Specification-Guided Controller Synthesis for Linear Systems and Safe Linear-Time Temporal Logic*

Matthias Rungger
Department of Electrical Engineering
UCLA
rungger@ee.ucla.edu

Manuel Mazo Jr.
Delft Center for Systems and Control
Delft University of Technology
m.mazo@tudelft.nl

Paulo Tabuada
Department of Electrical Engineering
UCLA
tabuada@ee.ucla.edu

ABSTRACT

In this paper we present and analyze a novel algorithm to synthesize controllers enforcing linear temporal logic specifications on discrete-time linear systems. The central step within this approach is the computation of the maximal controlled invariant set contained in a possibly non-convex safe set. Although it is known how to compute approximations of maximal controlled invariant sets, its exact computation remains an open problem. We provide an algorithm which computes a controlled invariant set that is guaranteed to be an under-approximation of the maximal controlled invariant set. Moreover, we guarantee that our approximation is at least as good as any invariant set whose distance to the boundary of the safe set is lower bounded. The proposed algorithm is founded on the notion of sets adapted to the dynamics and binary decision diagrams. Contrary to most controller synthesis schemes enforcing temporal logic specifications, we do not compute a discrete abstraction of the continuous dynamics. Instead, we abstract only the part of the continuous dynamics that is relevant for the computation of the maximal controlled invariant set. For this reason we call our approach specification guided. We describe the theoretical foundations and technical underpinnings of a preliminary implementation and report on several experiments including the synthesis of an automatic cruise controller. Our preliminary implementation handles up to five continuous dimensions and specifications containing up to 160 predicates defined as polytopes in about 30 minutes with less than 1GB memory.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods and Search**]: Control Theory; I.2.2 [**Automatic Programming**]: Program Synthesis

*The work of the first and last authors was partially supported by the NSF award 1035916 and by the NSF Expeditions in Computing project ExCAPE: Expeditions in Computer Augmented Program Engineering.

HSCC'13, April 8–11, 2013, Philadelphia, Pennsylvania, USA.

Keywords

Controller Synthesis, LTL Specification, Set Invariance

1. INTRODUCTION

Traditional control systems design is concerned with rather simple specifications such as asymptotic stability, *i.e.*, reaching some set as time tends to infinity. Although simple, these specifications are often hard to enforce due to the complexity of the models employed in control theory and, in particular, due to the infinite nature of the state space over which those models are defined. In sharp contrast is the work on formal verification of systems or in discrete-event controller synthesis. In those two realms, the problem specification is typically much more complex, involving not only specifications regarding the limiting behavior of the system at hand, but also sequencing of actions, choices between alternative actions, etc. These specifications, are typically formalized in a temporal logic [25] such as linear temporal logic (LTL), which lets the user specify both logical and temporal properties. In those areas of research more complex specifications can be tackled because of the finite state nature of the system models. The use of systems evolving on finite state spaces allows for the algorithmic treatment of problems and therefore the use of computers for the computation of its solution or be it the verification of a given property, or the design of a control strategy to enforce a property.

With the advent of ubiquitous computing platforms to sense and control physical systems, there has been an increasing level of concern regarding the safety of such systems. This has, in turn, sparked the interest of computer scientists to verify the correctness of software interacting with the physical world, as well as from control scientists aiming at synthesizing software that guarantees the safe control of a given physical plant. Until recently, most of the approaches to these problems relied on extensive testing to prove correctness of the designs at hand. In the past few years, however, there has been a surge of research aimed at the synthesis of correct-by-design control software. Under this new paradigm, testing or verification is no longer needed as the software is designed taking into account all possible contingencies described by the models of the system to be controlled.

As already mentioned, when the models employed to describe the system at hand have finitely many states, automated techniques are already available to synthesize controllers. Thus, the most common approach to provide correct-by-design synthesis techniques is to convert the infinite state models, usually employed in the modeling of physi-

cal processes, into finite state models. This translation, referred to as abstraction, needs to be carefully done so that properties guaranteed in the simpler finite state model carry through to the original infinite state system. While numerous theoretical results are available supporting such abstraction procedures [31], direct application of them, imposing uniform grids on the state space, usually results in computationally intractable models. This is especially the case when dealing with higher dimensional systems due to the *curse of dimensionality* resulting in an exponential growth of the finite models with the dimension of the system. There exists a great variety of different approaches [23, 33, 16, 12, 10, 22, 1, 27, 36, 20, 40, 35] and tools, *e.g.* LTL-Con [15], LTLMoP [11], TuLiP [41], Pessoa [21]. All of them with their different virtues. The systems considered range from simple double integrator dynamics [10], over linear dynamics [33, 16, 40] or nonlinear dynamics [22, 27, 20, 35] to hybrid dynamics [12] and stochastic systems [1, 36]. Different notion of abstractions like behavioral containment [23, 27], exact (bi)similar relations [33] or approximate notions [12, 22, 20] have been analyzed. Also the regarded specification language ranges from full LTL [33, 10], restricted versions of LTL [22, 40], to special cases of reach-avoid problems [27, 35]. However, most of them suffer from the curse of dimensionality, as they follow the same two step approach by first computing a finite abstraction and performing controller synthesis afterwards. Alternative approaches are thus needed in order to make the synthesis of correct-by-design controllers practical.

In the present work, we focus on an algorithm that avoids the computation of a finite state abstraction of the continuous system to be controlled. A similar approach, which also connects the computation of the abstraction to the specification is outlined in [13], where the authors focus on co-safe LTL specifications. We consider discrete-time linear time-invariant systems and a subset of the full linear temporal logic (LTL) called safe-LTL. Safe-LTL formulas always specify a safety property. These are properties whose violation can be checked by looking at a finite prefix of the violating run. The well-known automata theoretic approach to synthesize controllers that enforce LTL specifications, proceeds as follows, see [17, 19]. First, the negated specification is translated into a finite state automaton, which is then synchronized with the system. If the synchronized system has an accepting run, the property given by the specification does not hold. Hence, the objective of a controller is to prevent runs from being accepted which can be formalized as safety game or as the computation of the maximal controlled invariant set.

Therefore, we revisit the well-known fixed point iteration to compute the maximal controlled invariant subset (MCIS) of a given *safe* set K, see *e.g.* [3] or for a more detailed exposition the monographs [2, 5]. When the system is linear and the set K is polyhedral, all the intermediate computations can be performed exactly by computing polyhedral projections and intersections, see *e.g.* [37]. Thus, provided that the iterations terminate, we can solve the synthesis problem. Unfortunately, termination is not guaranteed in general. There exists some results which address this problem for a convex safe set K. The authors in [4, 5] exploit the contraction property of an asymptotically stabilizable system, and show that the MCIS can be under-approximated with arbitrary accuracy in finite time. The authors in [9], initialize the iteration with a controlled invariant set, and are able to provide an invariant under-approximation of the MCIS in every step of the iteration. Moreover, they show that their approximation approaches the MCIS in the limit. Unfortunately, in our case, we cannot assume the safe set K to be convex as it is given by unions and intersections of the polyhedral sets that correspond to the atomic propositions in a safe-LTL formula.

The fixed point iteration for a non-convex safe set is often carried out for piecewise linear systems [14, 26, 24, 42]. However, none of these approaches ensure finite termination. Furthermore, it is not straightforward to apply the solutions for convex safe sets to non-convex safe sets. Also, from a practical point of view, the exact computation of the MCIS with non-convex safe sets might not be a wise choice since the number of polytopes might grow combinatorially over the iterations. This constitutes a serious problem as illustrated in Section 6.2.

We propose a practical implementation of the MCIS computation for which we can guarantee termination. This is attained at the price of providing an under-approximation of the actual MCIS which is, nevertheless, a controlled invariant subset. The proposed technique is based on the notion of sets adapted to the dynamics of a linear system, see [31]. These are sets which in controllable normal form coordinates, also known as Brunvosky normal form, are Cartesian products of intervals. For (unions of) such sets, the computation of the MICS is relatively simple, and can be computed in a finite number of iterations. This was already observed in [37] and [38], and later generalized in [32, 33] and [28]. However, one rarely encounters problems in which the relevant sets are already adapted to the dynamics as assumed in [32, 33] and [28]. The idea we explore in this paper is to under-approximate the relevant sets by adapted sets and then perform the fixed point computation for the approximating sets. The approximations of the sets at hand are represented much in the same flavor as proposed in [6] and are stored in the form of binary decision diagrams [39, 30] (BDD). BDDs are selected as a data structure because of their efficiency in representing and manipulating binary functions or equivalently discrete sets.

One aspect of our work is that we do not require the computation of the complete discrete abstraction of the continuous dynamics. Instead, we abstract only the part of the continuous dynamics that plays a role in the computation on the MCIS. The computational implications of this are reflected on the experimental results, where we can solve problems up to five continuous variables, which was previously not possible, see Section 6.2. A second aspect of the present work is a certain completeness result for our approach: we show that our approximation is at least as good as any invariant set whose distance to the boundary of the safe set is lower bounded, see Section 4.2. A statement that is rarely found in the abstraction based synthesis community.

2. PRELIMINARIES

In what follows, we mostly operate in a space $X = Q \times \mathbb{R}^n$, given as the product of a discrete space Q and the Euclidean space $\mathbb{R}^n$. The projection of X onto Q and $\mathbb{R}^n$ is denoted by the mappings $\pi_Q : X \to Q$ and $\pi_{\mathbb{R}^n} : X \to \mathbb{R}^n$, respectively. Given a subset $K \subseteq X$ we use $K(q)$ to denote the set $K(q) = \{x \in \mathbb{R}^n \mid (q, x) \in K\}$. We define the Hausdorff distance $\mathsf{d_H}$ on X with respect to the product metric given by the

discrete metric on Q and the Euclidean distance d in $\mathbb{R}^n$. A cube in $\mathbb{R}^n$ with center $c \in \mathbb{R}^n$ and radius $r \in \mathbb{R}^n$, is denote defined by $\mathsf{B}(c, r) := \{x \in \mathbb{R}^n \mid \forall_{i\in\{1,\dots,n\}} : -r_i \leq x_i - c_i \leq r_i\}$.

3. THE SYNTHESIS PROBLEM

In this section, we introduce the synthesis problem, which is composed of three entities: a linear control *system* that describes the physical process we want to control; a set of *atomic propositions* that represent the sets of states that are of interest for the control task; and the safe-LTL *formula* that describes the desired closed-loop behavior of the system in terms of the atomic propositions.

3.1 The System

We consider discrete-time linear control systems:

$$x^+ = Ax + Bu, \tag{1}$$

given by the matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$. Throughout the paper, we assume the system (1) to be controllable.

3.2 The Specification

Let $\mathcal{P} \subseteq 2^{\mathbb{R}^n}$ denote the set of *atomic propositions*, with each element $P_i \in \mathcal{P}$ given by a convex polytope:

$$P_i = \mathcal{H}(C_i, c_i). \tag{2}$$

Here $\mathcal{H}(C_i, c_i) = \{x \in \mathbb{R}^n : C_i x \leq c_i\}$ denotes the set of solutions of the system of inequalities $C_i x \leq c_i$ with $C_i \in \mathbb{R}^{r_i\times n}$ and $c_i \in \mathbb{R}^{r_i}$. We assume that the sets $P_i \subseteq \mathbb{R}^n$ are bounded. The atomic propositions are linked to the system (1) through the map $h : \mathbb{R}^n \to 2^{\mathcal{P}}$ associating to each state $x \in \mathbb{R}^n$ the set of atomic propositions that are satisfied at x, *i.e.*, $h(x) = \{P_i \in \mathcal{P} : x \in P_i\}$. These atomic propositions are used to construct safe-LTL formulas formalizing the desired specification to be enforced by the controller to be synthesized. For a definition of the syntax and semantics of LTL and safe-LTL we refer the readers to [17, 19]. Here, we simply mention that safe-LTL formulas always define safety specifications and we provide a concrete example illustrating its usefulness as a specification formalism.

Cruise control example.

Consider a truck with a trailer as depicted in Figure 1. We want to design a controller ensuring that the highway speed limits are always satisfied. The longitudinal dynamics of the truck and trailer are given by the continuous-time linear control system

$$\dot{x} = \begin{bmatrix} 0 & -1 & 1 \\ \frac{k_s}{m} & -\frac{k_d}{m} & \frac{k_d}{m} \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u,$$

where the entries of the state vector $x = (d, v_1, v_2) \in \mathbb{R}^3$ correspond to the distance d between the truck and the trailer, the velocity of the trailer v_2 and the velocity of the truck v_1, respectively. A more detailed explanation of this model is given in Section 6.1.

Let the truck be driving on a highway on which three speed limits $v_a = 15.6\,\mathrm{m/s}$ (ca. 35 mph), $v_b = 24.5\,\mathrm{m/s}$ (ca. 55 mph) and $v_c = 29.5\,\mathrm{m/s}$ (ca. 66 mph) are imposed. We would like to design a controller guaranteeing that the truck obeys the velocity limits. We assume that each velocity limit sign on the highway is equipped with a radio transmitter

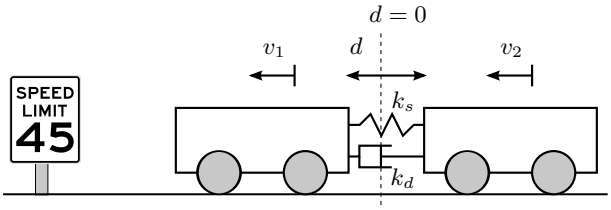

Figure 1: The truck with trailer.

that automatically transmits the limit to the truck. Then we would like to ensure that the truck obeys the current speed limit at most after $T \in \mathbb{N}$ time steps after the truck receives the information that the speed limit has changed.

For simplicity we consider only the limits v_a and v_b. A safe-LTL formula encoding the specification is given by:

$$\Box(\varphi_a \wedge \varphi_b) \tag{3}$$

where φ_a and φ_b are defined as:

$$m_a \implies \Diamond_{\leq T}(t_a W m_b) \quad \text{and} \quad m_b \implies \Diamond_{\leq T}(t_b W m_a)$$

respectively. The atomic proposition m_i, $i \in \{a, b\}$ encodes the fact that limit v_i is in place while the atomic proposition t_i encodes the satisfaction of the speed limit, that is, $v_1 \leq v_i$. The operator $\Diamond_{\leq T}$ requires $t_a W m_b$ to be satisfied in T or less steps while the formula $t_a W m_b$ requires t_a to be satisfied unless m_b becomes true, *i.e.*, the speed limit changes from m_a to m_b. We return to this example in Section 6.1 where we compute the maximal controlled invariant set.

3.3 The synthesis problem

There exists a well known automata theoretic approach to synthesize controllers with respect to LTL formulas and specifically, with respect to safe-LTL formulas. Safe-LTL formulas represent a fragment of full LTL formulas that specifies safety properties, while full LTL formulas can also specify liveness properties. Loosely speaking, a formula φ defines a safety property, if its violation can be checked by looking at a finite prefix of a trajectory. This property allows the use of finite automata in the synthesis of controllers enforcing safe-LTL instead of Büchi automata that require more complex algorithms, see *e.g.* [19]. Kupferman and Vardi [17] show that a *bad-prefix finite state automaton* can be constructed from a safe-LTL formula φ. This automaton accepts at least one bad prefix for every trajectory that violates the specification φ.

Let $A_{\neg\varphi} = (Q, Q_0, F, \delta, g, 2^{\mathcal{P}})$ denote the bad-prefix finite-state automaton with respect to the formula φ where Q is the set of states, $F \subseteq Q$ is the set of accepting states, $\delta \subset Q \times Q$ is the transition relation, $g : Q \to 2^{\mathcal{P}}$ is the output function and $2^{\mathcal{P}}$ is the output set. We define the transition system:

$$S_\varphi = (X, X_0, U, \delta_\varphi, Y, H) \tag{4}$$

as the synchronous product of system (1) and the bad-prefix automaton $A_{\neg\varphi}$ by:

- the set of states $X = \{(q, x) \in Q \times \mathbb{R}^n \mid g(q) = h(x)\}$;
- the initial states $X_0 = \{(q, x) \in Q_0 \times \mathbb{R}^n \mid g(q) = h(x)\}$;
- the set of inputs $U = \mathbb{R}^m$;

- the transition relation $\delta_\varphi \subseteq X \times X$;
- the set of outputs $Y = 2^{\mathcal{P}}$;
- the output map $H(q,x) = g(q)$ for each $(q,x) \in X$.

The transition relation δ_φ of the synchronous product is implicitly defined by $((q,x),(q',x')) \in \delta_\varphi$ iff there exists $u \in U$ such that

$$x' = Ax + Bu \wedge (q,q') \in \delta.$$

A *run* of the transition system S_φ associated to the initial state $(q,x) \in X_0$ is an infinite sequence $\zeta : \mathbb{N}_0 \to X$ with $\zeta(0) = (q,x)$ that satisfies $(\zeta(t),\zeta(t+1)) \in \delta_\varphi$ for all times $t \in \mathbb{N}_0$. Sometimes we will denote $\zeta(t)$ simply by ζ_t.

We refer the reader to [17, 19] for a detail description of the bad-prefix automaton and in particular to [13] where the very same bad-prefix automaton is used.

It is well-known that the controller enforcing φ on (1) corresponds to the controller forcing the system S_φ to stay within the *safe set*

$$K = \bigcup_{q \in Q \setminus F} \{q\} \times h^{-1} \circ g(q) \tag{5}$$

for all times. As long as trajectories remain in K, no finite-prefix of a trajectory violating the specification is reached and thus no violation of the specification ever occurs. Moreover, it is known that this controller is memoryless in the sense that it is given by a map $\mu : X \to 2^U$ and it can be chosen to be maximal, *i.e.*, any other controller $\mu' : X \to 2^U$ that enforces φ satisfies $\mu'(q,x) \subseteq \mu(q,x)$, see *e.g.* [31].

In summary, the synthesis of a controller enforcing a safe-LTL specification is reduced to the computation of a controlled invariant set for the system S_φ. Therefore, in the remaining parts of the paper, we solely focus on the computation of the maximal controlled invariant subset of K.

4. COMPUTATION OF INVARIANT SETS

Let us introduce the algorithm to compute the maximal controlled invariant subset (MCIS) of K defined by:

$$\mathcal{K}(K) = \big\{(q,x) \in X \,\big|\, \text{there exists a run } \zeta \text{ of } S_\varphi \text{ with } \zeta_0 = (q,x) \text{ and } \zeta_t \in K \text{ for all } t \in \mathbb{N}_0\big\}.$$

When no confusion arises we will denote the MCIS simply by $\mathcal{K}$. This set is computed by the well-known Algorithm 1, see *e.g.* [3]. The algorithm is initialized with the set K and proceeds iteratively by computing the set K_{i+1} as the intersection of K_i with the set of states that reach K_i in one step. Assuming that the algorithm terminates after $i \in \mathbb{N}_0$ steps, the output satisfies $K_i = \mathcal{K}(K)$, see *e.g.* [14, Lemma 2.1].

Algorithm 1 Computation of $\mathcal{K}(K)$

Input: K
Init: $K_0 := K$
  while $K_i \neq K_{i+1}$ **do**

$$K_{i+1} := K_i \cap \{z \in X \mid \exists z' \in K_i : (z,z') \in \delta_\varphi\} \tag{6}$$

  end while
Output: K_i

4.1 Approximation of the Maximal Controlled Invariant Set

For the following analysis, we assume the system (1) to be given in *special Brunovsky normal form*, see [7], *i.e.*, the system matrices are of form

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

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

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

The original Brunovsky normal form is similarly defined, only the last rows of the matrices A_{μ_i} contain possible non-zero entries. However, such rows can be cancelled out by a simple input transformation. Given that the system is controllable, all these transformations are carried out without loss of generality and automatically computed by our implementation. See [28] for a more detailed treatment.

In our approximation of the safe set K, we use the fact that for all $q \in Q$, the set $K(q)$ results from a finite union and intersection of atomic propositions $P_i \in \mathcal{P}$. Therefore, $K(q)$ is given as a finite union of polytopes

$$K(q) = \bigcup_{i=1}^{p_q} \mathcal{H}(C_i, c_i)$$

for some $C_i \in \mathbb{R}^{r_i \times n}$ and $c_i \in \mathbb{R}^{r_i}$. We now proceed by approximating these sets based on a uniform grid in $\mathbb{R}^n$ denoted by $[\mathbb{R}^n]_\rho$ and parameterized by $\rho \in \mathbb{R}_+$:

$$[\mathbb{R}^n]_\rho = \{x \in \mathbb{R}^n : x = \rho k, k \in \mathbb{Z}^n\}.$$

Let $\mathsf{C}_\rho(\xi)$ denote the *cell* associated to the grid point $\xi \in [\mathbb{R}^n]_\rho$:

$$\mathsf{C}_\rho(\xi) = \{x \in \mathbb{R}^n \mid \forall_{i \in \{1,\dots,n\}} : \xi_i \leq x_i \leq \xi_i + \rho\}.$$

and note that a cell is a Cartesian product of intervals and is thus adapted to the dynamics [31]. We define the over-approximation $\hat{K}$ and the under-approximation $\check{K}$ of the safe set K by

$$\begin{aligned} \hat{K}(q) &= \{x \in \mathbb{R}^n \mid \exists \xi \in [\mathbb{R}^n]_\rho : x \in \mathsf{C}_\rho(\xi) \cap K(q) \neq \varnothing\}, \\ \check{K}(q) &= \{x \in \mathbb{R}^n \mid \exists \xi \in [\mathbb{R}^n]_\rho : x \in \mathsf{C}_\rho(\xi) \subseteq K(q)\}. \end{aligned} \tag{7}$$

These approximations, being a union of adapted sets, are also adapted sets. It is straightforward to verify that $\check{K} \subseteq K \subseteq \hat{K}$ holds. Moreover, it follows immediately from the definition of MCIS that

$$\mathcal{K}(\check{K}) \subseteq \mathcal{K}(K) \subseteq \mathcal{K}(\hat{K}).$$

THEOREM 1. *Suppose system* (1) *is in special Brunovsky normal form and let* $\check{K}$ *and* $\hat{K}$ *be the sets as defined in* (7). *Then Algorithm 1, with input* $\check{K}$ *and* $\hat{K}$ *terminates in a finite number of iterations and the output* K_i *is the maximal controlled invariant subset of* $\check{K}$ *and* $\hat{K}$, *respectively.*

Although the proof of this theorem follows from the results in [33] we include it here for the sake of completeness. Before we prove finite termination of the algorithm, we need a technical result for the set

$$\mathsf{Pre}(D) = \{x \in \mathbb{R}^n \mid \exists u \in \mathbb{R}^m : Ax + Bu \in D\} \qquad (8)$$

and the following definition.

DEFINITION 1. *We call a set $D \subseteq \mathbb{R}^n$ finitely representable on $[\mathbb{R}^n]_\rho$ if there exist $\xi^1, \dots, \xi^p \in [\mathbb{R}^n]_\rho$ such that $D = \cup_i \mathsf{C}_\rho(\xi^i)$.*

LEMMA 1. *Suppose* (1) *is in special Brunovsky normal form. If D and D' are finitely representable sets then $D' \cap \mathsf{Pre}(D)$ is finitely representable.*

PROOF OF THEOREM 1. We show the assertion for the under-approximation $\mathcal{K}(\check{K})$, *i.e.*, Algorithm 1 is invoked with $\check{K}$. The statement for $\hat{K}$ follows analogously. We can write equation (6) as

$$K_{i+1} = K_i \cap \bigcup_{(q,q') \in \delta, q' \in \pi_Q(K_i)} \{q\} \times \mathsf{Pre}(K_i(q')).$$

Note that $K_0(q)$ is finitely representable for all $q \in Q$. Thus, we can use induction and invoke Lemma 1 to conclude that that the sets $K_i(q)$ are finitely representable for every $i \in \mathbb{N}_0$ and $q \in Q$.

Now we identify every $K_i(q)$ with its representing grid points $K_i(q) \cap [\mathbb{R}^n]_\rho$ and define the operator

$$G : 2^{\check{K}_\rho} \to 2^{\check{K}_\rho}, K_i \cap Q \times [\mathbb{R}^n]_\rho \mapsto K_{i+1} \cap Q \times [\mathbb{R}^n]_\rho$$

with $\check{K}_\rho = K \cap Q \times [\mathbb{R}^n]_\rho$ and K_i and K_{i+1} given in (6). Note that $(2^{\check{K}_\rho}, \subseteq)$ is a complete lattice and $2^{\check{K}_\rho}$ is finite. In addition, it is straightforward to see that G is monotone. Then, we apply Tarski's fixed point theorem, see *e.g.* [34] or [31, Corollary A.6], and conclude that there exists $i \in \mathbb{N}_0$ such that the maximal fixpoint $\mathcal{K} = G(\mathcal{K})$ of G is given by $\mathcal{K} = G^i(\check{K}_\rho)$, where G^i denotes the ith-fold composition of G with itself. □

4.2 Approximation Analysis

In this subsection we discuss how well we can approximate $\mathcal{K}(K)$ with $\mathcal{K}(\check{K})$. It is well-known that the MCIS can be arbitrarily well over-approximated [29]. However, this is in strong contrast to under-approximations. It is shown in [8] that replacing K with an arbitrarily small under-approximation may lead to an empty MCIS. Nevertheless, if K contains a controlled invariant set I "strictly inside", then by taking $\check{K}$ to be a sufficiently close approximation of K we can capture I in the sense that $I \subseteq \mathcal{K}(\check{K})$. In other words, if the synthesis problem has a solution for a safe set that is "strictly inside" K, we are guaranteed to find that solution. We now make these ideas precise.

DEFINITION 2. *A set $I \subseteq X$ is called controlled invariant with respect to S_φ if for every $z \in I$ there exists a run ζ of S_φ such that $\zeta_0 = z$ and $\zeta_t \in I$ for all $t \in \mathbb{N}_0$.*

We now formalize the meaning of "strictly inside" by requiring $\mathsf{d_H}(I, \partial K) \geq \gamma$ for some $\gamma > 0$ where $\mathsf{d_H}$ is the Hausdorff distance between the set I and the boundary of K denoted by ∂K. Whenever $\mathsf{d_H}(I, \partial K) \geq \gamma$ holds, we can construct an under-approximation $\check{K}$ of K containing I and thus we have the guarantee $I \subseteq \mathcal{K}(\check{K})$ by maximality of $\mathcal{K}(\check{K})$.

In the following, we use $\check{K}_\rho$ to indicate that the under-approximation $\check{K}$ is obtained with respect to the grid $[\mathbb{R}^n]_\rho$.

THEOREM 2. *Let $I \subseteq X$ be a controlled invariant set with respect to S_φ contained in $I \subseteq K \subseteq X$ satisfying $\mathsf{d_H}(I, \partial K) \geq \gamma$ for some $\gamma > 0$. Then, there exists $\rho \in \mathbb{R}_+$ such that $I \subseteq \mathcal{K}(\check{K}_\rho)$.*

PROOF. Suppose the assertion is not true. Then there exists a run ζ of S_φ with initial state $\zeta_0 \in I \backslash (\cup_{\rho \in \mathbb{R}_+} \mathcal{K}(\check{K}_\rho))$ with $\zeta_t \in I$ for all $t \in N_0$. Therefore, there exists $\gamma \in \mathbb{R}_+$ with $\mathsf{d_H}\big(\pi_{\mathbb{R}^n}(\zeta_t), \partial K\left(\pi_Q(\zeta_t)\right)\big) \geq \gamma$. Moreover, there exists $\rho_0 \in \mathbb{R}_+$ such that for all $\rho \in \mathbb{R}_+$, $\rho \leq \rho_0$ we have $\mathsf{d_H}(\check{K}_\rho, K) \leq \gamma/2$, which implies $\zeta_t \in \check{K}_\rho$ for all $t \in \mathbb{N}_0$ and therefore $\zeta_0 \in \mathcal{K}(\check{K}_\rho)$. A contradiction. □

4.3 Input Constraints

So far we have assumed that there are no constraints on the inputs of the linear system. However, it is often the case that the control inputs are restricted to a subset $V \subseteq U$ of the input space $U = \mathbb{R}^m$. If that is the case we can aim at computing an input constrained version of the MCIS. That is, in the definition of the transition relation δ_φ, see (4), we enforce the inputs to be elements in $U = V$ instead of $U = \mathbb{R}^m$. Let $\mathcal{K}_V(K)$ denote the MCIS with respect to the constrained input space V. We incorporate such input constraints by extending the state space of the system so as to incorporate the inputs. The system matrices of the extended system are given by

$$A' = \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}, \quad B' = \begin{bmatrix} 0 \\ I \end{bmatrix} \qquad (9)$$

where I is the identity matrix in $\mathbb{R}^m$. Each element P'_i of the set of atomic propositions $\mathcal{P}'$ is correspondingly modified to

$$P'_i = P_i \times V$$

which in turn results in a modified safe set K' according to (5). The following lemma ensures the completeness and soundness of this scheme.

LEMMA 2. *Let S_φ and S'_φ be the transition systems obtained as the synchronous product of* (1) *and* (9) *with the bad-prefix automaton, respectively, and let K and K' denote the corresponding safe sets. Then, the input constrained MCIS of S_φ, denoted by $\mathcal{K}_V(K)$, coincides with the MCIS of S'_φ, denoted by $\mathcal{K}'(K')$, on X:*

$$\mathcal{K}_V(K) = \pi_X(\mathcal{K}'(K')),$$

where π_X is the projection from $X \times V$ to X.

PROOF. Let $(q, x) \in \mathcal{K}_V(K)$. This implies, that there exists a run ζ of S_φ with initial state (q, x) such that $\zeta_t \in K$ for all $t \geq 0$. Let $\upsilon : \mathbb{N}_0 \to V$ denote the associated input sequence. We define the sequence $\upsilon' : \mathbb{N}_0 \to V$ by a shift operation $\upsilon'_t := \upsilon_{t+1}$. By the definition of the extended system (9) we are able to chose a run ζ' of S'_φ with initial state $\zeta'_0 = (q, x, \upsilon_0)$ that satisfies

$$\zeta'_t = (\zeta_t, \upsilon_{t-1}).$$

As S_φ obeys the constraints $\zeta_t \in X$ and $\upsilon_t \in V$ it follows that $\zeta'_t \in X \times V$ for all $t \geq 0$ which implies that $(q, x, \upsilon_0) \in \check{\mathcal{K}}'$ and thus $\mathcal{K}_V \subset \pi_X(\mathcal{K}')$.

For the reverse inclusion, let $(q', x', u') \in \mathcal{K}'$. This implies that there exists a run ζ' of S'_φ with initial sate (q', x', u') that satisfies $\zeta'_t \in X \times V$ for all $t \geq 0$. Let π_V be the projection from $X \times V$ to V and define the sequence $\upsilon_t := \pi_V(\zeta'_t)$. We see, again by the definition of the extended

system (9), that we can pick a run ζ of S_φ with initial state (q', x') that satisfies $(\zeta_t, \upsilon_t) = \zeta'_t$, which implies that $(q, x) \in \mathcal{K}_V$. $\square$

5. SYMBOLIC IMPLEMENTATION

In this section, we describe the implementation of our approach using binary decision diagrams (BDDs) and algebraic decision diagrams (ADDs). Decision diagrams are data structures that efficiently represent binary functions $f_b : \mathbb{B}^n \to \mathbb{B}$ with binary codomain ($\mathbb{B} = \{0, 1\}$) and binary functions $f_a : \mathbb{B}^n \to \mathbb{R}$ with real codomain. In particular, we focus on the two operations that are unique to our approach: the under-approximation of the atomic propositions and the computation of the set iterates (6). In what follows, we show how these two operations can be performed in a symbolic manner, *i.e.*, without iterating over the grid points. Moreover, all the operations are implemented using ordinary BDD manipulations like conjunction, disjunction, variable reordering and quantifier elimination as well as ADD manipulations like thresholding, addition and subtraction. Standard BDD packages as CUDD [30] provide efficient implementations for all of those operations.

5.1 Set Encoding

Our implementation of Algorithm 1 starts with the encoding of the safe set K, see (5), in terms of BDDs on a binary domain $\mathbb{B}^{Nn}$. In doing so, we assume that the sets $K(q) \subseteq \mathbb{R}^n$ are contained in the unit-cube $\mathsf{B}(0, 1)$. If that is not the case, we scale the sets and the continuous dynamics accordingly.

Once the grid $[\mathbb{R}^n]_\rho$ is fixed, the dimension Nn of the binary domain $\mathbb{B}^{Nn}$ corresponds to the product of the state space dimension $n \in \mathbb{N}$ and the number of bits $N \in \mathbb{N}$ that are required to binary encode the finite number of grid points of $[\mathsf{B}(0, 1)]_\rho = \mathsf{B}(0, 1) \cap [\mathbb{R}^n]_\rho$.

Given a grid point $\xi = (\xi_1, \dots, \xi_n) \in [\mathsf{B}(0, 1)]_\rho$ we denote its binary encoding by $\xi_\mathbb{B} = (\xi_{1_\mathbb{B}}, \dots, \xi_{n_\mathbb{B}}) \in \mathbb{B}^{Nn}$ with each $\xi_{i_\mathbb{B}} \in \mathbb{B}^N$.

5.2 Symbolic Approximation of Polytopes

Every polytope is obtained as a finite intersection of half-spaces. Henceforth, we focus on the approximation of half-spaces of the form

$$R = \{x \in \mathbb{R}^n \mid Cx \leq c\}$$

with $C \in \mathbb{R}^{1 \times n}$ and $c \in \mathbb{R}$. Once we have the BDD representation of the half-spaces, computing the intersection can be realized by computing the conjunction of the two BDDs representing these sets.

The under-approximation and the over-approximation of the set R is represented by the BDDs $\check{r} : \mathbb{B}^{Nn} \to \mathbb{B}$ and $\hat{r} : \mathbb{B}^{Nn} \to \mathbb{B}$, respectively, defined as:

$$\check{r}(\xi_\mathbb{B}) = 1 \quad \Leftrightarrow \quad \mathsf{C}_\rho(\xi) \subseteq R, \tag{10}$$

$$\hat{r}(\xi_\mathbb{B}) = 1 \quad \Leftrightarrow \quad \mathsf{C}_\rho(\xi) \cap R \neq \varnothing. \tag{11}$$

Rather than using the definition of $\check{r}$ and $\hat{r}$ we compute these BDDs by making use of the ADDs $\check{f} : \mathbb{B}^{Nn} \to \mathbb{R}$ and $\hat{f} : \mathbb{B}^{Nn} \to \mathbb{R}$ given by

$$\check{f}(\xi_\mathbb{B}) = \max_{x \in \mathsf{C}_\rho(\xi)} Cx \quad \text{and} \quad \hat{f}(\xi_\mathbb{B}) = \min_{x \in \mathsf{C}_\rho(\xi)} Cx.$$

The functions $\check{f}$ and $\hat{f}$ associate to each grid point ξ the maximum and minimum value of Cx with x restricted to $x \in \mathsf{C}_\rho(\xi)$. Notice that we have the following implications

$$\check{f}(\xi_\mathbb{B}) \leq c \Leftrightarrow \mathsf{C}_\rho(\xi) \subseteq R \quad \text{and} \quad \hat{f}(\xi_\mathbb{B}) \leq c \Leftrightarrow \mathsf{C}_\rho(\xi) \cap R \neq \varnothing.$$

Therefore, we are able to define the approximating BDDs $\check{r}$ and $\hat{r}$ simply by truncating the ADDs $\check{f}$ and $\hat{f}$. In particular, we obtain all grid points $\xi_\mathbb{B} \in \mathbb{B}^{Nn}$ that satisfy $\check{r}(\xi_\mathbb{B}) = 1$ respectively $\hat{r}(\xi_\mathbb{B}) = 1$ by

$$\check{r}^{-1}(1) = \{\xi_\mathbb{B} \in \mathbb{B}^{Nn} \mid \check{f}(\xi_\mathbb{B}) \leq c\}$$
$$\hat{r}^{-1}(1) = \{\xi_\mathbb{B} \in \mathbb{B}^{Nn} \mid \hat{f}(\xi_\mathbb{B}) \leq c\}.$$

The ADDs $\check{f}$ and $\hat{f}$ can in turn be efficiently computed as we outline in the reminder of this subsection.

To provide a clearer presentation, we consider the one-dimensional case $n = 1$ and assume that $C \geq 0$. We fix the particular encoding scheme as follows: a grid point $\xi \in [B(0, 1)]_\rho$ is obtained by a binary element $\xi_\mathbb{B} = b_0 \dots b_{N-1}$ of $\mathbb{B}^N$ by

$$\xi = \sum_{i=0}^{N-1} 2^{-i} b_i - 1.$$

With this encoding scheme in mind, we compute $\check{f}$ iteratively over the number of bits by the functions $\check{f}^i : \mathbb{B}^i \to \mathbb{R}$ obtained from

$$\check{f}^0(b_0) = Cb_0$$
$$\check{f}^{i+1}(b_0 \dots b_{i+1}) = \check{f}^i(b_0 \dots b_i) - 2^{-i} C(1 - b_{i+1}).$$

It is easy to verify that $\check{f} = \check{f}^{N-1}$. We proceed in an analogous way for the general case, as well as to compute $\hat{f}$.

5.3 Implementation of the Set Iterates

In this subsection, we describe the implementation of the main computation in Algorithm 1, *i.e.*, the iteration (6) given by

$$K \cap K'$$

with $K' = \{z \in X \mid \exists z' \in K : (z, z') \in \delta_\varphi\}$. In particular we focus on how to obtain a BDD representation of K' given a BDD representation of K. The set K' can be written in terms of

$$K' = \{(q, x) \in X \mid \exists q' \in \pi_Q(K) \wedge (q, q') \in \delta \wedge x \in \mathsf{Pre}(K(q))\}.$$

with the Pre operator as defined in (8). Notice that the elements in $(q, x) \in K'$ do not need to be output synchronized, *i.e.*, we do not require $g(q) = h(x)$. The synchronization of the elements in K' is achieved by the intersection of K' with the synchronized elements K in (6).

In the following we describe, how we obtain K' in two steps. First, we compute the set $\tilde{K}$ from K by

$$\tilde{K} = \{(q, x) \in X \mid q \in \pi_Q(K) \wedge x \in \mathsf{Pre}(K(q))\}$$

and subsequently K' from $\tilde{K}$ by

$$K' = \{(q, x) \in X \mid \exists q' \in \pi_Q(\tilde{K}) : (q, q') \in \delta \wedge x \in \tilde{K}(q')\}.$$

It is not difficult to see that the computation of K' via $\tilde{K}$ can be obtained through standard BDD operations like existential abstraction and conjunction.

In the remainder of this subsection, we focus on the computation of $\tilde{K}$ from K, which basically corresponds to the implementation of the Pre operator. For the sake of presentation, let us focus on the single input case.

Let $k : \mathbb{B}^{|Q|Nn} \to \mathbb{B}$ denote the BDD representation of the set K given by

$$k(q_{\mathbb{B}}, \xi_{\mathbb{B}}) = 1 \Leftrightarrow \{q\} \times \mathsf{C}_{\rho}(\xi) \subseteq K.$$

Notice that we assume that all sets $K(q)$ are obtained from the above approximation procedure. Hence, the sets $K(q)$ are finitely representable and therefore, k is an exact representation of K.

As detailed in [28], whenever the linear system is given in Brunovsky normal form, the Pre computation is implemented by the following simple existential abstraction and variable shift:

$$\tilde{k}(q_{\mathbb{B}}, \xi_{\mathbb{B}_1}, \ldots, \xi_{\mathbb{B}_n}) = 1 \Leftrightarrow \exists u_{\mathbb{B}} : k(q_{\mathbb{B}}, \xi_{\mathbb{B}_2}, \ldots, \xi_{\mathbb{B}_{n-1}}, u_{\mathbb{B}}) = 1.$$

We refer the reader to [28] for a more detailed explanation and the multi-dimensional input case.

Notice that all our computations, including the approximation of polytopes, are performed symbolically, *i.e.*, we avoid any iteration over the grid points $\xi_{\mathbb{B}}$. Moreover, none of the operations in our implementation includes the costly-to-obtain abstraction, *i.e.*, a BDD representation of the transition relation $\delta_{\varphi} \subseteq X \times X$, which is the case for several known approaches, *e.g.* [21, 10, 27, 35, 40] just to mention a few. In the proposed scheme, the computation of the abstraction is restricted to the safe set K. As a result, we are able to save memory as well as computation time.

6. EXPERIMENTAL RESULTS

In this section, we demonstrate the performance of the developed algorithm for some examples. In the first subsection, we synthesize a cruise controller for a truck on a highway. In the second part, we compare the performance of Algorithm 1 with the polyhedral approach, described in [14] and [26] in terms of an example from [24].

The accuracy of the computed solutions will be estimated by the upper bound $\hat{e}$ of the relative error between the volume of the MCIS and the volume of the computed under-approximation:

$$\hat{e} = \frac{\mathsf{vol}\,\mathcal{K}(\hat{K}) - \mathsf{vol}\,\mathcal{K}(\check{K})}{\mathsf{vol}\,\mathcal{K}(\check{K})} \geq \frac{\mathsf{vol}\,\mathcal{K}(K) - \mathsf{vol}\,\mathcal{K}(\check{K})}{\mathsf{vol}\,\mathcal{K}(K)}.$$

We implemented all the algorithms in C using the decision diagram library from the University of Colorado (CUDD) [30]. All computations are performed on an Intel Core i7 1.8GHz processor using 4GBytes of memory.

6.1 Cruise Controller for a Truck

We return to the example that was briefly described in Section 3.

Recall that we seek to design a controller for the truck with a trailer depicted in Figure 1. The objective of the controller is to enforce the highway speed limits. We start with the continuous-time linear model:

$$\dot{x} = \begin{bmatrix} 0 & -1 & 1 \\ \frac{k_s}{m} & -\frac{k_d}{m} & \frac{k_d}{m} \\ 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u,$$

where the entries of the state vector $x = (d, v_2, v_1) \in \mathbb{R}^3$ correspond to the distance d between the truck and the trailer, the velocity of the trailer v_2 and the velocity of the truck v_1, respectively. Here, we model the connection between the truck and the trailer by a spring-damper system with constants $k_s = 4500\,\mathrm{N/kg}$ and $k_d = 4600\,\mathrm{Ns/m}$. The mass of the trailer is fixed to $m = 1000\,\mathrm{kg}$. The input of our model is the constrained acceleration $u \in [-4, 4]\,\mathrm{m/s^2}$ of the truck.

We consider three speed limits $v_a = 15.6\,\mathrm{m/s}$ (ca. 35 mph), $v_b = 24.5\,\mathrm{m/s}$ (ca. 55 mph) and $v_c = 29.5\,\mathrm{m/s}$ (ca. 66 mph) and would like to design a controller, that guarantees that the truck obeys the effective velocity constraint. In particular, we would like to ensure that the truck obeys the current speed limit at most $T \in \mathbb{N}$ time steps after the controller receives the information that the speed limit has changed. The number of steps T, after which we enforce the speed limit is a parameter that we vary throughout the experiments.

Truck without trailer. We provide a complete description of a simplified version of this problem. However, we present the experimental results for the original example. The simplified version consists of the truck without a trailer subject only to two speed limits v_a and v_b. As a first step, we discretize the continuous-time system by sampling the solution of the system under piecewise constant inputs, with sampling rate $h = 0.4\,\mathrm{s}$. In the second step, we extend the state space with the input space so that we are able to account for the input constraints $u \in [-4, 4]$. The resulting system is given by

$$z^+ = \begin{bmatrix} 1 & h \\ 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0 \\ h \end{bmatrix} v \tag{12}$$

with $z = (v_1, u)$ and unconstrained input v. In addition to the model of the truck we introduce an automaton as a model for the environment, *i.e.*, a model for the change of speed limits over time. The automaton is sketched in Figure 2. It has two states, m_a and m_b, one for each speed limit and it can arbitrarily change between these states. The full plant model is given by the composition of the linear system (12) and the automaton in Figure 2.

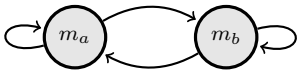

Figure 2: Model of the highway.

Notice that so far we assumed that our plant is given by a linear system and not as a switched system with linear dynamics. However, as we will describe below, for this particular example, it is an easy task to modify Algorithm 1 accordingly.

We define the atomic propositions by $P_0 = \mathcal{H}(C_0, c_0)$, $P_1 = \mathcal{H}(C_1, c_1)$ and $P_2 = \mathcal{H}(C_1, c_2)$ with

$$C_0 = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}, \quad c_0 = \begin{bmatrix} 0 \\ 35 \\ 4 \\ 4 \end{bmatrix}$$

and $C_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}$, $c_1 = 15.6\,\mathrm{m/s}$, $c_2 = 24.5\,\mathrm{m/s}$. We use P_0 to specify our operating speed range $5 \leq v_1 \leq 35\,\mathrm{m/s}$ as well as to account for the input constraints $-4 \leq u \leq 4\,\mathrm{m/s^2}$. The propositions P_1 and P_2 are used to enforce the speed limits v_a and v_b, respectively.

The specification is given by the safe-LTL formula (3) and the corresponding bad-prefix finite state automaton for $T = 2$ is depicted in Figure 3.

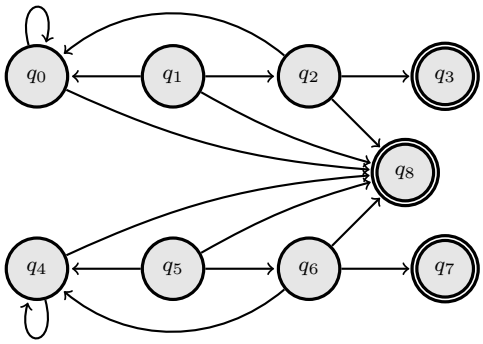

Figure 3: The bad-prefix finite state automaton accepting the prefixes of the undesired behavior.

The output function $g : Q \to Y$ is given by

$$\begin{aligned} g(q_0) &= \{m_a, P_0, P_1\}, & g(q_1) &= \{m_a, P_0\}, \\ g(q_2) &= \{m_a, P_0\}, & g(q_3) &= \{m_a, P_0\}, \\ g(q_4) &= \{m_b, P_0, P_2\}, & g(q_5) &= \{m_b, P_0\}, \\ g(q_6) &= \{m_b, P_0\}, & g(q_7) &= \{m_b, P_0\}, \\ g(q_8) &= \varnothing. \end{aligned}$$

The states q_0 and q_4 of the bad-prefix automaton represent the states where the speed limits v_a and v_b, respectively are met by the truck. For the states q_1 and q_5 the truck is allowed to violate the speed limit, but has to reach q_0 or q_4 in two steps. The same holds for q_2 and q_6 only that the truck needs to meet the speed limits in one step. Otherwise, the system ends up in one of the marked states, which represent the undesired behavior.

We now form the synchronous composition of the bad prefix finite state automaton with the plant. Recall that the plant is in fact the synchronous composition of the continuous dynamics in (12) with the automaton in Figure 2 that has $M = \{m_a, m_b\}$ as set of states. The set that we want to render invariant is given by $K_0 = Q_0 \times M \times \mathbb{R}^2$ with $Q_0 = \{q_0, q_1, q_2, q_4, q_5, q_6\}$. Note that we do not have control over the change of the speed limit. Hence, we need to compute a MCIS that is robust with respect to uncontrollable changes in speed limits. This is achieved by replacing the sets K_i in each iteration just before the computation of (6) in Algorithm 1 with

$$K_i' = \{(q, m, x) \in K_i \mid (q, m_a, x) \in K_i \wedge (q, m_b, x) \in K_i\}.$$

This replacement ensures that the elements in $\mathcal{K}$ are independent of the mode $m \in M$. A more general approach to handle the action of the environment is to consider a game whose solution would require replacing Algorithm 1 with a version that is robust with respect to environment actions.

We approximately computed the MCIS $\mathcal{K}$ where we used 10 bits in each dimension to approximate the atomic propositions. The continuous part of $\mathcal{K}$, associated to the states q_0 and q_4 is illustrated in the left subplots of Figure 4.

The cutoff corners of the sets clearly indicate the integrator dynamics of the truck. Notice that, even though for q_4 the truck is allowed to drive up to $v_b = 24.5\,\mathrm{m/s}$, it stays close to $v_a = 15.6\,\mathrm{m/s}$. That results from the fact, that the truck needs to be able to reduce its velocity to v_b within $Th = 0.8\,\mathrm{s}$ at all times. We show the approximation of the MCIS $\check{\mathcal{K}}$ for the case $T = 10$ in the right subplots of Figure 4. As expected, the truck is now allowed to drive at a much higher speed in mode m_b compared to the case $T = 2$. The run-time t_r of the algorithm to compute $\check{\mathcal{K}}$ for both cases is below a second and the error is bounded by $\hat{e} \leq 0.2$.

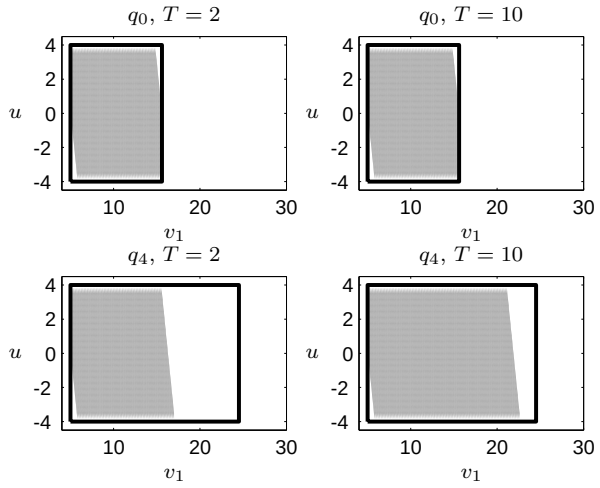

Figure 4: The set $\check{\mathcal{K}}$ associated to q_0 and q_4 using $N = 10$ bits per dimension for $T = 2$ and $T = 10$.

The run-times of Algorithm 1 and error estimates for various parameters $T \in \{2, 10\}$ and $N \in \{10, 11, 12\}$ are listed in Table 1. We conducted the computations with all three speed limits, which results in a maximum number of 33 discrete states on the bad-prefix automaton.

N\T	2		10	
	t_r	$\hat{e}$	t_r	$\hat{e}$
10	0.2s	0.24	0.2s	0.27
11	0.3s	0.12	0.3s	0.13
12	0.3s	0.06	0.3s	0.06

Table 1: Run-times of Alg. 1 and error bounds $\hat{e}$.

Truck with trailer. Now we consider the truck with trailer. Following the same steps as above, we sample the solution of the continuous-time system under piecewise constant inputs, in order to obtain a discrete-time system. Additionally, we augment the state space with the input space which results in the four-dimensional state vector

$$z = [d,\, v_2,\, v_1,\, u]^\mathsf{T}.$$

We use the same bad-prefix automaton to specify the desired system behavior. For this case, however, in addition to the speed limits we constrain the distance between the two trucks to the interval

$$-1/2\,\mathrm{m} \leq d \leq 1/2\,\mathrm{m}.$$

Thus, the polytopes associated to the atomic propositions $P_0 = \mathcal{H}(C_0, c_0)$, $P_1 = \mathcal{H}(C_1, c_1)$ and $P_2 = \mathcal{H}(C_1, c_2)$ are

given by

$$C_0 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad c_0 = \begin{bmatrix} 1/2 \\ 1/2 \\ 5 \\ 35 \\ 5 \\ 35 \\ 4 \\ 4 \end{bmatrix}$$

and $C_1 = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$, $c_1 = 15.6\,\mathrm{m/s}$, $c_2 = 24.5\,\mathrm{m/s}$.

The times required to compute the MCISs are listed in Table 2. Compared to the previous case, we can clearly observe the burden of the higher dimensional state space in terms of the run-times and tightness of the approximation. Also, for this example the number of discrete states is observable in the computation times, *i.e.*, as we increase $T = 2$ to $T = 10$ the bad prefix automaton becomes larger and so do the computation times.

N\T	2		10	
	t_r	$\hat{e}$	t_r	$\hat{e}$
10	1m39s	2.31	2m40s	2.38
11	4m09s	1.01	4m31s	1.04
12	6m48s	0.58	7m52s	0.62
13	10m38s	0.43	16m01s	0.46

Table 2: Run-times of Alg. 1 and error bounds $\hat{e}$.

6.2 Comparison with an Example from the Literature

In this subsection, we show some evidence of the previously mentioned claim, that the combinatorial complexity within the known polyhedral approach [14, 26] constitutes a serious problem when applied to controller synthesis for safe-LTL specifications.

For this purpose, we consider Example 5.1 introduced in [24]. This example describes a linear system with a constrained three dimensional state space and a constrained two dimensional input space. The set of atomic propositions is formulated in terms of subsets of the state space $Y = \mathsf{B}(\mathbf{0}, \mathbf{30})$, $O_1 = \mathsf{B}(\mathbf{10}, \mathbf{5})$, $O_2 = \mathsf{B}(-\mathbf{5}, \mathbf{5})$ and $O_3 = \mathsf{B}(-\mathbf{15}, \mathbf{10})$ together with subsets of the input space $V = \mathsf{B}(\mathbf{0}, \mathbf{2})$, $W_1 = \mathsf{B}(-\mathbf{3/2}, \mathbf{1/2})$, $W_2 = \mathsf{B}(-\mathbf{1/4}, \mathbf{1/4})$ and $W_3 = \mathsf{B}(\mathbf{2/5}, \mathbf{1/5})$. Here we use the bold notation, to indicate that the number corresponds to a vector in the appropriate dimension, e.g., for $Y = \mathsf{B}(\mathbf{0}, \mathbf{30})$ we have $\mathbf{0} = [0, 0, 0]^\mathsf{T}$ and $\mathbf{30} = [30, 30, 30]^\mathsf{T}$.

We interpret the sets Y and V as the domain of the problem, while the sets O_i and W_i represent obstacles, in the state space and input space, respectively. We computed the MCIS $\mathcal{K}$ for the following specifications of increasing complexity

$$\begin{aligned}
\varphi_0 &= \Box(Y \times V) \\
\varphi_1 &= \Box((Y \wedge \neg O_1) \times V) \\
\varphi_2 &= \Box(Y \times (V \wedge \neg W_1)) \\
\varphi_3 &= \Box((Y \wedge \neg O_1) \times (V \wedge \neg W_1)) \\
\varphi_4 &= \Box((Y \wedge_{i=1}^{2} \neg O_i) \times (V \wedge_{i=1}^{2} \neg W_i)) \\
\varphi_5 &= \Box((Y \wedge_{i=1}^{3} \neg O_i) \times (V \wedge_{i=1}^{2} \neg W_i)) \\
\varphi_6 &= \Box((Y \wedge_{i=1}^{3} \neg O_i) \times (V \wedge_{i=1}^{3} \neg W_i))
\end{aligned}$$

In all of the conducted computations the polyhedral algorithm terminated. We show the run-times of the computations in comparison with our symbolic implementation of Algorithm 1 in Figure 5. The subplot inside the main plot shows the outcome for $\varphi_0, \ldots, \varphi_4$. Clearly, for simple specifications the polyhedral approach outperforms the symbolic scheme. This simply results from the fact that the safe set first has to be approximated, which takes more than ninety percent of the time. However, as the problem becomes more complex the approximation effort lessens in comparison with the effort to compute the set iterates. For example, for the problem with two obstacles in the state space and input space, *i.e.*, φ_4, we can observe that the proposed symbolic approach starts to outperform the polyhedral approach.

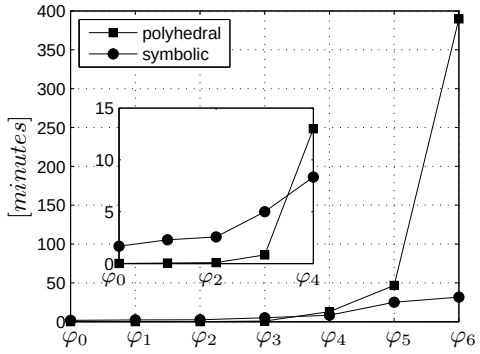

Figure 5: Comparison of the polyhedral approach with the symbolic scheme in terms of run-times.

We implemented the polyhedral approach using MATLAB and the multi-parametric toolbox [18]. The toolbox provides several methods to compute intersections and projections of polytopes. Specifically, we used the mex interface to the "fast" C implementation of the Fourier-Motzkin quantifier elimination to compute polyhedral projections, see `help polytope/projection`.

7. DISCUSSION

In this paper we presented an approach to the synthesis of controllers enforcing safe-LTL specifications on discrete-time linear systems. It was shown through examples that it scales up to 5 continuous variables thus placing it at the forefront of the existing controller synthesis techniques enforcing temporal logic specifications on continuous systems. We are currently working to increase the size of the systems that can be handled by improving the approximation of sets which constitutes the bottleneck of the current implementation.

8. REFERENCES

[1] A. Abate, J. P. Katoen, J. Lygeros, and M. Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, 16:624, 2010.

[2] J. P. Aubin. *Viability Theory*. Systems & Control: Foundations & Applications. Birkhäuser, 1991.

[3] D. Bertsekas and I. B. Rhodes. On the minimax reachability of target sets and target tubes. *Automatica*, 7:233–247, 1971.

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

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

[6] O. Bournez, O. Maler, and A. Pnueli. Orthogonal polyhedra: Representation and computation. In *HSCC*, LNCS, pages 46–60. Springer, 1999.

[7] P. Brunovský. A classification of linear controllable systems. *Kybernetika*, 6:173–188, 1970.

[8] P. Collins. Optimal semicomputable approximations to reachable and invariant sets. *Theory of Computing Systems*, 41:33–48, 2007.

[9] E. De Santis, M. D. Di Benedetto, and L. Berardi. Computation of maximal safe sets for switching systems. *IEEE TAC*, 49:184–195, 2004.

[10] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45:343–352, 2009.

[11] C. Finucane, G. Jing, and H. Kress-Gazit. LTLMoP website. `http://ltlmop.github.com/`, 2010.

[12] A. Girard and G. J. Pappas. Hierarchical control system design using approximate simulation. *Automatica*, 45:566–571, 2009.

[13] E. A. Gol, M. Lazar, and C. Belta. Language-guided controller synthesis for discrete-time linear systems. In *HSCC*, pages 95–104. ACM, 2012.

[14] E. C. Kerrigan. *Robust Constraint Satisfaction: Invariant Sets and Predictive Control*. PhD thesis, Dep. of Eng., University of Cambridge, 2000.

[15] M. Kloetzer and C. Belta. LTL-Con website. `http://iasi.bu.edu/Software.html`, 2006.

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

[17] O. Kupferman and M. Y. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19:291–314, 2001.

[18] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi-parametric toolbox (mpt). In *HSCC*, volume 2993, pages 121–124. Springer, 2004.

[19] T. Latvala. Efficient model checking of safety properties. In *In Model Checking Software. 10th International SPIN Workshop*, pages 74–88, 2003.

[20] R. Majumdar and M. Zamani. Approximately bisimilar symbolic models for digital control systems. In *CAV*, volume 7358, pages 362–377. Springer, 2012.

[21] M. Mazo Jr., A. Davitian, and P. Tabuada. Pessoa website.. `http://www.cyphylab.ee.ucla.edu/pessoa`, 2009.

[22] M. Mazo Jr., A. Davitian, and P. Tabuada. Pessoa: A tool for embedded controller synthesis. In *CAV*, volume 6174 of *LNCS*, pages 566–569. Springer, 2010.

[23] T. Moor, J. Raisch, and S. O'Young. Discrete supervisory control of hybrid systems based on l-complete approximations. *Discrete Event Dynamic Systems*, 12:83–107, 2002.

[24] 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.

[25] A. Pnueli. The temporal logic of programs. In *Proc. of 18th Annual Symp. on Foundations of Computer Science*, pages 46–57, 1977.

[26] S. V. Rakovic, P. Grieder, M. Kvasnica, D. Q. Mayne, and M. Morari. Computation of invariant sets for piecewise affine discrete time systems subject to bounded disturbances. In *Proc. of the 43rd IEEE CDC*, pages 1418–1423, 2004.

[27] G. Reißig. Computing abstractions of nonlinear systems. *IEEE TAC*, 56:2583–2598, 2011.

[28] M. Rungger, M. Mazo Jr., and P. Tabuada. Scaling up controller synthesis for linear systems and safety specifications. In *Proc. of the 51th IEEE CDC*, 2012.

[29] P. Saint-Pierre. Approximation of the viability kernel. *Applied Math & Optimization*, 29:187–209, 1994.

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

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

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

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

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

[35] Y. Tazaki and J. Imura. Discrete abstractions of nonlinear systems based on error propagation analysis. *IEEE TAC*, 57:550–564, 2012.

[36] I. Tkachev and A. Abate. On infinite-horizon probabilistic properties and stochastic bisimulation functions. In *Proc. of the 50th IEEE CDC and ECC*, pages 526–531, 2011.

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

[38] 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 CDC*, pages 1243–1248, 2001.

[39] I. Wegener. *Branching Programs and Binary Decision Diagrams - Theory and Applications*. SIAM Monographs on Discrete Mathematics and Applications, 2000.

[40] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding horizon temporal logic planning. *IEEE TAC*, 57:2817–2830, 2012.

[41] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R.M. Murray. TuLiP website. `http://sourceforge.net/apps/mediawiki/tulip-control/`, 2010.

[42] B. Yordanov, J. Tůmová, I. Černá, J. Barnat, and C. Belta. Formal analysis of piecewise affine systems through formula-guided refinement. *Automatica*, 2012.