Scaling up controller synthesis for linear systems and safety specifications

Matthias Rungger, Manuel Mazo, Jr. and Paulo Tabuada

***Abstract*—In this paper we revisit the problem of automatically synthesizing control software enforcing formal specifications given in temporal logic. Existing approaches to solve this problem rely on the explicit or implicit construction of finite abstractions of control systems. Unfortunately, the existing abstraction techniques do not scale beyond a small number of state variables. The objective of this paper is to scale up the construction of such abstractions by focusing on linear control systems and safety specifications. In this more restricted scenario the controller synthesis problem can be reduced to the computation of control invariant subsets. Hence, we focus on the control invariance problem and propose a computational technique exploiting controllability. We illustrate the proposed methods on several synthetic examples illustrating the computational limits of our algorithm.**

I. Introduction

The study of hybrid or cyber-physical systems in which discrete and continuous components naturally interact, leads to complex problems with specifications also straddling the discrete and the continuous worlds. Such specifications (*e.g.* synchronization, task planning, constraint satisfaction) go beyond the typical objectives studied in control and can be formalized using a *temporal logic* such as Linear Temporal Logic (LTL) [12], [7].

The literature from the control systems community addressing the synthesis problem for control systems with LTL specifications has been steadily increasing in recent years [10], [18], [16], [24], [25]. The idea in this line of work is to construct a finite state system abstraction that suitably "approximates" the behavior of the original system, and then solve the synthesis problem on the finite state abstraction. However, these approaches lead, in general, to very large abstractions often too large to compute and store without super computing capabilities.

A more efficient alternative consists in attempting the synthesis directly on the infinite state system [19], [20]. Given a LTL specification, one can construct an automaton over infinite words that recognizes all the infinite words that satisfy the LTL formula. Furthermore, if one restricts attention to the safe-LTL fragment, the specification can be translated into a *bad prefixes* finite automaton. This automaton accepts at least one bad prefix for every string that violates the specification.

M. Rungger and P. Tabuada are with the Electrical Engineering Department at the University of California, Los Angeles, CA 90095 USA.

M. Mazo, Jr. is with the INCAS^3, Assen 9400 AT, The Netherlands and also with the ITM, Faculty of Mathematics and Natural Sciences, University of Groningen, Groningen 9712 GP, The Netherlands.

This work was partially supported by the NSF awards 0834771 and 1035916.

By composing the bad prefixes automaton with the continuous dynamics, regarded as an infinite-state automaton, the synthesis problem for safe-LTL can be reduced to a control invariance problem. To see why this is the case, note that an accepting string of the composed automaton corresponds to a string of the controlled system that violates the specification. Hence, enforcing the specification is tantamount to avoiding entering the set of accepting states. In other words, the objective is to remain in the set of non-accepting states forever – a control invariance problem. We spare the details of this approach but refer the interesting reader to the article [16] and references therein.

In this paper we *exclusively* focus on the control invariant set computation for linear control systems in discrete time, which is the core computational procedure for the general synthesis problem enforcing safety properties.

II. Problem Formulation and Related Work

We study discrete-time linear control systems:

$$x(t+1) = Ax(t) + Bu(t) \tag{1}$$

defined by the matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and the set of inputs $U = \mathbb{R}^m$.

Let an *initial set* $\mathcal{P} \subset \mathbb{R}^n$ be given. Our work aims at computing the *maximal control invariant set* $\mathcal{K}$ contained in $\mathcal{P}$, that is,

$$\mathcal{K} = \left\{ x \in \mathbb{R}^n \;\middle|\; \exists u \in U^{\mathbb{N}_0} : \forall t \in \mathbb{N}_0 : A^t x + \sum_{i=0}^{t-1} A^{t-i-1} B u(i) \in \mathcal{P} \right\}. \tag{2}$$

An implementable approach, introduced in [3], to compute the maximal control invariant set contained in $\mathcal{P}$ is the iteration

$$K_{i+1} = K_i \cap \mathsf{Pre}(K_i), \quad K_0 := \mathcal{P} \tag{3}$$

where the Pre operator is defined by

$$\mathsf{Pre}(K_i) := \{x \in \mathbb{R}^n \mid \exists u \in U : \; Ax + Bu \in K_i\}. \tag{4}$$

Using for example the Pompeiu-Hausdorff distance to define set convergence, the maximal control invariant set is obtained as the limit $\mathcal{K} = \lim_{i\to\infty} K_i$ (see [3]).

It is well known (see [9], Lemma 2.1), that the existence of $i \in \mathbb{N}$ such that $K_{i+1} = K_i$ implies that $\mathcal{K} = K_i$. Still the computation of $\mathcal{K}$ is a difficult problem. Even if we consider

an autonomous system (i.e. $B = 0$ in (1)) and assume that the set $\mathcal{P}$ is compact and polyhedral, so that the intersection in (3) and the Pre operator are computable, there is no guarantee that $\mathcal{K}$ can be computed in a finite number of steps (see Example 5.7 in [5]). However, there exist different relaxation results in the literature addressing this problem.

The maximal control invariant set plays an important role within finite-time optimal control problems or predictive control problems. For such problems, the invariant set is frequently used as terminal set or to ensure the recursive feasibility of the optimization problem. Due to the bounded time analysis it is often the case that one only needs to compute K_N (for a finite $N \in \mathbb{N}$) instead of $\mathcal{K}$. Thus, the iteration (3) is carried out only for a finite number of steps. Typically one assumes that $\mathcal{P}$ and U are compact polyhedra, which implies that the intersection and Pre in (3) can be computed using quantifier elimination (see [9], [6], [11], [5]).

Of course, when the invariant set is used as a terminal constraint, the finite time analysis is not sufficient and one needs to compute $\mathcal{K}$. In such cases, we can exploit a contraction property of asymptotic stabilizable systems. In [4] (see also [5] Section 5.2) the author presents a finite termination criterion of the iteration (3) for a compact polyhedral initialization $K_0 = \mathcal{P}$, assuming a certain contraction property holds for the dynamics. For example, a linear system (1) satisfies the contraction property if it is asymptotically stable.

It is well known that the iteration (3) is computationally expensive as the sets in each iteration become more complex. In trying to provide efficient approximations to the maximal control invariant a variety of parameterized invariant set computation have been proposed, see [15], [2], [14].

In all the mentioned approaches the initial set $\mathcal{P}$ is assumed to be convex. However, in our approach we require the initial set $\mathcal{P}$ to be non-convex. This stems from the symbolic synthesis framework where the elements of $\mathcal{P}$ correspond to the *atomic propositions* over which the safe-LTL formulae are defined. Therefore, in order to provide a rich language, we allow $\mathcal{P}$ to be non-convex.

Extensions of the previous mentioned algorithms to a non-convex initial set $\mathcal{P}$ can be found in [8] and [13]. However, here the computation becomes even more complex, since the non-convexity introduces a combinatorial problem: In each iteration of (3) the number of sets whose union forms K_i might grow exponentially.

In order to provide a tractable problem, we assume that the system (1) is controllable and focus on a special shaped initial set $\mathcal{P}$. That is, we assume that the set $\mathcal{P}$ is given as a union of rectangles in the state space where system (1) is in *Brunovsky normal form*, often also referred to as controller canonical form.

Due to the special form of our problem, we know all the sets that are required to represent K_i in each iteration in advance and are able to provide an efficient implementation of those sets using binary decision diagrams, see [17].

The finite termination of iteration (3), under the assumption that the system (1) is in Brunovsky normal form, has a scalar unbounded input and $\mathcal{P}$ is a rectangle in $\mathbb{R}^n$ was already observed in [22] and [23]. The result was extended to multi-input systems and union of boxes that form a partition of the state space in [19] and [20] within a symbolic synthesis framework.

We use the results described in [19], [20] as starting point and present a new and efficient algorithm for the computation of the maximal control invariant set contained in $\mathcal{P}$ with respect to (1). By restricting the set $\mathcal{P}$ to be given as a finite union of boxes (in the canonical controller space), we show that the computation of the invariant set is basically performed by a finite number of shift-register operations. Note that for a general bounded polyhedron $\mathcal{P}$ we can *approximately* compute the maximal control invariant set contained in $\mathcal{P}$ by using an under-approximation of $\mathcal{P}$ in terms of our restricted sets.

Finally, observe that we do not assume any constraints on the input set $U = \mathbb{R}^m$. However, as we also demonstrate by some examples, input constraints can be incorporated in our approach by extending the given dynamics by one state for each constrained input.

III. Special Brunovsky Normal Form

Here we remind the reader that every controllable linear system (1) can be transformed into a linear system in special Brunovsky normal form.

Definition 1. *Suppose we are given a linear control system* (1) *defined by the matrices* $A \in \mathbb{R}^{n\times n}$ *and* $B \in \mathbb{R}^m$ *and assume without loss of generality* $\mathsf{rank}(B) = m$. *We say* (1) *is in* Brunovsky normal form *if*

$$A = \begin{bmatrix} A_{\mu_1} & 0 & \dots & 0 \\ 0 & A_{\mu_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A_{\mu_m} \end{bmatrix}, B = \begin{bmatrix} b_{\mu_1} & 0 & \dots & 0 \\ 0 & b_{\mu_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & b_{\mu_m} \end{bmatrix}$$

where $\mu_1, \dots, \mu_m \in \mathbb{N}$ *satisfy* $\sum_{i=1}^m \mu_i = n$, *and the matrices* $A_{\mu_i} \in \mathbb{R}^{\mu_i \times \mu_i}$, $b_{\mu_i} \in \mathbb{R}^{\mu_i \times 1}$ *are of the form*

$$A_{\mu_i} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ \alpha_1^{\mu_i} & \alpha_2^{\mu_i} & \alpha_3^{\mu_i} & \dots & \alpha_{\mu_i}^{\mu_i} \end{bmatrix}, \quad b_{\mu_i} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}.$$

Here $\lambda^{\mu_i} - \alpha_{\mu_i}^{\mu_i}\lambda^{\mu_i}_{\mu_i-1} - \dots - \alpha_2^{\mu_i}\lambda - \alpha_1^{\mu_i}$ *is the characteristic polynomial of* A_{μ_i}, *i.e.* $\det(A - \lambda I)$.

We say (1) *is in* special Brunovsky normal form *if the last row of all* A_{μ_i} *is zero, i.e.* $\alpha_j^{\mu_i} = 0$ *for all* $i = 1, \dots, m$ *and* $j = 1, \dots, \mu_i$.

Note that the numbers $\mu_1, \dots, \mu_m$ in Definition 1 are often referred to as *controllability indices* (see [1]).

It is well known that every controllable linear system can be transformed in Brunovsky normal form by applying a coordinate transformation.

Theorem 1 ([1])**.** *For every controllable linear system* (1), *there exist an invertible linear transformation* $Q \in \mathbb{R}^{n\times n}$ *such*

that the system defined by the matrices (QAQ^{-1}, QB) is in Brunovsky normal form.

Given the system in Brunovsky normal form, we apply a linear feedback transformation $G \in \mathbb{R}^{m\times n}$ by

$$u = -Gx + v$$

where the transformation matrix is given by

$$G = \begin{bmatrix} g_1 & 0 & \dots & 0 \\ 0 & g_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & g_m \end{bmatrix}, \quad g_i^\top = \begin{bmatrix} \alpha_1^{\mu_i} \\ \alpha_2^{\mu_i} \\ \vdots \\ \alpha_{\mu_i}^{\mu_i} \end{bmatrix}. \tag{5}$$

We can formulate the corollary.

Corollary 1. *For every controllable linear system* (1)*, there exist matrices $Q \in \mathbb{R}^{n\times n}$ and $G \in \mathbb{R}^{m\times n}$, with Q invertible, such that the system defined by $(QAQ^{-1} - BG, QB)$ is in special Brunovsky normal form.*

IV. Invariant Set Computation

We assume that system (1) is given in the *special Brunovsky normal form*. The set $\mathcal{P}$ is defined in terms of boxes.

Definition 2. *A set $P \subset \mathbb{R}^n$ is called a* box *if it is defined by the Cartesian product of n intervals of the form*

$$P = [a_1,\, b_1) \times \ldots \times [a_n, b_n).$$

where the $2n$ numbers $a_1, b_1, \ldots, a_n, b_n \in \mathbb{R}$ satisfy $a_i < b_i$ for all $i = 1, \ldots, n$.

Here we arbitrarily chose the intervals to be left-closed, right-open. One can equally chose left-open, right-closed intervals to define the boxes P.

In the following, we assume that the set $\mathcal{P}$ is given by a finite union of boxes P_i by

$$\mathcal{P} = \bigcup_{i=1}^{p} P_i.$$

In case we are given an initial set $\mathcal{P}$ that is not given as a union of boxes, we have to use an under approximation of the set $\mathcal{P}$ that consists of a union of boxes.

Now we outline the algorithm to compute the largest control invariant set and refine the Pre operator for the considered setting. Furthermore, we show the finite termination of the algorithm.

Given system (1) in special Brunovsky normal form, the Pre operator (4) with respect to a box is easily computed by shifting the intervals from one dimension to the next

$$\mathsf{Pre}(P_i) = \prod_{i=1}^{m} \mathbb{R} \times [a^i_{\nu_i+1},\, b^i_{\nu_i+1}) \times \ldots \times [a^i_{\nu_i+\mu_i}, b^i_{\nu_i+\mu_i})$$

where the numbers $\nu_1, \ldots, \nu_m$ are computed from the controllability indices $\mu_1, \ldots, \mu_m$ by $\nu_i = \sum_{j=1}^{i-1} \mu_j$.

Example 1. Consider system (1) in special Brunovsky normal form with $\mu_1 = \mu_2 = 2$. The system matrices are given by

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Given a box $P = [a_1, b_1) \times \ldots \times [a_n, b_n)$, the Pre operator results in

$$\mathsf{Pre}(P) = \mathbb{R} \times [a_1, b_1) \times \mathbb{R} \times [a_3, b_3) \qquad \square$$

It is straightforward to see that the Pre operator for the set $\mathcal{P}$ is simply given by

$$\mathsf{Pre}(\mathcal{P}) = \bigcup_{i=1}^{p} \mathsf{Pre}(P_i).$$

We describe the procedure which we use to compute the maximal control invariant set of $\mathcal{P}$ with respect to (1) in Algorithm 1. The algorithm takes as input the numbers $a_1^i, b_1^i, \ldots, a_n^i, b_n^i \in \mathbb{R}$ of the intervals defining $\mathcal{P}$ and the controllability indices $\mu_1, \ldots, \mu_m \in \mathbb{N}$. The controllability indices are needed in the Pre computation. The output is the set K_k that satisfies $K_k = K_{k+1}$ from which follows that $K_k = \mathcal{K}$.

Algorithm 1 Computation of $\mathcal{K}$

Input: $a_1^i, b_1^i, \ldots, a_n^i, b_n^i$; $\mu_1, \ldots, \mu_m$;
Init: $K_0 := \bigcup_i [a_1^i,\, b_1^i) \times \ldots \times [a_n^i, b_n^i)$
  while $K_k \neq K_{k+1}$ **do**
    $K_{k+1} = K_k \cap \mathsf{Pre}(K_k)$
  end while
Output: K_k

We show in the following theorem that Algorithm 1 terminates in a finite number of steps.

Theorem 2. *Suppose system* (1) *is in special Brunovsky normal form and $\mathcal{P} \subset \mathbb{R}^n$ is given by a finite union of boxes. Then Algorithm 1 terminates in a finite number of iterations and the output K_k is the maximal control invariant set contained in $\mathcal{P}$ with respect to* (1)*.*

Proof: Let $\mathcal{D}$ denote the set of boxes that are given by all possible combinations of $2n$ numbers out of the set

$$N = \{a_1^1, \ldots, a_n^1, \ldots, a_1^p, \ldots, a_n^p, b_1^1, \ldots, b_n^1, \ldots, b_1^p, \ldots, b_n^p\}.$$

Note that $\mathcal{D}$ is finite. We show by induction that for all $k \geq 0$ the sets K_k are given as finite union of boxes $P_i^k \in \mathcal{D}$, i.e.,

$$K_k = \bigcup_{i=1}^{p_k} P_i^k.$$

The base case $K_0 := \mathcal{P}$ is satisfied by definition. Now suppose K_k satisfies the claim. Then K_{k+1} is given by

$$\begin{aligned} K_{k+1} &= \left(\cup_{i=1}^{p_k} P_i^k\right) \bigcap \left(\cup_{i=1}^{p_k} \mathsf{Pre}(P_i^k)\right) \\ &= \bigcup_{i=1}^{p_k} \bigcup_{j=1}^{p_k} P_i^k \cap \mathsf{Pre}(P_j^k). \end{aligned}$$

A straightforward computation (using the special Brunovsky normal form) shows that K_{k+1} is again given as finite union of boxes $P_i^{k+1} \in \mathcal{D}$.

The iteration (3) can be seen as the iteration of the operator $\mathcal{T}$ taking K to $K \cap \mathsf{Pre}(K)$. Moreover, $\mathcal{T}$ acts on the Boolean algebra generated by $\mathcal{D}$ which is a finite lattice. Existence of a fixed point of operator $\mathcal{T}$ and termination in finitely many steps is now a consequence of Tarski's fixed-point theorem since $K_{i+1} \subset K_i$ and the finiteness of $\mathcal{D}$, see [21]. It is easy to see that $\mathcal{K}$ corresponds to the fixed-point of $\mathcal{T}$. ∎

From the proof of Theorem 2 it follows that the output set K_k of Algorithm 1 is either empty, or there exist numbers $a_1^j, b_1^j, \ldots, a_n^j, b_n^j \in \mathbb{R}$ such that $K_k = \bigcup_{j=1}^{p_k} [a_1^j,\, b_1^j) \times \ldots \times [a_n^j, b_n^j)$.

V. Implementation

In what follows we describe how the algorithm devised in the previous section can be efficiently implemented in a symbolic manner. We use Binary Decision Diagrams (BDD), a data structure that efficiently represents binary functions $f : \mathbb{B}^n \to \mathbb{B}$. By resorting to BDDs we perform all the operations required to compute the largest invariant set in a symbolic manner.

From Algorithm 1 given in the previous section, it is clear that all the sets we need to manipulate are given as unions of boxes, *i.e.*

$$K = \bigcup_{i=1}^{p} P_i,\ P_i = [a_1^i, b_1^i) \times [a_2^i, b_2^i) \times \cdots \times [a_n^i, b_n^i).$$

Note that each of the boxes P_i are completely determined by two points in $\mathbb{R}^n$, namely $a^i = (a_1^i, a_2^i, \ldots, a_n^i)$ and $b^i = (b_1^i, b_2^i, \ldots, b_n^i)$. Let K be given by the constants a_j^i, b_j^i, where $i \in \{1, 2, \ldots, p\}$ and $j \in \{1, 2, \ldots, n\}$. Note that all the intermediate sets generated by Algorithm 1 can be represented using only those same constants. We denote by N_e the cardinality of the set $\bigcup_{i,j}(\{a_j^i\} \cup \{b_j^i\})$, and by $N_b = \lceil \log N_e \rceil$.

A. Set encoding

To enable an efficient symbolic manipulation of the sets in the proposed algorithm, we encode them as BDDs. We start by ordering all the constants a_j^i, $b_j^i \in \mathbb{R}$ necessary to define sets. Let the vector $v \in \mathbb{R}^l$ contain the constants ordered in ascending order, *i.e.* $v_i \leq v_{i+1}$. We construct a set of basic intervals $\mathcal{I} = \bigcup_{i=1}^{l-1} s_i$, where $s_i = [v_i, v_{i+1})$. Any of the sets generated by Algorithm 1 can be constructed as the union of Cartesian products of intervals in $\mathcal{I}$. Note that $\mathcal{I}$ is a finite set, and thus we can encode its elements in binary using a finite number of bits N_b. For clarity of the exposition, let us use a *hashing* function $h : \mathbb{B}^{N_b} \to \mathcal{I}$ such that given a binary string d associated with an interval $s \in \mathcal{I}$, the function h returns the corresponding interval *i.e.* $h(d) = s$.

We encode a set $K \subset \mathbb{R}^n$ as a BDD $\mathbf{K} : \mathbb{B}^{nN_b} \to \mathbb{B}$ as:

$$\mathbf{K}(d_1, d_2, \ldots, d_n) = 1 \ \Leftrightarrow \forall\, x \text{ s.t. } x_i \in h(d_i),\, x \in K.$$

B. Symbolic operations

In order to symbolically implement the proposed algorithm, and once the initial safe set is encoded as shown in the previous section, we need to symbolically implement the operators $\cap$ and Pre.

Computing the set resulting from the intersection of two sets K and K', $K'' = K \cap K'$ can be realized computing the conjunction of the two BDD's representing these sets, *i.e.*: $\mathbf{K}'' = \mathbf{K} \wedge \mathbf{K}'$.

The computation of the $\mathsf{Pre}(K)$ of a given set K can also be computed efficiently on the BDD representation selected. For clarity of the exposition, let us just explain how this is done for single input systems. The Pre operation for boxes, as shown before, reduces to:

$$x = (x_1, x_2, \ldots, x_n) \in \mathsf{Pre}(K) \Leftrightarrow \exists\, q \,:\, (x_2, x_3, \ldots, x_n, q) \in K$$

Thus, the Pre can be computed on the BDD through an existential quantifier elimination:

$$\bar{\mathbf{K}}(d_1, d_2, \ldots, d_n) = 1 \Leftrightarrow \exists\, d_n\, \mathbf{K}(d_1, d_2, \ldots, d_n) = 1;$$

and a variable reordering:

$$\mathsf{Pre}(\mathbf{K})(d_1, d_2, \ldots, d_n) = 1 \ \Leftrightarrow \bar{\mathbf{K}}(d_2, d_3, \ldots, d_n, d_1) = 1.$$

Note that after the quantifier elimination, the inputs d_n in $\bar{\mathbf{K}}(d_1, d_2, \ldots, d_n)$ do not affect the evaluation of $\bar{\mathbf{K}}$. Quantifier elimination, conjunction and variable reordering are standard, and efficient, operations in BDD manipulation, and are supported in standard BDD packages as CUDD [17].

VI. Computation Experiments

In this section we demonstrate our approach by different examples. In the first example, we compute the largest control invariant set of a mobile robot moving in the plane. Subsequently, in order to test the scalability of our implementation we test it on a number of synthetic problems randomly generated. All the results reported in what follows were obtained on a computer with 4GB of RAM and a 2.66 GHz Intel Core 2 Duo CPU.

A. Mobile robot

As a first example, we study a mobile robot moving in the plane $(x, y) \in \mathbb{R}^2$. We assume that the robot is able to move omnidirectional in each direction and consider the velocity as the input, such that the model is given by simple integrator dynamics

$$\begin{bmatrix} x(t+1) \\ y(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix}.$$

In order to be able to enforce input constraints we extend the state space by one dimension for each input and define $x_1 = x$, $y_1 = y$, $x_2 = v_x$ and $y_2 = v_y$. Furthermore, we introduce two

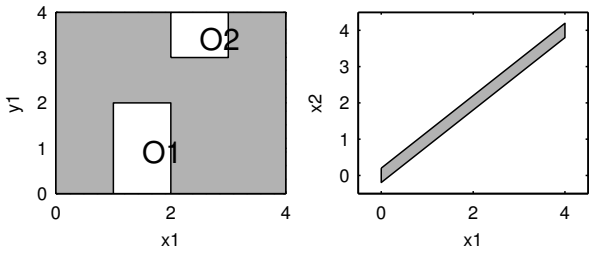

Fig. 1. Left: The workspace $\mathcal{S}$ projected onto x_1 and y_1 in the original coordinates. Right: The workspace $\mathcal{S}$ projected onto x_1 and x_2 in the coordinates of the Brunovsky normal.

new inputs u_1, u_2 and obtain a four-dimensional difference equation

$$\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ y_1(t+1) \\ y_2(t+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ y_1(t) \\ y_2(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}.$$

In our scenario, the robot is moving in an indoor environment with a length and height of $4\,\mathrm{m}$. This together with the input constraints $-0.2 \leq v_x, v_y \leq 0.2$ define the workspace $\mathcal{S}$ as rectangle

$$[0,\, 4] \times [-0.2,\, 0.2] \times [0,\, 4] \times [-0.2,\, 0.2]$$

excluding the obstacles

$$[1,\, 2] \times \mathbb{R} \times [0,\, 2] \times \mathbb{R}, \quad \text{and} \quad [2,\, 3] \times \mathbb{R} \times [3,\, 4] \times \mathbb{R}.$$

We illustrate a projection of the robot's workspace $\mathcal{S}$ onto x_1 and y_1 in the left sub-figure of Figure 1 by the gray shaded region.

Now we transform the robot system into the special Brunovsky normal form. The system is controllable and the controllability indices are given by $\mu_1 = \mu_2 = 2$. We follow the steps of Section III to obtain the state space transformation matrix

$$Q = \begin{bmatrix} Q' & 0 \\ 0 & Q' \end{bmatrix} \quad \text{with} \quad Q' = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}.$$

The entries of the feedback transformation matrix (5) are given by $g_1 = g_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}$. A projection of the workspace $\mathcal{S}$ of the robot under the transformation Q is illustrated in the right sub-figure of Figure 1. We *under* approximate the workspace in the transformed state space by a union of boxes. By using an under-approximation we ensure that the control invariant set is inside the workspace. We list the computational results in

TABLE I
COMPUTATIONAL RESULTS FOR THE ROBOT EXAMPLE.

$\mathsf{vol}(\mathcal{P})/\mathsf{vol}(\mathcal{S})$	$\lvert\mathcal{P}\rvert/10^6$	$[s.]$	$[kByte]$
0.86	0.26	8	70
0.90	0.99	35	102
0.95	17.5	631	201

Table I. Each row corresponds to a computation using a different number of boxes to approximate the workspace $\mathcal{S}$ in the Brunovsky normal form coordinates. The first entry represents the quality of the approximation, i.e., the ratio between the volume of the original workspace and the approximation. The second entry corresponds to the number of boxes used in the approximation. The time to compute the largest control invariant set and required memory to store the BDD that represents the invariant set are listed in entry three and four, respectively.

Observe that the number of boxes increases drastically with the quality of the approximation. We believe that this is a result of our naive implementation of the approximation algorithm (we use uniformly sized boxes) and can be avoided by using local refinement strategies at the boundary of the initial set.

We show the approximation $\mathcal{P}$ of the workspace for $\mathsf{vol}(\mathcal{P})/\mathsf{vol}(\mathcal{S}) = 0.86$ in the left sub-figure of Figure 2. We range over x_1 and x_2 and fix $y_1 = 1$ and $y_2 = 0$. The maximal control invariant set is illustrated in the right sub-figure again of Figure 2. One can clearly observe the integrator dynamics

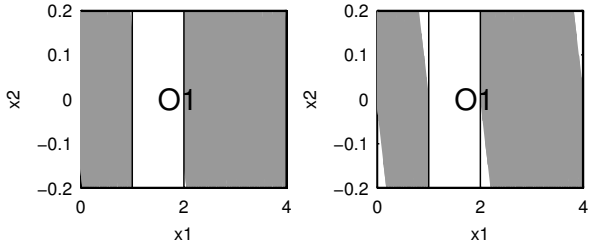

Fig. 2. Initial approximation $\mathcal{P}$ of $\mathcal{S}$ (left) and maximal control invariant set $\mathcal{K}$ ranging over x_1, x_2 with fixed $y_1 = 1$ and $y_2 = 0$ (right).

of the system.

B. Scalability

Here we assume that the system's dynamics is given already in the special Brunovsky normal form, the only synthetic data that needs to be generated is the set $\mathcal{P}$ for which the maximal invariant set needs to be found, and the controllability indexes of the system. We run first a number of experiments on systems with dimensions ranging from $n = 2$ to $n = 7$, and with number of inputs ranging from $m = 1$ to $m = 3$. In each of those experiments we selected sets $\mathcal{P}$ formed as a union of boxes given by intervals taken from a finite set $\mathcal{I}$. The set $\mathcal{I}$ only contains non-intersecting intervals of $\mathbb{R}$ whose union is a compact subset of $\mathbb{R}$. The total number of boxes that can be selected as part of the set $\mathcal{P}$ is $n^{|\mathcal{I}|}$, thus one can define a notion of density ρ of the set $\mathcal{P}$ in the full space $\mathbb{R}^n$ as $\rho = \frac{p}{n^{|\mathcal{I}|}}$, where p denotes the number of boxes forming the set $\mathcal{P}$. We run 20 experiments for each of the values of n, m and ρ, and fixed $|\mathcal{I}| = 10$. The average computation times of these experiments are reported in Tables II, III, IV.

These tables show how the approach scales better in ρ than in dimensionality. This is a natural consequence of the *curse of dimensionality*. With a fixed $|\mathcal{I}|$, when increasing the dimension of the system the number of possible boxes to describe the set $\mathcal{P}$ increases exponentially. Yet, thanks to the BDD representation this complexity is somehow mitigated.

TABLE II
AVERAGE RUNNING TIME IN [S.] FOR $m = 1$ AND $|\mathcal{I}| = 10$.

$n \setminus \rho$	0.05	0.1	0.15	0.2	0.25	0.3
2	0.016	0.012	0.011	0.012	0.009	0.010
3	0.011	0.012	0.013	0.014	0.014	0.018
4	0.027	0.058	0.066	0.065	0.074	0.081
5	0.202	0.611	0.745	0.821	0.915	1.037
6	2.801	9.320	11.85	13.26	15.76	19.07
7	42.45	166.8	244.8	292.6	365.2	431.8

TABLE III
AVERAGE RUNNING TIME IN [S.] FOR $m = 2$ AND $|\mathcal{I}| = 10$.

$n \setminus \rho$	0.05	0.1	0.15	0.2	0.25	0.3
2	0.011	0.010	0.010	0.011	0.01	0.012
3	0.012	0.012	0.013	0.012	0.01	0.013
4	0.023	0.034	0.044	0.059	0.07	0.076
5	0.177	0.324	0.473	0.625	0.78	0.932
6	2.902	5.495	8.166	11.09	13.87	16.76
7	45.43	83.09	120.2	190.4	291.8	371.0

TABLE IV
AVERAGE RUNNING TIME IN [S.] FOR $m = 3$ AND $|\mathcal{I}| = 10$.

$n \setminus \rho$	0.05	0.1	0.15	0.2	0.25	0.3
3	0.013	0.011	0.013	0.013	0.018	0.013
4	0.029	0.034	0.043	0.055	0.066	0.075
5	0.170	0.323	0.475	0.629	0.777	0.931
6	2.710	5.482	8.232	11.08	12.99	15.21
7	39.07	77.87	117.2	183.9	282.8	386.2

TABLE V
TIME IN [S.] FOR $p = 10^6$ AND $|\mathcal{I}| = 10$.

n	14	16	18	20
[s.]	673.5	1316	1576	2134

In order to test further the dimensional scalability of our approach, we fixed the number $p = 10^6$ and the number of inputs to $m = 2$ and ran again experiments with increasing dimensions of the system. Because of the long time needed to run each of this experiments, in Table V we only show the time it took for one randomly generated experiment to finalize. These experiments illustrate the capability of solving high-dimensional problems in a reasonable time.

VII. FINAL REMARKS

We have introduced a new algorithm to compute the maximal control invariant set with respect to discrete-time controllable linear systems. We have shown the finite termination of the algorithm by imposing certain assumptions on the initial set and using the controllability of the system. Specifically, we restrict the initial set to be given as union of boxes in the coordinates of the Brunovsky normal form. For a general initial set, we compute an approximation of the maximal control invariant set by using an under-approximation of the initial set. In the last section we have demonstrated that the approach scales well with the dimension and number of boxes. We were able to solve problems in 7 dimensions with $\mathcal{P}$ given by 10^8 boxes on a standard desktop computer within 5 to 6 minutes. Furthermore, when we kept the number of boxes constant to 10^6 we could reach up to 20 dimensions. However, the required number of boxes to approximate general polyhedra grows exponentially with the state space dimension and depends strongly on the shape of the initial set.

We are currently investigating efficient procedures to construct approximations of arbitrary polyhedra by unions of multi-resolution boxes which enable us to compare our results with results reported in the literature, e.g. [13].

REFERENCES

[1] P. J. Antsaklis and A. N. Michel. *A Linear Systems Primer*. Birkhäuser, 2007.

[2] N. Athanasopoulos and G. Bitsoris. Invariant set computation for constrained uncertain discrete-time linear systems. In *Proc. of the 49th IEEE Conf. on Decision and Control*, pages 5227–5232, 2010.

[3] D. Bertsekas. Infinite time reachability of state-space regions by using feedback control. *IEEE TAC*, 17:604–613, 1972.

[4] F. Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced lyapunov functions. In *Proc. of the 30th Conf. on Decision and Control*, pages 1755–1760, 1991.

[5] F. Blanchini and S. Miani. *Set-Theoretic Methods in Control*. Systems & Control: Foundations & Applications. Birkhäuser, 2008.

[6] F. Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*. LNCIS. Springer, 2003.

[7] E. M. M. Clarke, D. Peled, and O. Grumberg. *Model Checking*. MIT Press, 1999.

[8] T. Geyer, F. D. Torrisi, and M. Morari. Optimal complexity reduction of polyhedral piecewise affine systems. *Automatica*, 44:1728–1740, 2008.

[9] E. C. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, Department of Engineering, University of Cambridge, 2000.

[10] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE TAC*, 53:287–297, 2008.

[11] M. Lazar. *Model predictive control of hybrid systems:stability and robustness*. PhD thesis, Technische Universiteit Eindhoven, 2006.

[12] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer, 1992.

[13] E. Pérez, C. Ariño, F. X. Blasco, and M. A. Martínez. Maximal closed loop admissible set for linear systems with non-convex polyhedral constraints. *Journal of Process Control*, pages 529 – 537, 2011.

[14] S. V. Raković and M. Baric. Parameterized robust control invariant sets for linear systems: Theoretical advances and computational remarks. *IEEE TAC*, 55:1599–1614, 2010.

[15] S. V. Raković, E. C. Kerrigan D. Q. Mayne, and K. I. Kouramas. Optimized robust control invariance for linear discrete-time systems: theoretical foundations. *Automatica*, 43:831–841, 2007.

[16] P. Roy, P. Tabuada, and R. Majumdar. Pessoa 2.0: a controller synthesis tool for cyber-physical systems. In *HSCC*, pages 315–316, 2011.

[17] F. Somenzi. *CUDD: CU Decision Diagram Package. Release 2.5.0*. University of Colorado at Boulder, 2012. http://vlsi.colorado.edu/~fabio/CUDD/.

[18] P. Tabuada. *Verification and Control of Hybrid Systems – A Symbolic Approach*. Springer, 2009.

[19] P. Tabuada and G. J. Pappas. Model checking LTL over controllable linear systems is decidable. In *HSCC*, LNCS, pages 498–513. Springer, 2003.

[20] P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. *IEEE TAC*, 51:1862–1877, 2006.

[21] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[22] R. Vidal, S. Schaffert, J. Lygeros, and S. Sastry. Controlled invariance of discrete time systems. In *HSCC*, LNCS, pages 437–451. Springer, 2000.

[23] R. Vidal, S. Schaffert, O. Shakernia, J. Lygeros, and S. Sastry. Decidable and semi-decidable controller synthesis for classes of discrete time hybrid systems. In *Proc. of the 40th IEEE Conf. Decision and Control*, pages 1243–1248, 2001.

[24] T. Wongpiromsarn, U. Topcu, and N. Ozay. Receding horizon control for temporal logic specifications. In *HSCC*, pages 101–110, 2010.

[25] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray. TuLiP: a software toolbox for receding horizon temporal logic planning. In *HSCC*, pages 313–314, 2011.