that define $W(S_i)$ are of the form:

$$\begin{bmatrix} \bar{G}A_c^i & \bar{G}A_c^{i-1}B_c \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \le \lambda_{q^+}, \quad q \in Q.$$

We assumed (A_c, B_c) in Brunovsky normal form (7). It follows then, that in the above set of inequalities, variables x and u do not appear in the same inequalities, i.e., if a row of $\bar{G}A_c^i$ is zero, then the corresponding row of $\bar{G}A_c^{i-1}B_c$ is non-zero, and vice versa. From (4), to obtain $\mathrm{Pre}_{\Sigma'^{\ell}}(S_i)$ we eliminate u, i.e., we project the set $W(S_i)$ from $\mathbb{R}^n \times \mathbb{R} \times Q \times (\mathbb{R}^{kn})^Q$ to $\mathbb{R}^n \times Q \times (\mathbb{R}^{kn})^Q$. This yields constraints *only* on λ since for the rows of $\bar{G}A_c^{i-1}B_c$ that are non-zero, the corresponding rows of $\bar{G}A_c^i$ are zero. At the end of the n-th iteration, without loss of generality, describe the set S_n by:

$$S_n = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \widehat{G}_n x \le \lambda_q \\ \Delta_n \lambda_q \le 0 \end{array} \right\},$$

where $\Delta_n \lambda_q \le 0$ denotes all constraints involving only λ, and $\widehat{G}_n x \le \lambda$, the constraints involving both x and λ.

By noticing that for $i > n$, $GA_c^i = GA_c^{i-1}B_c = 0$ holds, it follows that no new constraints are generated by eliminating u as above beyond iteration n. Resuming the procedure nonetheless, observe that:

$$\mathrm{Pre}_{\Sigma'^{\ell}}(S_n) = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \widehat{G}_n x \le \lambda_{q^+} \\ \Delta_n \lambda_{q^+} \le 0 \end{array} \right\},$$

and therefore:

$$S_{n+1} = S_n \cap \mathrm{Pre}_{\Sigma'^{\ell}}(S_n) = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \widehat{G}_n x \le \lambda_q \\ \Delta_n \lambda_q \le 0 \\ \widehat{G}_n x \le \lambda_{q^+} \\ \Delta_n \lambda_{q^+} \le 0 \end{array} \right\},$$

that is, even though $\widehat{G}_n$ and Δ_n do not change for iterations $i \ge n$, termination is not reached yet since by computing $\mathrm{Pre}_{\Sigma'^{\ell}}(S_n)$ we get a new set of constraints, similar to the previous one, but on λ_{q^+}.

Since $q^+ = \delta(q)$, and $\mathcal{A}$ consists of a loop of length L, it follows from (13) that $\delta^L(q) = q$. In L more iterations, $\mathrm{Pre}_{\Sigma'^{\ell}}(S_{n+L-1})$ introduces $\widehat{G}_n x \le \lambda_{\delta^L(q)} = \lambda_q$, and $\Delta_n \lambda_{\delta^L(q)} = \Delta_n \lambda_q \le 0$, which already appear. Consequently, at this point, $S_{n+L} = S_{n+L-1} \cap \mathrm{Pre}_{\Sigma'^{\ell}}(S_{n+L-1}) = S_{n+L-1}$ holds, and the termination condition is met. Hereby, finite termination is proven in a total of $n + L - 1$ steps.

Let $S_{\mathcal{A}}$ be the result of eliminating q from S_{n+L-1}. The closed-form expression of $S_{\mathcal{A}}$ is:

$$S_{\mathcal{A}} = \left\{ (x, \lambda) \in \mathbb{R}^n \times \left(\mathbb{R}^{kn}\right)^Q \middle| \begin{array}{c} \widehat{G}_n x \le \widehat{R}_n \lambda \\ \widehat{H}\lambda \le \widehat{f} \\ \widehat{\Gamma}_n \lambda \le 0 \end{array} \right\}, \tag{18}$$

and its exact form is found in Section 7 with a detailed proof. □

Remark 2 (The multi-input case). *In this paper, we only discuss single-input systems to simplify the presentation. It is straightforward to extend our results to controllable systems with m inputs. In this case, the original system can be decomposed into m decoupled single-input subsystems. If the safe set can be re-written as a product of m sets, each involving only the corresponding states of the decoupled systems, then it suffices to apply our algorithm to each of the subsystems, and compute the product of the resulting CISs. In the general case, where the safe set cannot be decomposed, we can use arguments similar to those employed for the single-input case to prove finite termination in a total of $\mu + L - 1$ steps, where μ is the* maximal controllability index, *i.e., the size of the largest subsystem. For details see [2, Ch. 3].*

4.4 Proof of correctness

In this subsection we prove that the set computed by Algorithm 1 is indeed a controlled invariant set.

Theorem 4.2. *Consider a safe set D as in (1), and a discrete-time linear system Σ as in (7), for which Assumption 1 holds, and select an FTS $\mathcal{A}$ as in (13). For this setting, if Algorithm 1 returns a non-empty set $\mathcal{D}_{\mathcal{A}}$, then $\mathcal{D}_{\mathcal{A}}$ is a CIS of D for Σ.*

Proof. For any $x \in \mathcal{D}_{\mathcal{A}}$, there exists a $\hat{x} = (x, \lambda) \in S_{\mathcal{A}}$ such that $\pi_{\mathbb{R}^n}(\hat{x}) = x$. Since $S_{\mathcal{A}}$ is controlled invariant, it follows that $\hat{x}^+ = (x^+, \lambda^+) = (A_c x + B_c u, \lambda \circ \delta) \in S_{\mathcal{A}}$. Consequently, $\pi_{\mathbb{R}^n}(\hat{x}) \in \mathcal{D}_{\mathcal{A}}$. By noting that $A_c x + B_c u = \pi_{\mathbb{R}^n}((A_c x + B_c u, \lambda)) \in D$ the proof is now finished. □

4.5 Proof of completeness

In this subsection we establish a form of completeness. If the MCIS of a compact polyhedron D is non-empty, then Algorithm 1 always finds a non-empty CIS of D.

Theorem 4.3. *Consider a safe set D as in (1), and a discrete-time linear system Σ as in (7), for which Assumption 1 holds, and select an FTS $\mathcal{A}$ as in (13). For this setting, if the MCIS of D is non-empty, then Algorithm 1 returns a non-empty set $\mathcal{D}_{\mathcal{A}}$, which is a controlled invariant subset of D.*

Proof. Recall that in this paper, we require the state of the lifted system Σ'^{ℓ} in (15) to loop back to a hyper-rectangle in a finite number of steps, L. This is formalized by the FTS $\mathcal{A}$ in (13). The invariant hyper-rectangles proposed in [1] are of course a special case of returning every L steps. Let us denote by $\tilde{S}_{\text{MCIS}}$ the lifted version of S_{MCIS} in (12), proposed in [1], to the space where $S_{\mathcal{A}}$ lives. Then, $\tilde{S}_{\text{MCIS}}$ is a subset of $S_{\mathcal{A}}$ and consequently:

$$\tilde{S}_{\text{MCIS}} \subseteq S_{\mathcal{A}} \implies \pi_{\mathbb{R}^n}\left(\tilde{S}_{\text{MCIS}}\right) \subseteq \pi_{\mathbb{R}^n}(S_{\mathcal{A}}) \implies \mathcal{D} \subseteq \mathcal{D}_{\mathcal{A}},$$

where $\mathcal{D}$ is the CIS computed in [1], and $\mathcal{D}_{\mathcal{A}}$ the CIS computed by our algorithm. From [1, Th. 3.3], for which *compactness* of D is a requirement, if the MCIS of D is non-empty, then $\mathcal{D}$ is also non-empty. Therefore:

$$\text{MCIS of } D \neq \emptyset \implies \mathcal{D} \neq \emptyset \implies \mathcal{D}_{\mathcal{A}} \neq \emptyset,$$

for any FTS $\mathcal{A}$. This concludes the proof. □

4.6 Some observations

Upon concluding our technical results, and prior to the computational evaluation of our novel method, the following observations are in order:

(1) It suffices to consider *only* the simple FTS proposed in (13) of length L. If more complicated deterministic transitions of length L exist, e.g., a hyper-rectangle is visited more than once in the loop, Algorithm 1 will consider them *automatically*. To exhibit this, consider the two FTSs in Figure 4, both have loops of length 4. The first (Fig. 4a) moves the state between 3 collections of hyper-rectangles, labelled by the respective λ-variables. The other (Fig. 4b) considers as many collections as the length of the loop, which are visited consecutively. The CIS based on the FTS in Fig. 4a, is contained in the one based on the FTS in Fig. 4b.

(2) The proposed hierarchy is *not strict*. This can be easily verified with a toy example of a double integrator and a safe set of box constraints. In this case, our approach with $L = 1$ returns the MCIS, as any $L > 1$ does.

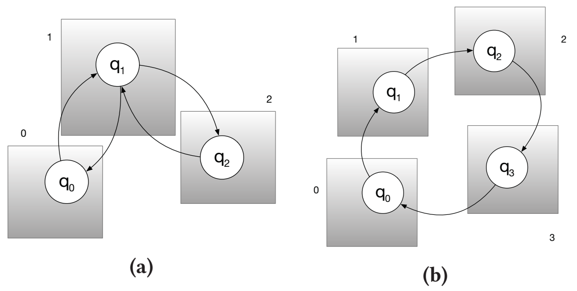


Figure 4: Two different FTSs with the same loop length: In (4a), the loop is $(q_0q_1q_2q_1)^{\omega}$. while in (4b), $(q_0q_1q_2q_3)^{\omega}$, where ω denotes infinite repetition.

(3) Denote by $\mathcal{D}_L$ and $\mathcal{D}_{L'}$ the CISs computed by Algorithm 1 based on loops of length L and L' respectively, where $L' < L$ and L is an integer multiple of L'. Then $\mathcal{D}_{L'} \subseteq \mathcal{D}_L$. Conceptually, the case of visiting a hyper-rectangle *at least* every L steps, contains all cases that do so in $L' < L$ steps if L' divides L. This suggests that increasing the length of the loop appropriately, may yield a larger CIS.

(4) In the general case, given $L' < L$, we can have $\mathcal{D}_{L'} \nsubseteq \mathcal{D}_L$. For example, in Fig. 5a, $\mathcal{D}_3$ does not contain $\mathcal{D}_2$. A different example is provided in Fig. 5b, where we can appreciate that $\mathcal{D}_2$ is actually larger than $\mathcal{D}_3$, but again without containment. Still, as expected from point (3), $\mathcal{D}_6$ contains both $\mathcal{D}_2$, and $\mathcal{D}_3$ as shown in Fig. 5c.

(5) An important question is whether by increasing L to infinity, the computed CIS converges to the MCIS. At the moment, this is under investigation and answering this question is part of our ongoing research.

The above observations suggest the following fact: our method enforces a simple hierarchy on the computed CISs. By increasing loop length L, and *computing the union of the resulting CIS with a CIS for a smaller length, we obtain a larger CIS.* Moreover, given a loop length L, the resulting CIS contains the CISs computed for all its divisors L'. In this sense, FTSs are simply ordered by the length of their loop, or as one could think more intuitively by the amount of memory that is allowed to the algorithm.

5 COMPUTATIONAL EVALUATION

In this section, we perform a computational evaluation[3, 4] of the proposed approach. We begin with the system from Section 2.

Example 2. *Consider again the following system:*

$$x^+ = \begin{bmatrix} 1.5 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} u.$$

Let D be a subset of $\mathbb{R}^2$ as in Fig. 6, with input constraints $u \in [-1, 1]$.

This is a system in $\mathbb{R}^2$ with constraints on the input, thus can be modeled as a system in $\mathbb{R}^3$ with unconstrained input.

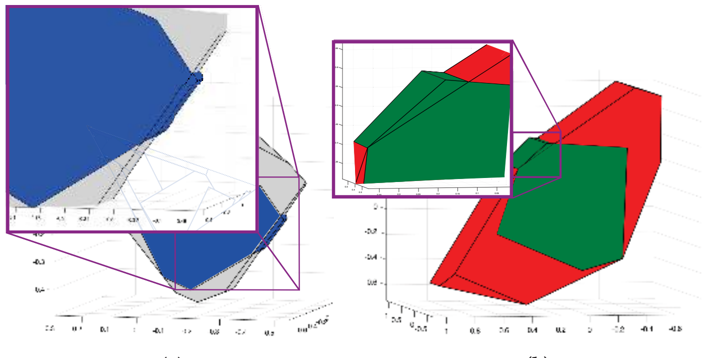
In Table 1, the first row shows how the overall computation time is impacted by increasing the states L of the utilized FTS. As expected, the larger L is, the more computation time increases -rather

[3]The code for our all examples, as well as the code that implements Algorithm 1, are available online at https://github.com/janis10/cis2m.

[4]The exact parameters of all examples in this paper can be found in https://github.com/janis10/cis2m/tree/master/HSCC20/paper-examples.

exponentially- on the grounds of projecting from an increasingly higher dimensional space to the original one. Following, the second row of Table 1 shows the ratio of the computed CIS's volume to that of the MCIS. Notice that as we increase L, the ratio approaches 1. There is thus a clear trade-off between quality and performance. Nonetheless, the gain in the size of the CIS is decreasing as L increases, indicating that in practice there is no need to consider increasingly larger FTSs.

To further demonstrate the performance and flexibility of our method, we compare with the approach proposed in [22]. There, the authors utilize Sum-Of-Squares (SOS) programming to compute ellipsoidal CISs. Although their approach was proposed for a more general class of discrete-time hybrid systems, we adapt it to the discrete-time linear systems case. For the example above, in Figure 6, the incremental union of the white set with each of the green stripes represents the CIS obtained with our method using an FTS with 1 to 5 states respectively, and in orange is the ellipsoidal CIS obtained from [22]. This figure highlights one of the advantages of our method, which stems from the proposed hierarchy. If the computed CIS is deemed insatisfactory, an incrementally larger CIS can be obtained by increasing the states of the FTS. In Figure 6 we can observe that by using $L = 2$ we already obtain a CIS larger than the one produced by the techniques in [22].


(a) (b)

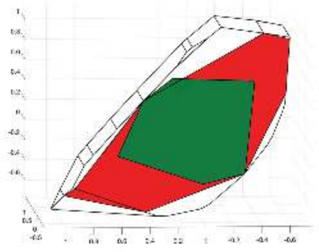

(c)

Figure 5: (a) CIS for $L = 2$ (blue), CIS for $L = 2$ (grey). (b,c) CIS for $L = 2$ (red), CIS for $L = 3$ (green), CIS for $L = 6$ (white).

Table 1: Execution times and percentage of MCIS volume captured by the CIS computed by Algorithm 1. States of the FTS range from $L = 1$ to $L = 6$. System is in $\mathbb{R}^3$.

	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$	$L = 6$
Time (sec.)	0.32	0.65	1.53	6.98	10.42	21.92
% of MCIS volume	71.59	86.24	89.72	94.81	97.55	99.72

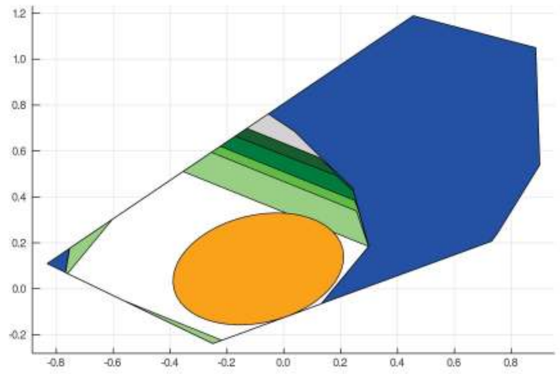


Figure 6: Comparison of our approach with [22]. Legend: Original set D (blue); MCIS of D (light gray); CISs from Algorithm 1 (white and different shades of green) using an FTS with 1 to 5 states; CIS from [22] (orange).

For the next experiment we consider the system of a truck with N trailers [30], which is shown in Fig. 7.

Example 3. *Consider the continuous-time system of a truck with N trailers:*

$$\begin{cases} \dot{d}_i = v_{i-1} - v_i \\ \dot{v}_0 = \frac{k_s}{m} d_1 - \frac{k_d}{m} v_0 + \frac{k_d}{m} v_1 + u \\ \dot{v}_i = \frac{k_s}{m}(d_i - d_{i+1}) + \frac{k_d}{m}(v_{i-1} - 2v_i + v_{i+1}) \\ \dot{v}_N = \frac{k_s}{m} d_N - \frac{k_d}{m} v_M + \frac{k_d}{m} v_{N-1} \end{cases}, i = 1, \ldots, N,$$

where velocities $v_0, \ldots, v_N$, and distances $d_1, \ldots, d_N$ are as in Fig. 7. Discretize with sampling time T_s to obtain a discrete-time linear system. The state is $x = [d_1, \ldots, d_N, v_0, \ldots, v_N]$, and hence N trailers correspond to dimension $n = 2N + 1$ of the discrete-time system.

Table 2 shows in turn how the performance of the algorithm is affected by increasing the dimension n of the system. Again, the exponential increase in computation time demonstrates how cumbersome the problem becomes in high dimensions, especially when compared for example with [22], which is significantly faster. Be that as it may, comparing the volumes of the computed CISs, reveals that the proposed approach returns considerably larger sets for $n = 3, 5$. Furthermore, it still has the ability to compute even larger sets if needed by increasing L, which could be the case for $n = 7$. Hence, depending on the application, the ability to compute larger CIS can be quite useful. As a reference, the volume of the MCIS is provided in the third row. Computing the MCIS using the

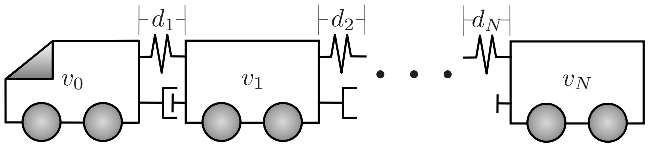


Figure 7: Illustration of a truck with N trailers.

Table 2: Execution time and CIS volume for Example 3 as the number of trailers N increases: using Algorithm 1 with $L = 2$, the algorithm in [22], and MPT3.
Dash "$-$" denotes negligibly small sets, and "NA" that the value is not available since the algorithm did not converge within the given iteration upper bound.

		$n = 3$	$n = 5$	$n = 7$	$n = 9$
Alg. 1	Time (sec.)	0.303	4.34	57.89	852.7
with $L = 2$	Volume	19.91	0.262	$4 \cdot 10^{-5}$	$-$
Alg. in	Time (sec.)	0.07	0.11	0.13	0.17
[22]	Volume	1.006	0.068	$1 \cdot 10^{-3}$	$-$
MPT3	Time (sec.)	1.05	4.5	363.2	>7200
50 iterations	Volume	21.68	20.75	NA	NA

`invariantSet()` function of the Multi-Parametric Toolbox (MPT3) [15], which is based on the iterative procedure (6), is extremely more costly computationally for higher dimensions. For this reason, we set an upper bound of 50 iterations for MPT3. Table 2 shows that even though MPT3 computes the MCIS fairly quickly for $n = 3, 5$, it failed to converge in 50 iterations for larger values of n, and simulations were aborted, thus the computed set is not a CIS.

It is worth noticing, that for our algorithm almost the entire execution time was consumed in projecting from the lifted space to the original space. This is readily understood by taking into account that the more copies of the original problem we make when composing it with FTS $\mathcal{A}$, the larger is the lifted space, and in turn the more effort it requires to return to the original space by projection [16]. This highlights though, the opportunity for significantly improving performance by exploiting the specific structure of the MCIS computed in closed-form in the higher dimensional space, which is the focus of our current research.

Remark 3. *Even though this paper considers* exact *CISs, for the sake of comparison we assessed the performance of one of the state-of-the-art algorithms [20] in computing approximations of the MCIS by solving SDPs*[5]. *Table 3 shows our findings for Example* 3, *$n = 3, 5$, and using polynomials of different degrees d. For reference, in [20], a system with similar constraints on the state and $n = 2$ is presented, and the approximations are adequate for $d \geq 8$. Therefore, given the times for $d = 8$, we conclude that just solving the SDP performs worse in terms of execution time compared to both methods in Table 2. Moreover, formulating the problem, i.e., all computations prior to calling the solver, was considerably time consuming as we required more accurate approximations in MATLAB using YALMIP [24] .*

For reference, all the simulations were conducted with an iMac (Late 2012), with a 4 core Intel Core i7 Processor @ 3.4 GHz, and 32 GB 1600 MHz DDR3 RAM. All optimization problems have been solved with MOSEK v8.1 [3].

6 DISCUSSION & CONCLUSION

In this paper, we presented a novel hierarchy for computing controlled invariant sets of discrete-time linear systems. The proposed methodology works as follows: 1) composes the system with an FTS with L states, effectively creating L copies of the problem; 2) lifts these copies to a higher dimensional space where the MCIS is computed in closed-form; 3) projects this set back to the original domain resulting in a CIS.

The resulting algorithm expresses the hierarchy via the FTS used: increasing the upper bound for the loop length, potentially yields larger CISs. In turn, this introduces a trade-off between quality, in terms of the size of the CIS, and performance, in terms of the running time. Consequently, due to this inherent flexibility, the algorithm can adapt to a specific application's requirements. Moreover, it is complete as it is guaranteed to compute a non-empty CIS if the MCIS is not empty.

The proposed hierarchy is understood as follows: computing the union of a CIS for a given loop length L, with the CIS obtained for a smaller loop length L', can return larger controlled invariant sets. In the general case, there can be instances where increasing the loop length from L' to L, does not necessarily imply that the CIS for L, by itself, will contain the CIS for L'. An exception to this, is that given a length L, the CIS based on L contains any CIS based on a divisor of L. Ideally of course, the result we would like to establish is a much stricter hierarchy based only on the upper bound of the loop length, so that a given L yields a larger CIS than any $L' < L$. To do so however, requires embedding different lengths in the same FTS, resulting in disjunctions of sets during the execution of our algorithm, something that would potentially sacrifice properties such as convexity of the computed sets, and deteriorate the performance. Moreover, as discussed in Section 4.6, it is still unknown to us if by increasing L to infinity, the computed CIS converges to the MCIS. Exploring this line of work is one of the foci of our current research.

At the same time, the performance of the proposed method can be significantly improved by exploiting the specific structure of the MCIS in the lifted space, which is computed in closed-form. Such result, is part of our future research directions towards more efficiently computing controlled invariant sets.

We anticipate no real impediments to extending the results to linear systems with disturbances, which will be part of our future work.

Last but not least, we would like to emphasize the ensuing fact: as mentioned extensively in the introduction, numerous approaches for different classes of systems are fundamentally based on the iterative procedure of [4], which always returns the MCIS. We believe that many results based on the algorithm in [4] can be generalized to use the proposed algorithm so as to benefit from guaranteed finite termination, closed-form expressions, and reduced computational cost.

Table 3: Times (sec.) to solve the SDP approximating the MCIS with the algorithm in [20], using polynomials of degree d for Example 3, and $n = 3, 5$.

	$d = 4$	$d = 6$	$d = 8$
$n = 3$	1.102	1.639	4.468
$n = 5$	1.743	12.42	126.48

[5]Towards this, we adapted the code found on https://homepages.laas.fr/henrion/ to our example.

7 DERIVATION OF THE CLOSED-FORM IN THEOREM 4.1

We demonstrate here how to derive the closed-form expression of the set $S_{\mathcal{A}}$ in (17). Initialize procedure (6) with:

$$S_0 = D^\ell = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \bar{G}x \leq \lambda_q, \\ H\lambda_q \leq f \end{array} \right\}.$$

Denote the following for the remainder of this proof:

$$q^i = \delta^i(q),$$

where δ^i is the i-fold composition of δ with itself. To compute $\mathrm{Pre}_{\Sigma'^\ell}(S_0)$, we use the auxiliary set $W(S_0)$ in (5):

$$W(S_0) = \left\{ \begin{array}{c} (x,u) \in \mathbb{R}^n \times \mathbb{R}, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \bar{G}(A_c x + B_c u) \leq \lambda_{q^1} \\ H\lambda_{q^1} \leq f \end{array} \right\}.$$

Since we assume (A_c, B_c) to be in Brunovsky normal form, in the above set of inequalities variables x and u do not appear in the same inequalities, i.e., if a row of A_c is zero, then the corresponding row of B_c is non-zero, and vice versa. Then the inequalities involving u are of the form $g_{jn}u \leq \lambda_{q^1,jn}$, $j = 1, \ldots, k$. To obtain $\mathrm{Pre}_{\Sigma'^\ell}(S_0)$ from (4), we eliminate the variable u. Using the Fourier-Motzkin Elimination method [27] we obtain:

$$\bigwedge_{p \in P_1} \bigwedge_{t \in T_1} \left(g_t \lambda_{q^1,p} - g_p \lambda_{q^1,t} \leq 0 \right), \tag{19}$$

where:

$$P_i = \left\{ p \middle| g_p = g_{j(n+1-i)} > 0 \right\},$$
$$T_i = \left\{ t \middle| g_t = g_{j(n+1-i)} < 0 \right\},$$
$$R_i = \left\{ r \middle| g_r = g_{j(n+1-i)} = 0 \right\}.$$

Let $\Gamma_1 \lambda_{q^1} \leq 0$ be equivalent to (19), then:

$$\mathrm{Pre}_{\Sigma'^\ell}(S_0) = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{ccc} & \left(\bar{G}A_c\right)_{R_1} x \leq \left(\lambda_{q^1}\right)_{R_1}, & \\ H\lambda_{q^1} \leq f & , & \Gamma_1 \lambda_{q^1} \leq 0 \end{array} \right\},$$

where the subscript R_1 denotes keeping only the rows with indices in the set R_1. Consequently, at the end of the first iteration:

$$\begin{aligned} S_1 &= S_0 \cap \mathrm{Pre}_{\Sigma'^\ell}(S_0) \\ &= \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{ccc} \bar{G}x \leq \lambda_q & , & \left(\bar{G}A_c\right)_{R_1} x \leq \left(\lambda_{q^1}\right)_{R_1}, \\ H\lambda_q \leq f & , & H\lambda_{q^1} \leq f, \\ & & \Gamma_1 \lambda_{q^1} \leq 0 \end{array} \right\}. \end{aligned}$$

Applying this procedure iteratively, at the n-th step:

$$S_n = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{ccc} \bar{G}x \leq \lambda_q, & \cdots & \left(\bar{G}A_c^n\right)_{R_n} x \leq \left(\lambda_{q^n}\right)_{R_n}, \\ H\lambda_q \leq f, & \cdots & H\lambda_{q^n} \leq f, \\ \Gamma_1 \lambda_{q^1} \leq 0, & \cdots & \Gamma_n[\lambda_{q^1} \ldots \lambda_{q^n}] \leq 0 \end{array} \right\}.$$

At this point, notice that during the next iteration of the procedure, $W(S_n)$ has the new inequality $\bar{G}(A_c^{n+1}x + A_c^n B_c u) \leq \lambda_{q^{n+1}}$. However, $A_c^{n+1} = A_c^n B = 0$, and hence no new constraints are generated by eliminating u anymore. Let us write the set S_n more compactly as:

$$S_n = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \widehat{G}_n x \leq \widehat{R}\,\lambda_q, \\ H_n \lambda_q \leq f_n, \\ \Delta_n \lambda_q \leq 0 \end{array} \right\}. \tag{20}$$

where matrices $\widehat{G}_n$, $\widehat{R}$, H_n, Δ_n, and vector f_n are of the appropriate forms. Even though eliminating u at each of the subsequent iterations does not result in the generation of new inequalities, the reader should notice that $S_n \cap \mathrm{Pre}_{\Sigma'^\ell}(S_n)$, still introduces inequalities involving λ. These inequalities are:

$$H_n \lambda_{q^+} \leq f_n \text{ and } \Delta_n \lambda_{q^+} \leq 0,$$

and consequently:

$$S_{n+1} = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{ccc} \widehat{G}_n x \leq \widehat{R}\,\lambda_q & , & \\ H_n \lambda_q \leq f_n & , & H_n \lambda_{q^+} \leq f_n, \\ \Delta_n \lambda_q \leq 0 & , & \Delta_n \lambda_{q^+} \leq 0 \end{array} \right\}.$$

Since $q^+ = \delta(q)$, and $\mathcal{A}$ consists of a loop of length L, it follows from (13) that $\delta^L(q) = q$. In a total of $n + L$ more iterations, $\mathrm{Pre}_{\Sigma'^\ell}(S_{n+L-1})$ introduces:

$$\widehat{G}_n x \leq \widehat{R}\,\lambda_{\delta^L(q)} = \lambda_q,$$
$$H_n \lambda_{\delta^L(q)} = \Delta_n \lambda_q \leq f_n,$$
$$\Delta_n \lambda_{\delta^L(q)} = \Delta_n \lambda_q \leq 0,$$

which already appear. Consequently, at this point, $S_{n+L} = S_{n+L-1} \cap \mathrm{Pre}_{\Sigma'^\ell}(S_{n+L-1}) = S_{n+L-1}$ holds, and the termination condition is met. Hereby, finite termination is proven in a total of $n + L - 1$ steps, thus obtaining:

$$S_{n+L-1} = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} \widehat{G}_n x \leq \widehat{R}\,\lambda_q \\ \widehat{H}\lambda_q \leq \widehat{f} \\ \widehat{\Gamma}\,\lambda_q \leq 0 \end{array} \right\}, \tag{21}$$

where:

- $\widehat{G}_n$, $\widehat{R}$ are as in (20),
- $\widehat{H}$, $\widehat{f}$ are such that $\widehat{H}\lambda_q \leq \widehat{f}$ is equivalent to $H\lambda_q \leq f$, $q \in Q$, and
- $\widehat{\Gamma}$ is such that $\widehat{\Gamma}\,\lambda \leq 0$ is equivalent to:

$$\Gamma_n[\lambda_{q^s} \ldots \lambda_{q^{(s+n-1)}}] \leq 0, \quad s = 0, \ldots, L-1 \quad .$$

At this point, the set computed in (21) can be simplified to some degree towards a more efficient algorithmic implementation. Let us write the above set as follows:

$$S_{n+L-1} = \bigcup_{s=0}^{L-1} S_{n+L-1,q_s},$$

where for each set S_{n+L-1,q_s} we fix the value of $q = q_s, \forall q_s \in Q$:

$$S_{n+L-1,q_s} = \left\{ \begin{array}{c} x \in \mathbb{R}^n, \\ q \in Q, \\ \lambda \in \left(\mathbb{R}^{kn}\right)^Q \end{array} \middle| \begin{array}{c} q = q_s \\ \widehat{G}_n x \leq \widehat{R}\,\lambda_q \\ \widehat{H}\lambda_q \leq \widehat{f} \\ \widehat{\Gamma}\,\lambda_q \leq 0 \end{array} \right\}. \tag{22}$$

We now show that the projections of all these sets coincide in $\mathbb{R}^n$. Towards this, we first eliminate $q \in Q$, and work in $\mathbb{R}^n \times (\mathbb{R}^{kn})^Q$. In (22), q comes into play directly only in the constraint $q = q_s$, and eliminating q is equivalent to dropping said constraint. Denote two sets with q projected out as S_{q_s}, and $S_{q_{(s+1)}} \subset \mathbb{R}^n \times (\mathbb{R}^{kn})^Q$. Now we can show that these two sets are related to each other through a rotation in the λ-coordinates. Consider a point $\left(x, \lambda_{q_0}, \ldots, \lambda_{q_{L-1}}\right) \in S_{q_s}$. By rotating the λ-coordinates we obtain $\left(x, \lambda_{q_1}, \ldots, \lambda_{q_{L-1}}, \lambda_{q_0}\right) \in S_{q_{(s+1)}}$, which proves our claim. Given that x-coordinate is invariant under these rotations, one can intuitively realize that the projections of these sets onto $\mathbb{R}^n$, i.e., the x-coordinates, are identical.

To show that, consider:

$$R = \begin{bmatrix} \mathbb{I} & 0 & \ldots & 0 \\ 0 & 0 & \ldots & \mathbb{I} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \mathbb{I} & \ldots & 0 \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} \mathbb{I} & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 \end{bmatrix},$$

where R is a permutation matrix, and P defines the orthogonal projection onto the x-coordinates. Then we write S_{q_s}, and $S_{q_{(s+1)}}$ as:

$$S_{q_s} = \left\{ (x, \lambda) \in \mathbb{R}^n \times \left(\mathbb{R}^{kn}\right)^Q \;\middle|\; G_{q_s}\,[x\ \lambda] \leq \tilde{f} \right\},$$
$$S_{q_{(s+1)}} = \left\{ (x, \lambda) \in \mathbb{R}^n \times \left(\mathbb{R}^{kn}\right)^Q \;\middle|\; G_{q_{(s+1)}}\,[x\ \lambda] \leq \tilde{f} \right\},$$

and the equality $G_{q_s} = G_{q_{(s+1)}}\,R$ follows.

Similarly, the corresponding projections on $\mathbb{R}^n$ are:

$$\pi_{R^n}\left(S_{q_s}\right) = \left\{ P\,[x\ \lambda] \;\middle|\; G_{q_s}\,[x\ \lambda] \leq \tilde{f} \right\},$$
$$\pi_{R^n}\left(S_{q_{(s+1)}}\right) = \left\{ P\,[x\ \lambda] \;\middle|\; G_{q_{(s+1)}}\,[x\ \lambda] \leq \tilde{f} \right\},$$

and for any point in $\pi_{R^n}\left(S_{q_{(s+1)}}\right)$, using the change of coordinates $R\,[x\ \lambda]$, and noticing that $P\,R = P$:

$$\pi_{R^n}\left(S_{q_{(s+1)}}\right) = \left\{ P\,[x\ \lambda] \;\middle|\; G_{q_{(s+1)}}\,[x\ \lambda] \leq \tilde{f} \right\},$$
$$\Rightarrow \pi_{R^n}\left(S_{q_{(s+1)}}\right) = \left\{ PR\,[x\ \lambda] \;\middle|\; G_{q_{(s+1)}}R\,[x\ \lambda] \leq \tilde{f} \right\},$$
$$= \left\{ P\,[x\ \lambda] \;\middle|\; G_{q_s}\,[x\ \lambda] \leq \tilde{f} \right\} = \pi_{R^n}\left(S_{q_s}\right).$$

Therefore:

$$\pi_{\mathbb{R}^n}\left(S_{n+L-1,q_s}\right) = \pi_{\mathbb{R}^n}\left(S_{n+L-1,q_{s'}}\right), \text{ for all } s, s' = 0, \ldots, L-1 .$$

Thus, it suffices to compute just one of the sets in (21) in the lifted space. Denote this set by $S_{\mathcal{A}}$:

$$S_{\mathcal{A}} = \left\{ (x, \lambda) \in \mathbb{R}^n \times \left(\mathbb{R}^{kn}\right)^Q \;\middle|\; \begin{matrix} \widehat{G}_n x \leq \widehat{R}_n \lambda \\ \widehat{H}\lambda \leq \widehat{f} \\ \widehat{\Gamma}_n \lambda \leq 0 \end{matrix} \right\}, \tag{23}$$

where:

- $\widehat{G}_n$, $\widehat{R}_n$ are such that $\widehat{G}_n x \leq \widehat{R}_n \lambda$ realizes $\bar{G}x \leq \lambda_{q_0}$, and $\left(\bar{G}A_c\right)_{R_i} x \leq \left(\lambda_{q_i}\right)_{R_i}$, $i = 1, \ldots, n$, without loss of generality, and
- $\widehat{H}$, $\widehat{f}$, and $\widehat{\Gamma}_n$ as in (21).

This concludes the proof.

REFERENCES

[1] T. Anevlavis and P. Tabuada. 2019. Computing controlled invariance sets in two moves. In *2019 IEEE Conference on Decision and Control (CDC)*. 6249–6254.

[2] Panos J. Antsaklis and Anthony Michel. 2006. *Linear Systems* (1st ed.). Birkhäuser Basel.

[3] MOSEK ApS. 2019. *The MOSEK optimization toolbox for MATLAB manual. Version 8.1.0.75.* http://docs.mosek.com/8.1/toolbox/index.html

[4] Dimitri Bertsekas. 1972. Infinite time reachability of state-space regions by using feedback control. *Automatic Control, IEEE Transactions on* AC-17 (11 1972), 604 – 613. https://doi.org/10.1109/TAC.1972.1100085

[5] F. Blanchini. 1999. Survey Paper: Set Invariance in Control. *Automatica* 35, 11 (Nov. 1999), 1747–1767. https://doi.org/10.1016/S0005-1098(99)00113-2

[6] Franco Blanchini, Fouad Mesquine, and Stefano Miani. 1995. Constrained stabilization with an assigned initial condition set. *Internat. J. Control* 62, 3 (1995), 601–617. https://doi.org/10.1080/00207179508921559 arXiv:https://doi.org/10.1080/00207179508921559

[7] Franco Blanchini and Stefano Miani. 2008. *Set–Theoretic Methods in Control.* Birkhauser, Boston, Basel, Berlin.

[8] F. Borrelli. 2003. *Constrained Optimal Control of Linear and Hybrid Systems.* Springer Berlin Heidelberg.

[9] Pavol Brunovský. 1970. A classification of linear controllable systems. *Kybernetika* 6 (1970), 173–188.

[10] E. De Santis, M. D. Di Benedetto, and L. Berardi. 2004. Computation of maximal safe sets for switching systems. *IEEE Trans. Automat. Control* 49, 2 (Feb 2004), 184–195. https://doi.org/10.1109/TAC.2003.822860

[11] S. Di Cairano and F. Borrelli. 2013. Constrained tracking with guaranteed error bounds. In *52nd IEEE Conference on Decision and Control.* 3800–3805. https://doi.org/10.1109/CDC.2013.6760469

[12] J Duncan Glover and Fred C. Schweppe. 1971. Control of Linear Dynamic Systems with Set Constrained Disturbances. *Automatic Control, IEEE Transactions on* 16 (11 1971), 411 – 423. https://doi.org/10.1109/TAC.1971.1099781

[13] Alessandro Gasparetto and Stefano Miani. 2004. Dynamic Model of a Rotating Channel Used in the Steel Industry and Implementation of a Controller. *Journal of Vibration and Control* 10, 3 (2004), 423–445. https://doi.org/10.1177/1077546304038875 arXiv:https://doi.org/10.1177/1077546304038875

[14] Jeremy H. Gillula, Shahab Kaynama, and Claire J. Tomlin. 2014. Sampling-based Approximation of the Viability Kernel for High-dimensional Linear Sampled-data Systems. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC '14).* ACM, New York, NY, USA, 173–182. https://doi.org/10.1145/2562059.2562117

[15] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. 2013. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference.* Zürich, Switzerland, 502–510. http://control.ee.ethz.ch/~mpt.

[16] C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski. 2008. On Polyhedral Projection and Parametric Programming. *Journal of Optimization Theory and Applications* 138, 2 (01 Aug 2008), 207–220. https://doi.org/10.1007/s10957-008-9384-4

[17] Shahab Kaynama, John Maidens, Meeko Oishi, Ian M. Mitchell, and Guy A. Dumont. 2012. Computing the Viability Kernel Using Maximal Reachable Sets. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '12).* ACM, New York, NY, USA, 55–64. https://doi.org/10.1145/2185632.2185644

[18] Eric Kerrigan. 2011. Robust Constraint Satisfaction: Invariant Sets and Predictive Control. *Ph.D. dissertation* (10 2011).

[19] E. C. Kerrigan and J. M. Maciejowski. 2000. Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, Vol. 5. 4951–4956 vol.5. https://doi.org/10.1109/CDC.2001.914717

[20] Milan. Korda, Didier. Henrion, and Colin N. Jones. 2014. Convex Computation of the Maximum Controlled Invariant Set For Polynomial Control Systems. *SIAM Journal on Control and Optimization* 52, 5 (2014), 2944–2969. https://doi.org/10.1137/130914565 arXiv:https://doi.org/10.1137/130914565

[21] M. Lazar. 2006. *Model predictive control of hybrid systems : stability and robustness.* Ph.D. Dissertation. Department of Electrical Engineering. https://doi.org/10.6100/IR612103

[22] Benoît Legat, Paulo Tabuada, and Raphaël M. Jungers. 2018. Computing controlled invariant sets for hybrid systems with applications to model-predictive control. In *6th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2018, Oxford, UK, July 11-13, 2018.* 193–198. https://doi.org/10.1016/j.ifacol.2018.08.033

[23] Z. Liu and N. Ozay. 2019. Safety Control with Preview Automaton. In *2019 IEEE Conference on Decision and Control (CDC).* 1557–1564.

[24] J. Lofberg. 2004. YALMIP : a toolbox for modeling and optimization in MATLAB. In *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508).* 284–289. https://doi.org/10.1109/CACSD.2004.1393890

[25] Oded Maler, Amir Pnueli, and Joseph Sifakis. 1995. On the synthesis of discrete controllers for timed systems. In *STACS 95*, Ernst W. Mayr and Claude Puech (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 229–242.

[26] Z Manna and Amir Pnueli. 1987. *A Hierarchy of Temporal Properties*. Technical Report. Stanford, CA, USA.

[27] Theodore Samuel Motzkin. 1936. *Beiträge zur Theorie der linearen Ungleichungen*. Azriel Press.

[28] A. Oustry, M. Tacchi, and D. Henrion. 2019. Inner Approximations of the Maximal Positively Invariant Set for Polynomial Dynamical Systems. *IEEE Control Systems Letters* 3, 3 (July 2019), 733–738. https://doi.org/10.1109/LCSYS.2019.2916256

[29] S. V. Rakovic, P. Grieder, M. Kvasnica, D. Q. Mayne, and M. Morari. 2004. Computation of invariant sets for piecewise affine discrete time systems subject to bounded disturbances. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, Vol. 2. 1418–1423 Vol.2. https://doi.org/10.1109/CDC.2004.1430242

[30] M. Rungger, M. Mazo, and P. Tabuada. 2012. Scaling up controller synthesis for linear systems and safety specifications. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. 7638–7643. https://doi.org/10.1109/CDC.2012.6426081

[31] Matthias Rungger, Manuel Mazo, Jr., and Paulo Tabuada. 2013. Specification-guided Controller Synthesis for Linear Systems and Safe Linear-time Temporal Logic. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control (HSCC '13)*. ACM, New York, NY, USA, 333–342. https://doi.org/10.1145/2461328.2461378

[32] M. Rungger and P. Tabuada. 2017. Computing Robust Controlled Invariant Sets of Linear Systems. *IEEE Trans. Automat. Control* 62, 7 (July 2017), 3665–3670. https://doi.org/10.1109/TAC.2017.2672859

[33] Mohamed Amin Ben Sassi and Antoine Girard. 2012. Controller synthesis for robust invariance of polynomial dynamical systems using linear programming. *Systems & Control Letters* 61, 4 (2012), 506 – 512. https://doi.org/10.1016/j.sysconle.2012.01.004

[34] S. W. Smith, P. Nilsson, and N. Ozay. 2016. Interdependence quantification for compositional control synthesis with an application in vehicle safety systems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*. 5700–5707. https://doi.org/10.1109/CDC.2016.7799145

[35] P. Tabuada and G. J. Pappas. 2006. Linear Time Logic Control of Discrete-Time Linear Systems. *IEEE Trans. Automat. Control* 51, 12 (Dec 2006), 1862–1877. https://doi.org/10.1109/TAC.2006.886494

[36] René Vidal, Shawn Schaffert, John Lygeros, and Shankar Sastry. 2000. Controlled Invariance of Discrete Time Systems. In *Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control (HSCC '00)*. Springer-Verlag, London, UK, UK, 437–450.