Unifying Control and Verification of Cyber-Physical Systems (UnCoVerCPS)

WP3 - Online Verification for Control

Deliverable 3.3: Report on Compositional Verification and Incremental Verification in Interaction with Online Controller Adaptation

---

WP3	**D3.3 – Report on compositional verification and incremental verification in interaction with online controller adaptation**
Authors	Maria Prandini - PoliMi Olaf Stursberg, Zonglin Liu - UKS Goran Frehse - UGA Matthias Althoff - TUM Alexander Rausch, Jens Oehlerking - Bosch
Short Description	This deliverable described the achievements of the UnCoVerCPS project in two directions: compositional verification of cyber-physical systems, where a divide-and-conquer approach is adopted so as to verify smaller subsystems instead of applying a monolithic verification approach; and incremental online verification for controller adaptation, where constraints for control tuning are computed online, which calls for incremental computational methods and the study of the impact of time-varying constraints on the controller properties.
Deliverable Type	Report
Dissemination level	Public
Delivery Date	30 Dec 2018
Contributions by	PoliMi, UKS, UGA, TUM, Bosch
Internal review by	Goran Frehse, Geoff Pegman, Xavier Fornari
External review by	
Internally accepted by	Matthias Althoff
Date of acceptance	

Document history:

Version	Date	Author	Description
1.0	15 October 2018	Prandini et al.	Draft version
2.0	26 October 2018	Prandini et al.	Internal review version
3.0	20 November 2018	Prandini et al.	Revised after internal review
4.0	8 December 2018	Prandini et al.	Final revision

Contents

1 Introduction

This deliverable describes the achievements of the UnCoVerCPS project under tasks 3.3 and 3.4 of work package 3, investigating new methods to reduce significantly the computation time for formal verification of continuous and hybrid systems to the purpose of verifying if planned actions are safe during the operation of the system. Specifically, two research directions have been pursued, that is: compositional verification and incremental verification for online controller tuning.

The first research direction is a further enhancement of the achievements on verification in work package 3 of this project. More precisely, the new methods for reachability computations documented in Deliverable 3.2 could be integrated in the proposed compositional verification schemes, and the results on compositional reachability analysis in Deliverable 3.1 are extended here to further subsystems interconnections. Compositional verification can also be instrumental to the design of networked predictive control strategies as the one described in Deliverable 2.2, where the problem is naturally formulated compositionally.

The second research direction is explicitly connected to the control part in work package 2 and, in particular, to the development of task 2.4 documented in Deliverable 2.3 where methods for interleaving reachability computations with controller optimization are presented.

Figure 1 provides a schematic view of the project structure and the interconnections of tasks 3.3 and 3.4 with the other tasks.

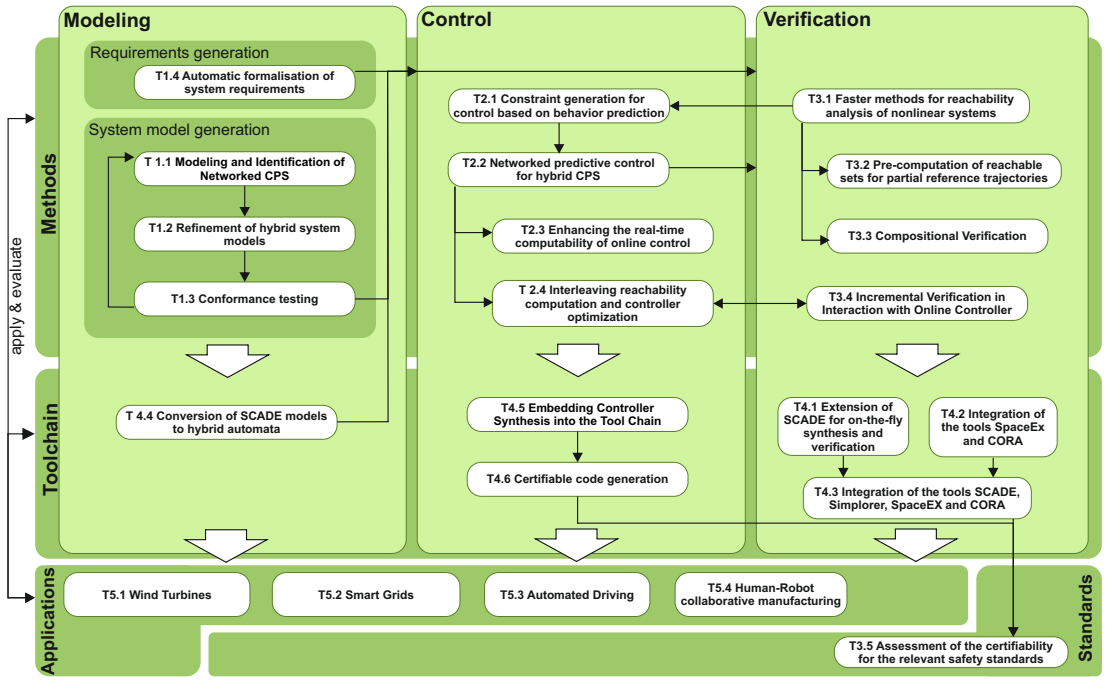


Figure 1: Structure and information flow of the UnCoVerCPS work packages and tasks.

As for compositional verification, the idea is to cope with the curse of dimensionality by a divide-and-conquer approach and verify smaller subsystems instead of applying a monolithic verification approach. One can exploit the specific interconnected structure of the system or decompose it appropriately in parts that have little interaction. This is addressed for the class of linear systems (Section 2.1) and mixed logical dynamical (MLD) systems (Section 2.2), which are equivalent to piecewise affine systems under well-posedness conditions. A parallel decomposition approach can be adopted for the verification of multiple MLD systems which are coupled via some budget constraint due to resource sharing (Section 2.2.2).

As for incremental verification for online controller tuning, the goal is twofold:

- use coarse model abstractions to provide rapid although conservative verification results that can be used to formally verify safety in the online operation of a system such as in autonomous driving applications. The coarse model abstractions are refined continually if computation time is available so as to allow for improving the system performance (see Section 3.1);

- provide guarantees on the performance of the controller that is adapted to the time-varying constraints originated by the dynamic environment where the system is evolving. In particular, the focus is on recursive feasibility and stability of model predictive control with time-varying state constraints originating, e.g., from safety requirements in autonomous driving or human-robot cooperation (see Section 3.2).

2 Compositional verification

The verification of a dynamic system is a challenging problem and, despite recent advances, scalability remains a major obstacle. The goal of this task is to examine how an analysis of the components of a system can be exploited to provide guarantees for the composed system. This becomes particularly compelling to address large scale applications where multiple subsystems are interacting like, e.g., in a smart grid involving loads, generators, storage systems, that are interconnected through the electrical grid.

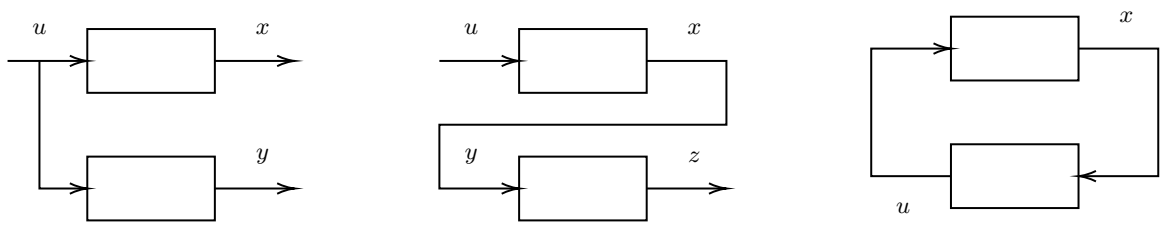
The idea behind compositional verification is to apply costly computations to smaller subsystems, and then combine the results to infer properties about the composed systems. Because the analysis steps are so much more complex than the system description itself, we can assume without much penalty that we have a monolithic description of the system, and then subdivide it into subsystem that suit our method of analysis. In the remainder of this section, when we speak of decomposing the system, this does not necessarily imply that we actually construct the entire system description and then manipulate the description to obtain subsystems. Often, a structural analysis of the system, which can itself be compositional, can identify which subsystems should be analyzed in which configuration. However, by speaking about the (implicitly or explicitly) composed system, we can take the decomposition to the level of individual variables, which allows us to identify the decomposition that is most advantageous in terms of the approximation error. We distinguish the following cases of how variables depend on each other, illustrated in Fig.2:

1. *Parallel*: If variables do not depend on each other (maybe with common inputs), the decomposition is straightforward.

2. *Cascade*: The variables depend on each other in a cascade structure, where the variables can be ordered such that one depends on its predecessors but not on its successors. This structure can be analyzed compositionally by starting with the first component in the cascade, then moving on to its successor, etc.

3. *Feedback*: Some variables are connected in a feedback loop. Decomposing the loop could lead to unstable behavior, in which case the analysis would be overly conservative. Therefore, it may be better to analyze the entire loop together (without decomposition). However, the algorithm we propose in Sect. 2.1.3 can handle even this case, since it exploits a mixed composed/decomposed approach.

It is clear that, in general, a separate analysis of the components will incur an (over-) approximation error that can be much greater than the composed system. Note that this error



(a) parallel (b) cascade (c) feedback

Figure 2: Dependencies between subsystems

is distinctly worse in set-based analysis compared to numerical simulation: As we will show in Section 2.1.3 with reference to linear systems, the approximation error is at most linear in the diameter of the set. Taking numerical simulation to be the limit case of set-based analysis with sets of diameter zero, this means that the approximation error is zero if our algorithm would be applied to numerical analysis. This is due to the mixed composed/decomposed approach that will be described later.

In Section 2.1, we describe the decomposition problem for the important class of linear time-invariant systems with nondeterministic inputs. This class of systems actually refers to differential inclusions: approximation errors or modeling uncertainties can be added as set-valued bias terms that guarantee that the actual behavior is included in the set of solutions of the system. The question of the approximation error that can be achieved will be discussed on a general level in Section 2.1.2 and in Section 2.1.3 with specific reference to the reachability algorithm.

In Section 2.2 a verification method for cascade Mixed Logical Dynamical (MLD) systems exploiting decomposition and model reduction is presented. MLD systems ([8]) are equivalent to various classes of hybrid models [23, 7], and, in particular, to PieceWise Affine (PWA) systems commuting between a finite set of affine dynamics (the modes), each one associated with a polyhedral region in the partitioned joint state and input space. The PWA systems class is important from a theoretical as well as a practical point of view, since arbitrarily complex nonlinear systems can be approximated with PWA systems with an approximation error that depends on the size of the polyhedral regions.

2.1 Decomposing continuous-time systems

We consider a linear time invariant (LTI) dynamical system of the form

$$\dot{x} = Ax, \tag{1}$$

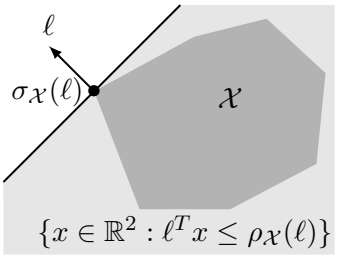


Figure 3: Geometric interpretation of the support function of a convex set $\mathcal{X}$

with $x \in \mathbb{R}^n$, $x(0) \in \mathcal{X}_0$. The solutions of (1) are functions $x(t) = e^{At}x(0)$, which we can write as sets

$$\mathcal{X}(t) = e^{At}\mathcal{X}_0.$$

We call the set-valued function $\mathcal{X}(t)$ a *reach tube*.

Assuming we have an algorithm to approximate the reachable states of the system, can we decompose A and compute the reachable states on the components at a lower cost? Without loss of generality, we consider in the sequel a decomposition of the system into two subsystems. By applying this procedure recursively, we can achieve a decomposition into an arbitrary number of subcomponents.

We decompose the system into two subsystems, one producing a reach tube $\mathcal{Y}(t)$ and one producing a reach tube $\mathcal{Z}(t)$ such that the result is an overapproximation of the reach tube $\mathcal{X}(t)$.

Definition 1 (Cartesian decomposition)**.** *Reach tubes $\mathcal{Y}(t)$, $\mathcal{Z}(t)$ are a* Cartesian decomposition *of a reach tube $\mathcal{X}(t)$ if*

$$\mathcal{X}(t) \subseteq \mathcal{Y}(t) \times \mathcal{Z}(t). \tag{2}$$

Notation Let us introduce some notation. We denote with I_n the identity matrix and with 0_n the zero matrix of dimension $n \times n$. The p-norm of an n-dimensional vector x is $\|x\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$. From the dual norm property [24], the matrix norm induced by the p-norm on vectors satisfies, for any $1 \leq p \leq \infty$,

$$\left\|A^T\right\|_{\frac{p}{p-1}} = \|A\|_p . \tag{3}$$

The logarithmic norm of A is $\mu_p(A) = \lim_{\delta \to 0_+}(\|I_n + \delta A\|_p - 1)/\delta$ [41]. It provides the bound

$$\left\|e^{At}\right\|_p \leq e^{\mu_p(A)t}. \tag{4}$$

Moreover, from (3) we deduce

$$\mu_{\frac{p}{p-1}}(A^T) = \mu_p(A). \tag{5}$$

Decomposing the Initial States We can deduce from (2) that the best choice for the initial states of the subsystems is their projection.

Lemma 2.1. *The smallest sets of initial states of any Cartesian decomposition are* $\mathcal{Y}_0 = \begin{pmatrix} I_{n_1} & 0_{n_2} \end{pmatrix} \mathcal{X}_0$ *and* $\mathcal{Z}_0 = \begin{pmatrix} 0_{n_1} & I_{n_2} \end{pmatrix} \mathcal{X}_0$.

Proof. Note that

$$\mathcal{Y}(t) \times \mathcal{Z}(t) = \begin{pmatrix} I_{n_1} \\ 0_{n_2} \end{pmatrix} \mathcal{Y}(t) \oplus \begin{pmatrix} 0_{n_1} \\ I_{n_2} \end{pmatrix} \mathcal{Z}(t).$$

Setting $t = 0$ in $\mathcal{X}(t) \subseteq \mathcal{Y}(t) \times \mathcal{Z}(t)$, we get

$$\mathcal{X}_0 \subseteq \begin{pmatrix} I_{n_1} \\ 0_{n_2} \end{pmatrix} \mathcal{Y}_0 \oplus \begin{pmatrix} 0_{n_1} \\ I_{n_2} \end{pmatrix} \mathcal{Z}_0.$$

Multiplying both sides with $\begin{pmatrix} I_{n_1} & 0_{n_2} \end{pmatrix}$, we get

$$\begin{aligned} \begin{pmatrix} I_{n_1} & 0_{n_2} \end{pmatrix} \mathcal{X}_0 &\subseteq \begin{pmatrix} I_{n_1} & 0_{n_2} \end{pmatrix} \begin{pmatrix} I_{n_1} \\ 0_{n_2} \end{pmatrix} \mathcal{Y}_0 \oplus \begin{pmatrix} I_{n_1} & 0_{n_2} \end{pmatrix} \begin{pmatrix} 0_{n_1} \\ I_{n_2} \end{pmatrix} \mathcal{Z}_0 \\ &= \begin{pmatrix} I_{n_1} \end{pmatrix} \mathcal{Y}_0 \oplus 0_n = \mathcal{Y}_0, \end{aligned}$$

and similarly for $\mathcal{Z}_0$. □

In general we have $\mathcal{X}_0 \neq \mathcal{Y}_0 \times \mathcal{Z}_0$. This means that even if the two subsystems perfectly model the dynamics of the system, the Cartesian decomposition incurs an approximation error due to the overapproximation of the initial states. Let

$$\hat{\mathcal{X}}(t) := e^{At} \hat{\mathcal{X}}_0, \text{ with } \hat{\mathcal{X}}_0 := \mathcal{Y}_0 \times \mathcal{Z}_0. \tag{6}$$

It is straightforward that any Cartesian decomposition satisfies

$$\hat{\mathcal{X}}(t) \subseteq \mathcal{Y}(t) \times \mathcal{Z}(t), \tag{7}$$

and equality holds if the decomposition perfectly matches the system dynamics. Depending on how we compute $\mathcal{Y}(t), \mathcal{Z}(t)$, equality may not hold.

Definition 2 (Exactness)**.** *A decomposition is* exact *if* $\hat{\mathcal{X}}(t) = \mathcal{Y}(t) \times \mathcal{Z}(t)$.

Before we move on to examining the error of decompositions that are not exact, we first introduce the notation and some basic notions about convex sets.

2.1.1 Fundamentals of set approximations

Minkowski sum and support function Recall that the Minkowski sum of two sets $\mathcal{X}$ and $\mathcal{Y}$ is the set of sums of elements of $\mathcal{X}$ and $\mathcal{Y}$, namely $\mathcal{X} \oplus \mathcal{Y} = \{x + y : x \in \mathcal{X} \text{ and } y \in \mathcal{Y}\}$. There is a relation between products of sets and Minkowski sum: if $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{Y} \subseteq \mathbb{R}^m$, then $\mathcal{X} \times \mathcal{Y} = (\mathcal{X} \times \{0_m\}) \oplus (\{0_n\} \times \mathcal{Y})$.

Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a compact convex set. The support function of $\mathcal{X}$, denoted $\rho_{\mathcal{X}}$, is defined as $\rho_{\mathcal{X}} : \mathbb{R}^n \to \mathbb{R}$,

$$\rho_{\mathcal{X}}(\ell) = \max_{x \in \mathcal{X}} \ell^T x.$$

We will use the following well-known properties.

Proposition 1. *For all compact convex sets $\mathcal{X}$, $\mathcal{Y}$ in $\mathbb{R}^n$, for all $n \times n$ real matrices A, all positive scalars λ, and all vectors $\ell \in \mathbb{R}^n$, we have:*

1. $\rho_{A\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(A^T \ell)$
2. $\rho_{\lambda\mathcal{X}}(\ell) = \rho_{\mathcal{X}}(\lambda\ell) = \lambda\rho_{\mathcal{X}}(\ell)$
3. $\rho_{\mathcal{X} \oplus \mathcal{Y}}(\ell) = \rho_{\mathcal{X}}(\ell) + \rho_{\mathcal{Y}}(\ell)$

Approximations of Convex Sets We recall some basic notions for approximating convex sets. Throughout the paper, we denote convex sets as $\mathcal{X} \subseteq \mathbb{R}^n$, $\mathcal{Y} \subseteq \mathbb{R}^{n_1}$, $\mathcal{Z} \subseteq \mathbb{R}^{n_2}$, with $n_1 + n_2 = n$. Let $\mathcal{B}_p^n$ be the ball of the p-norm in n dimensions, $\mathcal{B}_p^n = \{x \mid \|x\|_p \leq 1\}$. Its support function is $\rho_{\mathcal{B}_p^n}(d) = \|d\|_{\frac{p}{p-1}}$. We measure the approximation accuracy as the Hausdorff distance with respect to a given p-norm:

$$d_H^p(\mathcal{X}, \hat{\mathcal{X}}) = \inf\{\varepsilon \geq 0 \mid \hat{\mathcal{X}} \subseteq \mathcal{X} \oplus \varepsilon\mathcal{B}_p^n \text{ and } \mathcal{X} \subseteq \hat{\mathcal{X}} \oplus \varepsilon\mathcal{B}_p^n\}.$$

Another useful characterization of the Hausdorff distance is the following. Let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^n$ be polytopes. Then,

$$d_H^p(\mathcal{X}, \mathcal{Y}) = \max_{\ell \in \mathcal{B}_p^n} |\rho_{\mathcal{Y}}(\ell) - \rho_{\mathcal{X}}(\ell)|.$$

In the special case $\mathcal{X} \subseteq \mathcal{Y}$, the absolute value can be removed.

Lemma 2.2. *Let $\hat{\mathcal{X}}$ with $\mathcal{X} \subseteq \hat{\mathcal{X}}$, and let $\varepsilon > 0$. Then, $d_H^p(\mathcal{X}, \hat{\mathcal{X}}) \leq \varepsilon$ if and only if for all $d \in \mathbb{R}^n$,*

$$\rho_{\hat{\mathcal{X}}}(d) \leq \rho_{\mathcal{X}}(d) + \varepsilon \|d\|_{\frac{p}{p-1}}.$$

Coordinate transformations scale the approximation error as follows.

Lemma 2.3. *If $\mathcal{X} \subseteq \hat{\mathcal{X}}$, then $d_H^p(M\mathcal{X}, M\hat{\mathcal{X}}) \leq \|M\|_p d_H^p(\mathcal{X}, \hat{\mathcal{X}})$. This bound is tight in the sense that for some $\varepsilon > 0$ there is a point on the boundary of $M(\mathcal{X} \oplus \varepsilon\mathcal{B}_p)$ whose distance to $M\mathcal{X}$ in the p-norm is $\|M\|_p$.*

Proof. Expressed with support functions,

$$\begin{aligned}\rho_{M\hat{\mathcal{X}}}(d) = \rho_{\hat{\mathcal{X}}}(M^T d) &\leq \rho_{\mathcal{X}}(M^T d) + \varepsilon\rho_{\mathcal{B}_p}(M^T d) \\ &\leq \rho_{\mathcal{X}}(M^T d) + \left\|M^T\right\|_{\frac{p}{p-1}} \varepsilon \|d\|_{\frac{p}{p-1}}\end{aligned}$$

With (3), $\left\|M^T\right\|_{\frac{p}{p-1}} = \|M\|_p$.

The existence of the point on the boundary follows from the definition of the matrix norm, which implies existence of d, and the definition of the support function, which implies existence of a point. □

The Hausdorff distance is subadditive for the Cartesian product.

Lemma 2.4. *For $d = \binom{a}{b}$, $\|a\|_p \leq \|d\|_p$, $\|b\|_p \leq \|d\|_p$, and $\|d\|_1 = \|a\|_1 + \|b\|_1$.*

Lemma 2.5. *Assume that $\mathcal{Y} \subseteq \hat{\mathcal{Y}} \subseteq \mathbb{R}^{n_1}$ and $\mathcal{Z} \subseteq \hat{\mathcal{Z}} \subseteq \mathbb{R}^{n_2}$. Then, for $1 \leq p < \infty$, $d_H^p(\mathcal{Y} \times \mathcal{Z}, \hat{\mathcal{Y}} \times \hat{\mathcal{Z}}) \leq d_H^p(\mathcal{Y}, \hat{\mathcal{Y}}) + d_H^p(\mathcal{Z}, \hat{\mathcal{Z}})$. For $p = \infty$, $d_H^\infty(\mathcal{Y} \times \mathcal{Z}, \hat{\mathcal{Y}} \times \hat{\mathcal{Z}}) \leq \max\left\{d_H^\infty(\mathcal{Y}, \hat{\mathcal{Y}}), d_H^\infty(\mathcal{Z}, \hat{\mathcal{Z}})\right\}$.*

Proof. Let $p \geq 1$, $d_H^p(\mathcal{Y}, \hat{\mathcal{Y}}) \leq \varepsilon_1$, and $d_H^p(\mathcal{Z}, \hat{\mathcal{Z}}) \leq \varepsilon_2$. By definition, $\mathcal{Y} \subseteq \hat{\mathcal{Y}} \subseteq \mathcal{Y} \oplus \varepsilon_1\mathcal{B}_p$ and $\mathcal{Z} \subseteq \hat{\mathcal{Z}} \subseteq \mathcal{Z} \oplus \varepsilon_2\mathcal{B}_p$. Expressed with support functions, for any direction d_1,

$$\rho_{\hat{\mathcal{Y}}}(d_1) \leq \rho_{\mathcal{Y}}(d_1) + \varepsilon_1\rho_{\mathcal{B}_p^{n_1}}(d_1) = \rho_{\mathcal{Y}}(d_1) + \varepsilon_1 \|d_1\|_{\frac{p}{p-1}},$$

and similarly for $\hat{\mathcal{Z}}$ and any direction d_2. Applying

$$\mathcal{Y} \times \mathcal{Z} = \begin{pmatrix} I_{n_1} \\ 0_{n_2} \end{pmatrix} \mathcal{Y} \oplus \begin{pmatrix} 0_{n_1} \\ I_{n_2} \end{pmatrix} \mathcal{Z},$$

we get for any direction $d = \binom{d_1}{d_2}$, with Lemma 2.4,

$$\begin{aligned}\rho_{\hat{\mathcal{Y}}\times\hat{\mathcal{Z}}}(d) &= \rho_{\hat{\mathcal{Y}}}(d_1) + \rho_{\hat{\mathcal{Z}}}(d_2) \\ &\leq \rho_{\mathcal{Y}}(d_1) + \varepsilon_1 \|d_1\|_{\frac{p}{p-1}} + \rho_{\mathcal{Z}}(d_2) + \varepsilon_2 \|d_2\|_{\frac{p}{p-1}} \\ &= \rho_{\mathcal{Y}\times\mathcal{Z}}(d) + \varepsilon_1 \|d_1\|_{\frac{p}{p-1}} + \varepsilon_2 \|d_2\|_{\frac{p}{p-1}} \\ &\leq \rho_{\mathcal{Y}\times\mathcal{Z}}(d) + (\varepsilon_1 + \varepsilon_2) \|d\|_{\frac{p}{p-1}}.\end{aligned}$$

Let $\varepsilon = \max\{\varepsilon_1, \varepsilon_2\}$ and $p = \infty$, then with Lemma 2.4,

$$\begin{aligned}\rho_{\hat{\mathcal{Y}}\times\hat{\mathcal{Z}}}(d) &\leq \rho_{\mathcal{Y}\times\mathcal{Z}}(d) + \varepsilon \|d_1\|_1 + \varepsilon \|d_2\|_1 \\ &= \rho_{\mathcal{Y}\times\mathcal{Z}}(d) + \varepsilon \|d\|_1.\end{aligned}$$

□

We pay special attention to the case where $\mathcal{X}$ is a polyhedron, since we frequently encounter it in practice. The following lemma allows us to efficiently compute the distance between convex sets $\mathcal{X}, \hat{\mathcal{X}}$ if the constraints of $\mathcal{X}$ are given.

The asphericity of a non-empty bounded convex set is the minimum ratio of the circumradius (the radius of the exterior ball of smallest volume) to the inradius (the radius of the interior ball of largest volume). In general, it can be computed with arbitrary accuracy by solving a sequence of LP problems [15]. In particular, in the case of initial sets given by hyper-rectangles it is trivial to compute it in the sup-norm.

Lemma 2.6 (see [31]). *Let $\mathcal{X}$ be a polytope given by constraints $\bigwedge_i f_i^T x \leq g_i$, $i \in \{1, \dots, m\}$, and we assume WLOG that $||f_i||_p = 1$. Let $p_{\mathcal{X}}$ be the asphericity of $\mathcal{X}$ (in the p-norm). For any polytopic over-approximation, $\hat{\mathcal{X}} \supseteq \mathcal{X}$, it holds*

$$d_H^p(\mathcal{X}, \hat{\mathcal{X}}) \leq p_{\mathcal{X}} \max_i \rho_{\hat{\mathcal{X}}}(f_i) - g_i.$$

2.1.2 Decomposition error

In the following subsections, we compare the error incurred by the decomposition $\hat{\mathcal{X}}(t)$ instead of $\mathcal{X}(t)$.

Similarity Transformations Sometimes the decomposition requires a coordinate transformation with similarity matrix S, i.e., we decompose the system

$$\dot{\bar{x}} = SAS^{-1}\bar{x}$$

with reach tubes $\bar{\mathcal{Y}}(t)$, $\bar{\mathcal{Z}}(t)$ and obtain as approximation of the original system

$$\hat{\mathcal{X}}(t) \subseteq S^{-1}\big(\bar{\mathcal{Y}}(t) \times \bar{\mathcal{Z}}(t)\big), \tag{8}$$

with initial states $\bar{\mathcal{Y}}_0 = S_1\mathcal{X}_0$ and $\bar{\mathcal{Z}}_0 = S_2\mathcal{X}_0$, where $\binom{S_1}{S_2} = S$.

Approximation of the Initial States The distance from $\hat{\mathcal{X}}(t)$ to $\mathcal{X}(t)$ is quantified as follows.

Lemma 2.7. *Let $\mathcal{X}(t) = e^{At}\mathcal{X}_0$, $\hat{\mathcal{X}}(t) = e^{At}\hat{\mathcal{X}}_0$. Then*

$$d_H^p\big(\mathcal{X}(t), \hat{\mathcal{X}}(t)\big) \leq e^{\mu t} d_H^p\big(\mathcal{X}_0, \hat{\mathcal{X}}_0\big),$$

where $\mu = \mu_p(A)$ is the logarithmic p-norm of A.

Proof. From Lemma 2.3 and inequality (4), it follows that

$$d_H^p\big(\mathcal{X}(t), \hat{\mathcal{X}}(t)\big) \leq ||e^{At}|| d_H^p\big(\mathcal{X}_0, \hat{\mathcal{X}}_0\big) \leq e^{\mu t} d_H^p\big(\mathcal{X}_0, \hat{\mathcal{X}}_0\big).$$

□

Note that a similarity transformation does not change how the error evolves over time.

Polyhedral initial states If $\mathcal{X}_0$ is a polytope in constraint form, then Lemma 2.6 can be used to efficiently compute the error bound ε of the initial states for a given similarity matrix S. Note that if $S^{-1} = S^T$, then $f_i^1 = S_1^T S_1 f_i, f_i^2 = S_2^T S_2 f_i$.

Let $\pi_j(v)$ denote the projection of the vector $v \in \mathbb{R}^n$ onto the variables associated to the j-th block, for $j \in \{1, \ldots, b\}$.

Proposition 2. *Let $\mathcal{X}_0$ be a polytope given by constraints $\bigwedge_i f_i^T x \leq g_i$, with asphericity $p_{\mathcal{X}_0}$, and assume WLOG that $||f_i|| = 1$. Consider a decomposition into b blocks. Then*

$$d_H(\mathcal{X}_0, \hat{\mathcal{X}}_0) \leq p_{\mathcal{X}_0} \max_i \sum_{j=1}^{b} \rho_{\mathcal{X}_0}\Big(S_j^T \pi_j\big((S^{-1})^T f_i\big)\Big) - \rho_{\mathcal{X}_0}(f_i). \tag{9}$$

Proof. First we prove the case $b = 2$ blocks. The general case follows by induction. We let $\hat{\mathcal{X}}_0 := S^{-1}(\bar{\mathcal{Y}}_0 \times \bar{\mathcal{Z}}_0)$, and want to prove that

$$d_H(\mathcal{X}_0, \hat{\mathcal{X}}_0) \leq p_{\mathcal{X}_0} \max_i \rho_{\mathcal{X}_0}(S_1^T \pi_1(S f_i)) + \rho_{\mathcal{X}_0}(S_2^T \pi_2(S f_i)) - \rho_{\mathcal{X}_0}(f_i). \tag{10}$$

We have that

$$\begin{aligned}
d_H(\mathcal{X}_0, \hat{\mathcal{X}}_0) &= d_H(\mathcal{X}_0, S^T(S_1\mathcal{X}_0 \times S_2\mathcal{X}_0)) \\
&= \max_{\ell \in \mathcal{B}_p^n} \rho_{S^T(S_1\mathcal{X}_0 \times S_2\mathcal{X}_0)}(\ell) - \rho_{\mathcal{X}_0}(\ell) \\
&= \max_{\ell \in \mathcal{B}_p^n} \rho_{S_1\mathcal{X}_0 \times S_2\mathcal{X}_0}(S\ell) - \rho_{\mathcal{X}_0}(\ell) \\
&= \max_{\ell \in \mathcal{B}_p^n} \rho_{(S_1\mathcal{X}_0 \times \{0_{n_2}\}) \oplus (\{0_{n_1}\} \times S_2\mathcal{X}_0)}(S\ell) - \rho_{\mathcal{X}_0}(\ell) \\
&= \max_{\ell \in \mathcal{B}_p^n} \rho_{S_1\mathcal{X}_0 \times \{0_{n_2}\}}(S\ell) + \rho_{\{0_{n_1}\} \times S_2\mathcal{X}_0}(S\ell) - \rho_{\mathcal{X}_0}(\ell)
\end{aligned}$$

Putting this together,

$$d_H(\mathcal{X}_0, \hat{\mathcal{X}}_0) = \max_{\ell \in \mathcal{B}_p^n} \rho_{\mathcal{X}_0}(S_1^T \pi_1(S\ell)) + \rho_{\mathcal{X}_0}(S_2^T \pi_2(S\ell)) - \rho_{\mathcal{X}_0}(\ell). \tag{11}$$

The conclusion follows from applying Lemma 2.6, and the homogeneity property of the support function.

Iteratively applying the previous proposition, we can handle an arbitrary number of blocks.

□

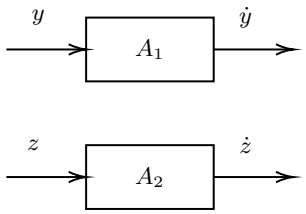


Figure 4: Block-diagonal structure illustrated as a block diagram

Block-Diagonal Decomposition A straightforward decomposition is possible if A is block diagonal:

$$A = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \quad \Rightarrow \quad e^{At} = \begin{pmatrix} e^{A_1 t} & 0 \\ 0 & e^{A_2 t} \end{pmatrix}.$$

Any matrix can be brought to block diagonal form by a coordinate transformation of the form $y = Sx$. For instance, any square matrix A can be brought to *real Jordan form* [24]

$$SAS^{-1} = \begin{pmatrix} J_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & J_K \end{pmatrix}, \quad \text{with} \quad J_k = \begin{pmatrix} C_k & I & 0 \\ 0 & \ddots & I \\ 0 & 0 & C_k \end{pmatrix},$$

with two-dimensional block of the form $C_k = \left(\begin{smallmatrix} a_k & b_k \\ -b_k & a_k \end{smallmatrix}\right)$ for the eigenvalue $a_k \pm ib_k$. The size of the blocks J_k corresponds to the multiplicity of the k-th eigenvalue of A. Every system A with distinct eigenvalues can therefore be decomposed into 2-dimensional subsystems, so the decisive question is whether the initial condition can also be decomposed in the transformed coordinates.

Proposition 3. *Consider a system $\dot{x} = Ax$ and nonsingular S such that*

$$SAS^{-1} = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix},$$

with $A_1 \in \mathbb{R}^{n_1 \times n_1}$ and $A_2 \in \mathbb{R}^{n_2 \times n_2}$. Let $S_1 \in \mathbb{R}^{n_1 \times n}$ be the first n_1 rows of S and $S_2 \in \mathbb{R}^{n_2 \times n}$ be the remaining rows, i.e., $\left(\begin{smallmatrix} S_1 \\ S_2 \end{smallmatrix}\right) = S$. Let $\dot{Y} = A_1 Y$, $\dot{Z} = A_2 Z$ with $\mathcal{Y}_0 = S_1 \mathcal{X}_0$ and $\mathcal{Z}_0 = S_2 \mathcal{X}_0$. Then $\hat{\mathcal{X}}(t) = S^{-1}\big(\mathcal{Y}(t) \times \mathcal{Z}(t)\big)$.

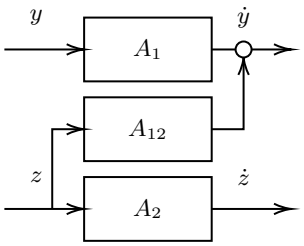


Figure 5: Upper-triangular structure illustrated as a block diagram

Upper triangular decomposition If the system is of upper triangular form, it is straightforward and well-known that the solution $x(t)$ can be computed by first computing $z(t)$ and then solving for $y(t)$ using $z(t)$ as an input. The extension to sets is however not entirely trivial, since reach tubes do not preserve trajectories. Different sets of trajectories could produce the same reach tube $\mathcal{Z}(t)$, but might lead to different reach tube $\mathcal{X}(t)$. We show through set-based integration that this is not the case and the decomposition is valid even with sets.

Proposition 4. *Consider a system $\dot{x} = Ax$ with matrix A of the form*

$$A = \begin{pmatrix} A_1 & A_{12} \\ 0 & A_2 \end{pmatrix}.$$

Let the decomposed systems be

$$\dot{Y} = A_1 Y + A_{12} Z, \tag{12}$$

$$\dot{Z} = A_2 Z \tag{13}$$

with $\mathcal{Y}_0 = \begin{pmatrix} I_{n_1} & 0_{n_2} \end{pmatrix} \mathcal{X}_0$ and $\mathcal{Z}_0 = \begin{pmatrix} 0_{n_1} & I_{n_2} \end{pmatrix} \mathcal{X}_0$. Then $\hat{\mathcal{X}}(t) \subseteq \mathcal{Y}(t) \times \mathcal{Z}(t)$.

Proof. The solution of the system is

$$\mathcal{X}(t) = e^{At}\mathcal{X}_0 = \sum_{k=0}^{\infty} A^k \tfrac{t^k}{k!} \mathcal{X}_0 = \begin{pmatrix} e^{A_1 t} & \int_0^t e^{A_1(t-s)} A_{12} e^{A_2 s} ds \\ 0 & e^{A_2 t} \end{pmatrix} \mathcal{X}_0$$

In the decomposition, we first obtain the solution $\mathcal{Z}(t) = e^{A_2 t}\mathcal{Z}_0$. For a given trajectory $Z(t)$, the solution for $\mathcal{Y}(t)$ is known to be

$$\mathcal{Y}(t) = e^{A_1 t}\mathcal{Y}_0 \oplus e^{A_1 t} \int_{s=0}^{t} e^{-A_1 s} A_{12} Z(s) ds,$$

Since $Z(t) \in \mathcal{Z}(t)$, we have that $\mathcal{Y}(t)$ includes all possible solutions (but maybe more). □

The degree of the overapproximation obtained through the decomposition depends on the diameter of $\mathcal{Z}(t)$, which in turn depends only on the diameter of $\mathcal{Z}_0$ and $\left\| e^{A_2 t} \right\|$.

The *real Schur decomposition* transforms any square matrix A into upper triangular form with block size 2×2. A potential advantage of the Schur form over the real Jordan form is numerical stability. The downside of any block triangular decomposition is the fact that it cannot be parallelized.

2.1.3 Decomposed reachability algorithm

In this section, we present the reachability algorithm, which was published in [13]. All set-based compositional verification approaches known to us use the following sequence:

1. decomposing the system in continuous time,
2. discretizing time and computing the appropriate initial sets,
3. computing the reachable sets of the subsystems,
4. either combine the reach sets to a reach set of the composed system, or check properties directly on the subsystem.

The starting point of this algorithm – and ultimately the root of its success – was the observation that the time discretization step (essentially, computing the matrix exponential) is orders of magnitudes faster than the set computations required for reachability. Based on this observation, we change the order of the first two steps to discretize the composed system instead:

1. discretizing time and computing the appropriate initial sets,
2. decomposing the system,
3. computing the reachable sets of the subsystems,
4. either combine the reach sets to a reach set of the composed system, or check properties directly on the subsystem.

Discretizing Time Starting from a continuous-time system of the form

$$\dot{x} = Ax + u, \qquad u \in \mathcal{U}, x(0) \in \mathcal{X}_0,$$

we bring the system to a discrete-time form

$$\mathcal{X}(k+1) = \Phi\mathcal{X}(k) \oplus \mathcal{V}(k),$$

where

$$\begin{aligned} \Phi &= e^{A\delta} \\ \mathcal{X}(0) &= \mathrm{CH}\big(\mathcal{X}_0, \Phi\mathcal{X}_0 \oplus \delta\mathcal{U}(0) \oplus E_\psi(\mathcal{U}(0),\delta) \oplus E^+(\mathcal{X}_0,\delta)\big) \\ \mathcal{V}(k) &= \delta\mathcal{U}(k) \oplus E_\psi(\mathcal{U}(k),\delta), \quad \forall\ k = 0,1,\ldots,N \end{aligned} \tag{14}$$

Note that in this general form we admit the possibility that the set $\mathcal{U}$ may change from one time step to the next. If we are only interested in looking at the system at discrete points in time – without covering the states reached in the dense time between – the transformation is

$$\begin{aligned} \Phi &= e^{A\delta} \\ \mathcal{X}(0) &= \mathcal{X}_0 \\ \mathcal{V}(k) &= \Phi_1(A,\delta)\mathcal{U}(k), \quad \forall\ k = 0,1,\ldots,N \end{aligned} \tag{15}$$

The auxiliary error terms (bloating terms) are

$$\begin{aligned} E_\psi(\mathcal{U}(k),\delta) &:= \boxdot(\Phi_2(|A|,\delta) \boxdot (A\mathcal{U}(k))) \\ E^+(\mathcal{X}_0,\delta) &:= \boxdot(\Phi_2(|A|,\delta) \boxdot (A^2\mathcal{X}_0)), \end{aligned}$$

where the matrices $\Phi_1(A,\delta)$ and $\Phi_2(A,\delta)$ are defined via

$$\Phi_1(A,\delta) := \sum_{i=0}^{\infty} \frac{\delta^{i+1}}{(i+1)!}A^i, \;\; \Phi_2(A,\delta) := \sum_{i=0}^{\infty} \frac{\delta^{i+2}}{(i+2)!}A^i.$$

Decomposed Reachability We now consider the system decomposed into b row-blocks:

$$\mathcal{X}(k+1) = \begin{pmatrix} \Phi_{11} & \cdots & \Phi_{1b} \\ \vdots & \ddots & \vdots \\ \Phi_{b1} & \cdots & \Phi_{bb} \end{pmatrix} \mathcal{X}(k) \oplus \mathcal{V}(k). \tag{16}$$

In this recurrence, the approximation error of the k-th step is propagated, and possibly amplified, in step $k+1$. This can be partly avoided by using a non-recursive form [21]. We present two scenarios, which differ in whether the sequence of input sets is constant or not. Let Φ_{ij}^k be the submatrix of Φ^k corresponding to the indices of the submatrix Φ_{ij} of Φ.

Constant input sets Assuming that the sets $\mathcal{V}$ do not depend on k, the non-recurrent form of (16) is:

$$\begin{aligned} \mathcal{X}(k) &= \Phi^k\mathcal{X}(0) \oplus \mathcal{W}(k) \\ \mathcal{W}(k+1) &= \mathcal{W}(k) \oplus \Phi^k\mathcal{V}, \quad \mathcal{W}(0) := \{\mathbf{0}_n\}. \end{aligned}$$

The decomposed map, for $i = 1, \ldots, b$, is:

$$\begin{aligned} \hat{\mathcal{X}}_i(k) &= \bigoplus_{j=1}^{b} \Phi_{ij}^k \hat{\mathcal{X}}_j(0) \oplus \hat{\mathcal{W}}_i(k) \\ \hat{\mathcal{W}}_i(k+1) &= \hat{\mathcal{W}}_i(k) \oplus [\Phi_{i1}^k \cdots \Phi_{ib}^k]\mathcal{V}, \quad \hat{\mathcal{W}}_i(0) := \{\mathbf{0}_2\}. \end{aligned} \tag{17}$$

Note that the set $[\Phi_{i1}^k \cdots \Phi_{ib}^k]\mathcal{V}$ in (17) is of low dimension and corresponds to the i-th block.

Time-varying input sets Assuming that the sequence of inputs depends on k, the non-recurrent form of (16) is:

$$\begin{aligned} \mathcal{X}(k) &= \Phi^k \mathcal{X}(0) \oplus \mathcal{W}(k) \\ \mathcal{W}(k+1) &= \Phi \mathcal{W}(k) \oplus \mathcal{V}(k), \quad \mathcal{W}(0) := \{\mathbf{0}_n\}. \end{aligned}$$

The decomposed map, for $i = 1, \ldots, b$, is:

$$\begin{aligned} \hat{\mathcal{X}}_i(k) &= \bigoplus_{j=1}^{b} \Phi_{ij}^k \hat{\mathcal{X}}_j(0) \oplus \hat{\mathcal{W}}_i(k) \\ \hat{\mathcal{W}}_i(k+1) &= \bigoplus_{j=1}^{b} \Phi_{ij} \hat{\mathcal{W}}_j(k) \oplus \hat{\mathcal{V}}_i, \quad \hat{\mathcal{W}}_i(0) := \{\mathbf{0}_2\}. \end{aligned} \tag{18}$$

Error Analysis in Discrete Time We briefly present the results from [13]. We base the analysis on the matrix norm of the transformation matrix Φ. There exist constants K_Φ and α_Φ such that

$$\left\| \Phi^k \right\|_p \leq K_\Phi \alpha_\Phi^k, \qquad k \geq 0.$$

If $\Phi = e^{A\delta}$, one choice is $\alpha_\Phi = e^{\lambda\delta}$ with λ the spectral abscissa (largest real part of any eigenvalue of A), although it may not be possible to compute the corresponding K_Φ efficiently. In this case, $\alpha_\Phi \leq 1$ if the system is stable. Another choice is to let $\alpha_\Phi = e^{\mu\delta}$, with μ the logarithmic norm of A and $K_\Phi = 1$. In this case, α_Φ may be larger than 1 even for stable systems. Note that in both cases $\alpha_\Phi \to 1$ as $\delta \to 0$.

It turns out that the approximation error is linear in the width of the initial states and the inputs, and in the decomposition errors of the initial states and the input sets. For unstable systems, or time steps not large enough, the input set can become the dominating source of error, e.g., considering cases with $\alpha_\Phi > \frac{1}{2}$.

Proposition 5. *[13] Let the decomposition error of the initial states $\mathcal{X}(0)$ be bounded by $\varepsilon^x \geq d_H^p\big(\mathcal{X}(0), \hat{\mathcal{X}}(0)\big)$, and let the decomposition error of $\mathcal{V}$ be bounded by $\varepsilon^v \geq d_H^p\big(\mathcal{V}, \hat{\mathcal{V}}\big)$. Let Δ_j^x be the diameter of $\hat{\mathcal{X}}_j(0)$, and $\Delta_{\text{sum}}^x = \sum_{j=1}^b \Delta_j^x$. Let Δ_j^v be the diameter of $\hat{\mathcal{V}}_j$, and*

$\Delta^v_{\text{sum}} = \sum_{j=1}^{b} \Delta^v_j$. *Then the approximation error due to decomposition, at step k, is bounded by*

$$d_H^p\big(\hat{\mathcal{X}}(k), \mathcal{X}(k)\big) \le K_\Phi\Big(\alpha_\Phi^k\big(b\Delta^x_{\text{sum}} + \varepsilon^x\big) + \big(b\Delta^v_{\text{sum}} + \varepsilon^v\big)\alpha_\Phi \frac{1-\alpha_\Phi^{k-1}}{1-\alpha_\Phi}\Big) + \varepsilon^v.$$

If $\alpha_\Phi < 1$ (stable system), the error is bounded for all k by

$$d_H^p\big(\hat{\mathcal{X}}(k), \mathcal{X}(k)\big) \le K_\Phi\Big(b\Delta^x_{\text{sum}} + \varepsilon^x + \big(b\Delta^v_{\text{sum}} + \varepsilon^v\big)\frac{\alpha_\Phi}{1-\alpha_\Phi}\Big) + \varepsilon^v.$$

Error Analysis in Continuous Time In dense time, the decomposition error of $\mathcal{V}$ is bounded by

$$\varepsilon^v = \delta d_H^p\big(\mathcal{U}, \hat{\mathcal{U}}\big).$$

The decomposition error for the initial states is more complex and harder to estimate. We consider the idealized case where the system is stable with $\alpha_\Phi = e^{-\lambda\delta}$, $\lambda > 0$, for an infinitesimal time step $\delta \to 0$. Then $\alpha_\Phi \to 1 - \lambda\delta$ and $\frac{\alpha_\Phi}{1-\alpha_\Phi} \to \frac{1}{\lambda\delta}$, so that the decomposition error due to the inputs does not go to zero in Prop. 5. Let $\Delta_{\mathcal{X}_0}$, $\Delta_{\mathcal{U}}$ be the sum of the diameters of decomposed sets of $\mathcal{X}_0$ and $\mathcal{U}$. Let $\varepsilon_0^x = d_H^p\big(\mathcal{X}_0, \hat{\mathcal{X}}_0\big)$ and $\varepsilon_0^v = d_H^p\big(\mathcal{U}, \hat{\mathcal{U}}\big)$. For both the discrete time and the dense time case, $\varepsilon^x \to \varepsilon_0^x$, $\Delta^x_{\text{sum}} \to \Delta_{\mathcal{X}_0}$, $\Delta^v_{\text{sum}} \to \delta\Delta_{\mathcal{U}}$ and $\varepsilon^v \to \delta\varepsilon_0^v$. Then Prop. 5 gives a nonzero upper bound

$$d_H^p\big(\hat{\mathcal{X}}(k), \mathcal{X}(k)\big) \le K_\Phi\Big(b\Delta_{\mathcal{X}_0} + \varepsilon_0^x + \big(b\Delta_{\mathcal{U}} + \varepsilon_0^v\big)\frac{1}{\lambda}\Big) + \mathcal{O}(\delta).$$

This indicates that a small time step may be problematic for systems with large time constants (small λ).

Experimental Results In [13], we compared the above algorithms to the best-performing analysis tools: SpaceEx for continuous time and Hylaa for discrete time. On 9 standard benchmarks, with dimensions from 8 to 10913, we obtained a speedup of a factor of 1x–79x. In continuous time, we were able to handle systems up to dimension 10913, while SpaceEx crashed for dimensions larger than 384.

In sum, we can report a speedup of up to two orders of magnitude and pushing the size of treatable systems also by two orders of magnitude. It remains open under which characteristics a system can benefit from a large speedup with our algorithm. While sparsity of the dynamic matrix is a positive indicator, our experiments indicate that it is neither a guarantee nor a requirement.

2.1.4 Industrial case study

In this section we present the results of the evaluation of the decomposition approach presented in Section 2.1.3 on an industrial use case at Bosch.

The source model of the use case consists of partial differential equations (PDEs) which describe chemical processes within an one-dimensional pipe. We discretized the system of PDEs spatially into N homogeneous cells in order to obtain non-linear ODEs. Furthermore, the ODEs are linearized in order to be able to perform reachability analysis with the novel decomposition approach [13] and SpaceEx [1, 19]. Here, SpaceEx is our reference implementation as one of the most performant reachabililty analysis tools for linear systems. Increased performance of reachability analysis tools are of considerable interest within Bosch as they allow the use of more complex and accurate models and enable us to use reachability analysis in automated controller synthesis approaches that provide us with safety guarantees. In case of the use case considered within this section, more accurate models can be realized by increasing the number of spatial cells N. In our evaluations we set N to three different values $N \in \{10, 100, 200\}$. The number of state variables x_i for each model is then equal to N. Being able to increase the size of the model in terms of number of states in a meaningful way makes the selected use case especially suitable for investigating the scalalbility of the novel decomposition approach against existing methods.

The parameters of the reachability tools were set as follows: The discretization time σ (decomposition algorithm) and the sampling time (SpaceEx) were set to $5ms$ with a total time horizon of $T = 5s$. Within SpaceEx, we used the state-of-the-art support function algorithm LGG with flow pipe tolerance set to -1. The non-deterministic initial set χ_0 was defined to be a hypercube from 0 to 0.1 ($\forall i \in 1, \ldots, N : x_i(0) \in [0, 0.1]$).

All experiments were performed on an standard HP Elitebook 840 G2 notebook with Intel Core i5-5300U 2.3GHz and 16GB RAM running Linux. SpaceEx version was v0.9.8f.

Table 1 summarizes the experimental results regarding the runtimes of the novel decomposition algorithm and SpaceEx. Figures 6 to 11 show the reachable sets computed by the new algorithm/SpaceEx for the last state x_N (end of the one-dimensional pipe) for $N \in \{10, 100, 200\}$ over time t.

Regarding computation time, we observe that new decomposition approach is considerably faster for the more accurate spatial discretization in the models with $N = 100$ and $N = 200$ in our data set. The speedup can be more than two orders of magnitude. At the same time, the new algorithm does not introduce any over-approximation compared to SpaceEx. The potential over-approximation introduced by the decomposition approach in contrast to the

	Runtime [s]			
N	new decomp. approach	orig. SpaceEx	Speedup	O.A. [%]
10	0.09	0.26	2.89	-16.80
100	0.56	22.80	40.71	-1.98
200	1.52	169.96	111.82	-3.06

Table 1: Runtime comparison and over approximation (O.A.) results between the implementation of the novel decomposition approach and the state-of-the-art algorithm in SpaceEx for different numbers of state variables N.

results of SpaceEx were computed in the same way as described in [13]: The bounds of the reachset from SpaceEx for the state $x_N(t = T)$ were taken as the baseline and we report the relative deviation. In fact, the bounds of the reachsets are a bit smaller for the reported evaluation at $x_N(T)$ ($N \in \{10, 100, 200\}$) (since the numbers in column "O.A." are negative).

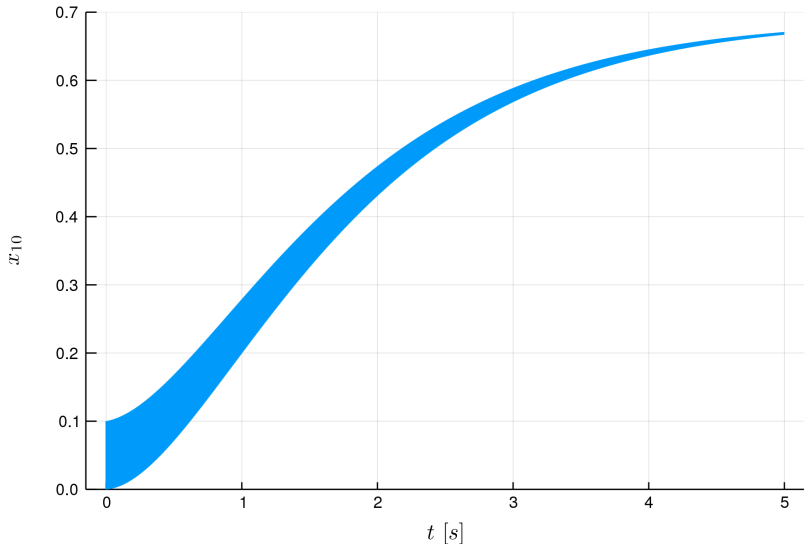


Figure 6: Reachsets computed for state x_{10} ($N = 10$) by the new decomposition approach.

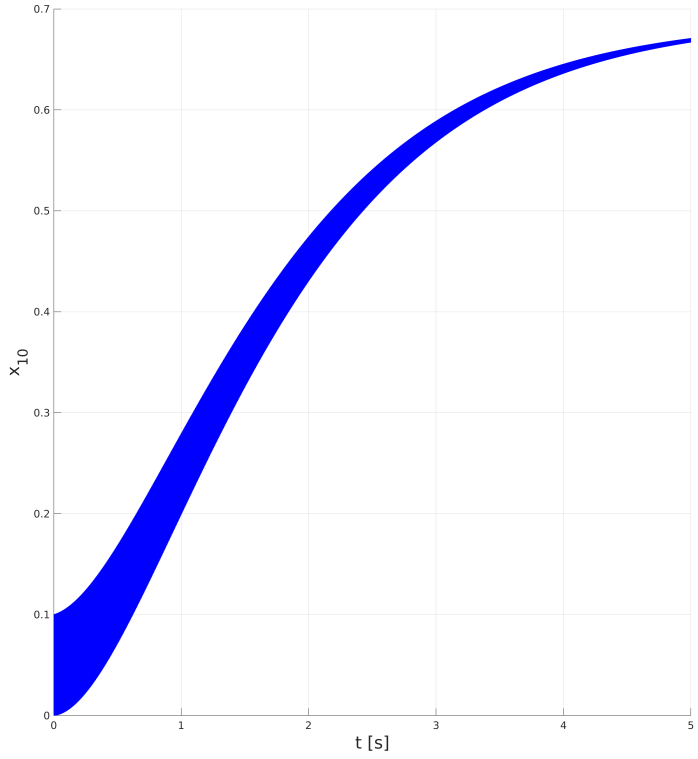


Figure 7: Reachsets computed for state x_{10} ($N = 10$) by SpaceEx.

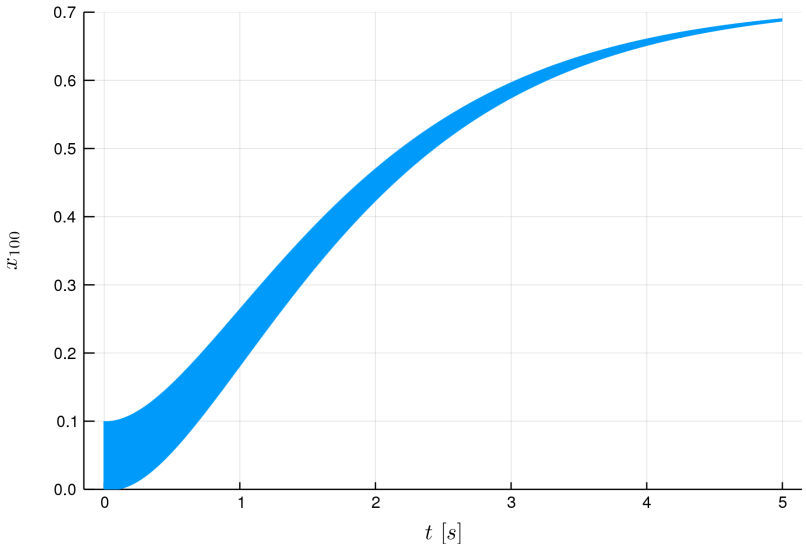


Figure 8: Reachsets computed for state x_{100} ($N = 100$) by the new decomposition approach.

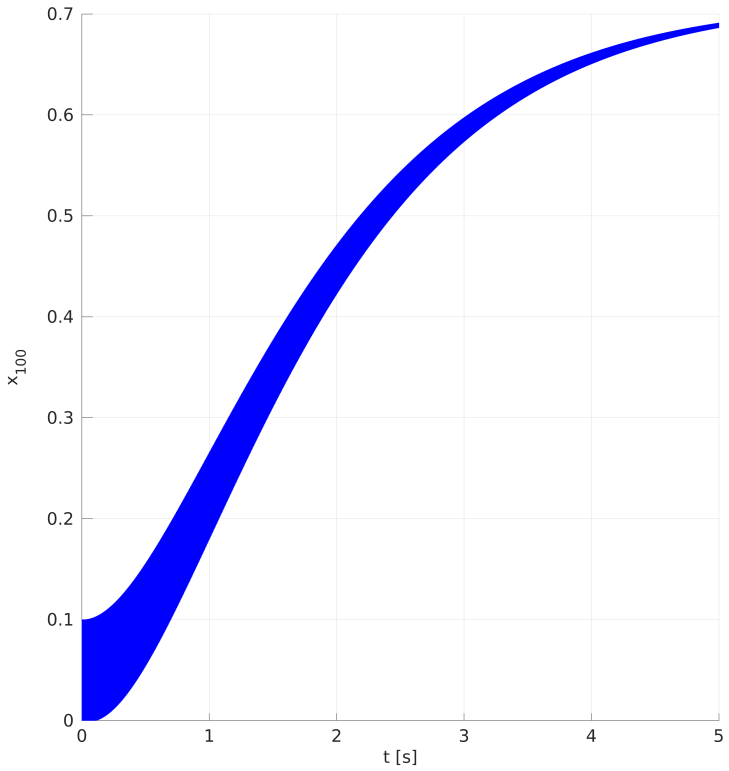


Figure 9: Reachsets computed for state x_{100} ($N = 100$) by SpaceEx.

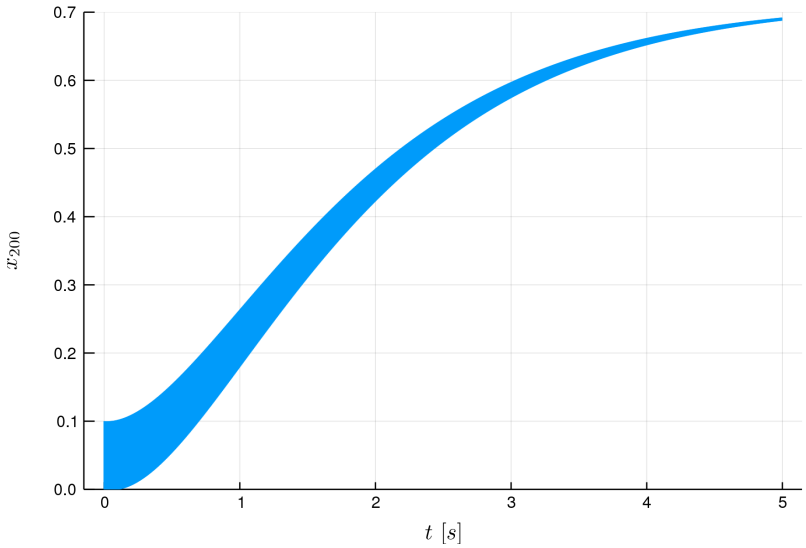


Figure 10: Reachsets computed for state x_{200} ($N = 200$) by the new decomposition approach.

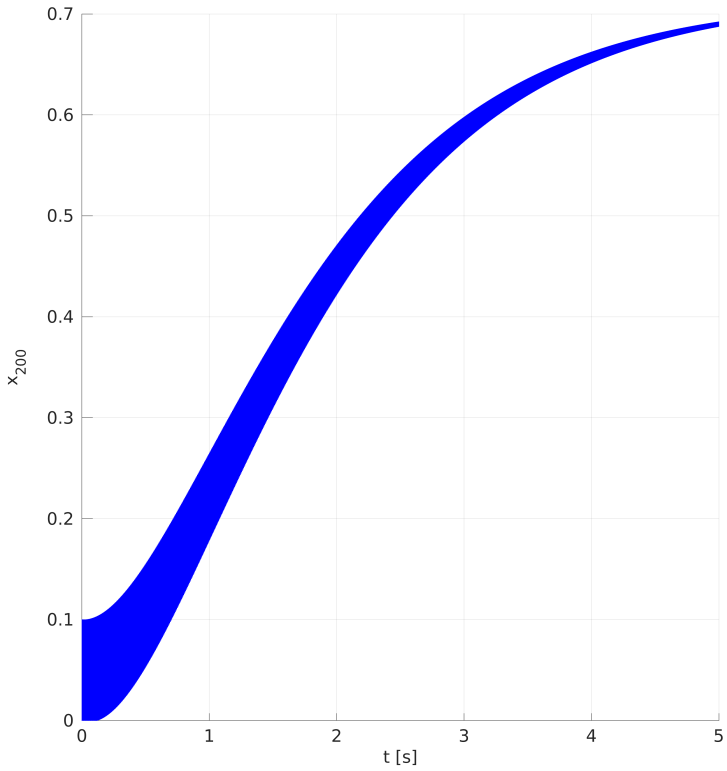


Figure 11: Reachsets computed for state x_{200} ($N = 200$) by SpaceEx.

2.2 Verification of structured mixed logical dynamical systems

Differently from the previous section that was focusing on compositional verification for continuous linear systems, in this section we address compositional verification for systems which are hybrid, i.e., have both a continuous and a discrete state/input component, but still are characterized a (piecewise) linear dynamics governing the continuous state evolution.

More specifically, we consider the problem of verifying a specification that is given in terms of a finite horizon behavior of some output variable of a mixed logical dynamical (MLD) systems ([8]). The output can possibly coincide with the state of the MLD system. More specifically, we shall focus on

i) MLD systems that can be partitioned into a cascade of smaller MLD subsystems,

ii) a system composed of multiple MLD systems that are coupled by some joint constraint.

In both cases verification is reformulated as an optimization program, which is solved through an iterative procedure that is guaranteed to terminate in a finite number of iteration. The solution for cascade systems exploits non influential input detection and model reduction, while the one for constraint coupled systems rests on a parallel scheme for the solution of the optimization program.

2.2.1 Integrating non-influential input detection and model reduction for cascaded systems

We consider a Mixed Logical Dynamical (MLD) system $\tilde{S}$ described by:

$$\begin{aligned} \tilde{x}(k+1) &= A\tilde{x}(k) + B_u u(k) + B_\delta \delta(k) + B_z z(k) + B_{aff} \\ \tilde{y}(k) &= C\tilde{x}(k) + D_u u(k) + D_\delta \delta(k) + D_z z(k) + D_{aff} \\ E_x x(k) &+ E_u u(k) + E_\delta \delta(k) + E_z z(k) \leq E_{aff} \end{aligned} \tag{19}$$

where $\tilde{x} \in \mathbb{R}^{\tilde{n}}$ is the state, $y \in \mathbb{R}^{\tilde{p}}$ is the output, and $u = [u_1, u_2, \ldots, u_m]^\top$ is the input vector, which comprises m scalar control inputs $u_i \in [\underline{u}_i, \overline{u}_i]$, $i = 1, \ldots, m$. As for δ and z, they are respectively binary and continuous auxiliary variables: $\delta \in \{0,\ 1\}^{r_l}$ and $z \in \mathbb{R}^{r_c}$. For MLD systems, the assumption of well-posedness translates into the uniqueness of the solution of the inequalities in (19), i.e., given a state-input pair it is always possible to find a unique corresponding value for the auxiliary variables δ and z.

Our goal is to verify if there exists some assignment for the inputs u_i, $i = 1, \ldots, m$, that make system (19) satisfy a certain specification starting from a given initial condition $\tilde{x}(0) = \tilde{x}_0$. The specification is expressed in terms of the behavior of the output $\tilde{y}$ along a

finite horizon of length $T+1$. Typically, it is given in terms of some constraint on the value of $\tilde{y}$:

$$\tilde{y}^{\star}_{l,h} \leq \tilde{y}(t+h) \leq \tilde{y}^{\star}_{u,h},\ h = 0, 1, \ldots, T. \tag{20}$$

for some $t \geq 0$, where $\tilde{y}^{\star}_{l,h}$ and $\tilde{y}^{\star}_{u,h}$ denote respectively some lower and upper bound on $\tilde{y}(t+h)$.

The verification problem can be rephrased as an input design problem where we aim at determining an input sequence such that the specification is satisfied. Among all possible solutions to this input design problem, we aim at the one that enforces the specification by setting as few inputs as possible (influential inputs), or, equivalently, by maximizing the number of inputs that can take an arbitrary value without compromising the satisfaction of the specification (non-influential inputs). This problem arises within a verification context where the aim is to test the correct functioning of some given system, whose evolution is affected by some inputs. In such a context, the specification may represent an unsafe behavior for the system and the goal is to verify if the system is safe for each possible assignment of the inputs, and, if this is not the case, what are the inputs responsible for the unsafe behavior (influential input detection).

According to the approach that we proposed in [42], the problem can be structured into the following two main steps:

- enlarge the MLD system by embedding in its description the specification, and translate the specification into a reachability condition with some suitable target set for the system output;
- reformulate the problem of reaching the target set as an optimization problem, where, simultaneously, we look for the input sequences that steer the output of the enlarged system into the target set and we also detect which inputs are non-influential.

Depending on the system at hand – and, in particular, for high dimensional systems with both continuous and logic state components – the resulting optimization problem may be difficult to solve. If the system under analysis can be partitioned into a cascade of smaller subsystems, then, it is possible to exploit the cascade structure to simplify the problem. More specifically, the idea developed in [42] is to start from the last subsystem in the cascade, solve the optimization problem for it, and use that solution to formulate an 'intermediate' specification that involves the outputs of the preceding subsystem in the cascade. Then, by iterating this procedure, we can trace all the way back to the inputs of the original system. Note that the detection of non-influential inputs can play a significant role also within the

iteration procedure, since the tightness of each intermediate specification decreases as the number of non-influential inputs increases.

The described decomposition method can be integrated with model reduction of an MLD system so as to reduce its complexity. The model reduction method that we developed in [43] and reported in Deliverable 1.2 of the project can be particularly useful to this purpose since it preserves the input/output behavior of the system and detects inputs that do not affect the output and hence are non-influential.

In the rest of the section we recall the decomposition approach in [42] and describe an iterative algorithm that integrates the model reduction method proposed in [43].

We start by embedding the specification into an enlarged description of the system dynamics and then look for the inputs that make the enlarged system reach a certain region of the state space, which corresponds to the specification being satisfied. For ease of explanation, we focus on the case when the output $\tilde{y}$ is a scalar, i.e., $\tilde{p} = 1$. The procedure that we shall describe can then be easily extended to the more general case of $\tilde{p} > 1$ by applying it to each component of the output.

We introduce T additional scalar state variables $\tilde{x}_{add,i}$, $i = 0, \ldots, T-1$, to store the sequence $\tilde{y}(t+i)$, $i = 0, \ldots, T-1$. The additional state variables evolve according to the equations:

$$\begin{aligned}
&\tilde{x}_{add,0}(k+1) = \tilde{x}_{add,1}(k) \\
&\tilde{x}_{add,1}(k+1) = \tilde{x}_{add,1}(k) \\
&\vdots \\
&\tilde{x}_{add,T-2}(k+1) = \tilde{x}_{add,T-1}(k) \\
&\tilde{x}_{add,T-1}(k+1) = C\tilde{x}(k) + D_u u(k) + D_\delta \delta(k) + D_z z(k) + D_{aff}
\end{aligned} \tag{21}$$

If we define the enlarged state and output variables $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^p$, with $n = \tilde{n} + T$ and $p = T + 1$, as follows

$$x(k) = \begin{bmatrix} \tilde{x}_{add,0}(k) \\ \tilde{x}_{add,1}(k) \\ \vdots \\ \tilde{x}_{add,T-1}(k) \\ \tilde{x}(k) \end{bmatrix} \quad y(k) = \begin{bmatrix} \tilde{y}(k-T) \\ \tilde{y}(k-T+1) \\ \vdots \\ \tilde{y}(k-1) \\ \tilde{y}(k) \end{bmatrix}$$

and embed (21) into (19), we obtain the following enlarged system S:

$$x(k+1) = Ax(k) + B_u u(k) + B_\delta \delta(k) + B_z z(k) + B_{aff}$$

$$y(k) = Cx(k) + D_u u(k) + D_\delta \delta(k) + D_z z(k) + D_{aff}$$
$$E_x x(k) + E_u u(k) + E_\delta \delta(k) + E_z z(k) \leq E_{aff}$$

with appropriately defined matrices and the initialization of the additional state variable set to zero, i.e., $x(0) = x_0 = [0, \ldots, 0, \tilde{x}_0^\top]^\top$.

The problem of designing the inputs of $\tilde{S}$ in (19) so as to impose the satisfaction of a specification with linear constraints of the form (20) translates into that of designing the inputs of S so as to make its output y reach the target set $\mathcal{Y}_f$ given by:

$$\mathcal{Y}_f := \{y \in \mathbb{R}^p : H_a y \leq H_b\}$$

where

$$H_a = \begin{bmatrix} Ip \\ -Ip \end{bmatrix} \quad H_b = \begin{bmatrix} \tilde{y}^\star_{u,0} & \ldots & \tilde{y}^\star_{u,T} & -\tilde{y}^\star_{l,0} & \ldots & -\tilde{y}^\star_{l,T} \end{bmatrix}^\top,$$

with I_p denoting the $p \times p$ identity matrix.

Note that specifications with linear constraints jointly involving $\tilde{y}(k+i)$, $i = 0, 1, \ldots, T$, would still translate into y belonging to some appropriately defined polytopic target set $\mathcal{Y}_f$.

In [42] a procedure to determine a control input sequence that steers the output of the system in the target set $\mathcal{Y}_f$ in some finite time say T_f starting from x_0 while simultaneously minimizing the number of influential inputs is described.

Let us consider the case when the overall system $\mathcal{S}$ is structured or can be reduced to the cascade of two systems, say $\mathcal{S}_1$ feeding with its output $\mathcal{S}_2$, as in Figure 12.

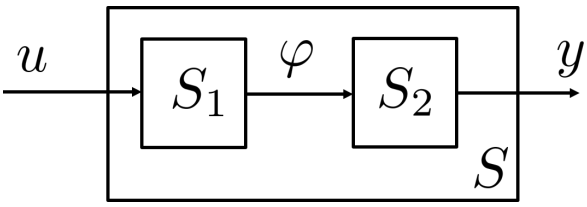


Figure 12: System S is the cascade of subsystems S_1 and S_2.

The problem then becomes that of designing the inputs of $\mathcal{S}_1$ so as to make its outputs evolve as the testing signals of $\mathcal{S}_2$. Note that the detection of non-influential inputs can simplify the problem, since setting only a limited number of inputs of S_1 can indeed save time and reduce the effort in the testing phase.

Exploiting the cascade structure of S, Algorithm 1 decomposes the problem into two sub-problems for the lower dimensional systems $\mathcal{S}_1$ and $\mathcal{S}_2$. In Algorithm 1 variables x_1 and x_2 denote, respectively, the state of systems S_1 and S_2.

Algorithm 1 Iterative verification algorithm for a cascading system with model reduction

Input: Systems S_1, S_2 with output φ and y; Spec $y(T) \in \mathcal{Y}_f$; Initialization $x_{1,0}$, $x_{2,0}$.
$(S_{2,r}, x_{2,r}, \varphi_r) = \mathit{Reduce}(S_2, y)$
$(S_{1,r}, x_{1,r}, u_r) = \mathit{Reduce}(S_1, \varphi_r)$
$T_2 = \mathit{MinimumTimeForSpec}(S_{2,r}, \mathcal{Y}_f, x_{2,r,0})$
while 1
  $\varphi_r^* = \mathit{solveNonInfluentialInputs}(P_2(S_{2,r}, \mathcal{Y}_f, T_2, x_{2,r,0}))$
  $\Phi_{r,f} = \mathit{translateTime2Target}(\varphi_r^*)$
  $u_r^* = \mathit{solveNonInfluentialInputs}(P_1(S_{1,r}, \Phi_{r,f}, T_2, x_{1,r,0}))$
  if isFeasible(P_1)
    return u_r^*
  else
    $T_2 \leftarrow T_2 + 1$
  end
end
Output: Input sequence u^* that steers y in the target set $\mathcal{Y}_f$ by using as few inputs as possible.

The algorithm starts applying a routine called *reduce* to obtain a reduced order system $S_{2,r}$ of S_2 that preserves its input/output behavior. System $S_{2,r}$ has state $x_{2,r}$ and input φ_r obtained from the state x_2 and the input φ of S_2 by removing those state component and inputs that are not affecting the output y. Then, a reduced order model $S_{1,r}$ with state $x_{1,r}$ and input u_r is obtained that preserves its input/output behavior, where the output is φ_r.

The minimum number T_2 of time steps needed for system $S_{2,r}$ to reach $\mathcal{Y}_f$ is determined by running a routine called *MinimumTimeForSpec*. Then, non-influential inputs for system $S_{2,r}$, initialized at $x_{2,r,0}$, are computed with reference to the reachability condition $y(T_2) \in \mathcal{Y}_f$ (problem P_2). The computation of non-influential inputs is achieved by function *solveNonInfluentialInputs*. The resulting optimal control input sequence φ_r^* is then used to formulate an intermediate finite horizon specification on the output of $S_{1,r}$ (function *translateTime2Target*). Note that the detection of non-influential inputs at this stage allows to relax the constraints on the output of $S_{1,r}$ (or, equivalently, to enlarge the target set Φ_f), making it more likely to obtain a feasible problem for S_1. The same procedure for the computation of non-influential inputs is then repeated for system $S_{1,r}$. In principle, the resulting problem might be unfeasible, and, if that is the case, Algorithm 1 restarts the computations in the loop with an augmented time horizon. Note also that if the algorithm stops at the first iteration, we are

guaranteed that the solution u^* returned is also of minimum length. We refer the reader to [42] for the description of functions *MinimumTimeForSpec*, *solveNonInfluentialInputs*, and *translateTime2Target*, and to Deliverable 1.2 for the model reduction procedure implemented in function *reduce*.

2.2.2 A parallel verification scheme for constraint coupled systems

Let us consider an MLD system described by

$$\begin{aligned} x(k+1) &= Ax(k) + B_u u(k) + B_\delta \delta(k) + B_z z(k) + B_{aff} \\ y(k) &= Cx(k) + D_u u(k) + D_\delta \delta(k) + D_z z(k) + D_{aff} \\ E_x x(k) &+ E_u u(k) + E_\delta \delta(k) + E_z z(k) \leq E_{aff} \end{aligned} \tag{22}$$

where $x \in \mathbb{R}^{n_c} \times \{0,\ 1\}^{n_l}$ is the state composed of both continuous and binary variables, $u \in \mathbb{R}^{m_c} \times \{0,\ 1\}^{m_l}$ is the input vector comprising a continuous and a discrete component, and $\delta \in \{0,\ 1\}^{r_l}$ and $z \in \mathbb{R}^{r_c}$ are binary and continuous-valued auxiliary variables.

Suppose that we need to verify if there exists some input sequence $u(0), u(1), \ldots, u(T)$ such that the output y reaches some polyhedral target set at time T: $y(T) \in \mathcal{Y}_f = \{H_a y \leq H_b\}$.

If we set $\theta = \left[u(0)^\top, \ldots, u(T)^\top, z(0)^\top, \ldots, z(T)^\top, \delta(0)^\top, \ldots, \delta(T)^\top\right]^\top$, then, the problem becomes finding $\theta \in (\mathbb{R}^{m_c} \times \{0,\ 1\}^{m_l})^{T+1} \times (\mathbb{R}^{r_c})^{T+1} \times (\{0,\ 1\}^{r_l})^{T+1}$ such that $G_1\theta \leq G_2 + G_3 x(0)$, where G_1, G_2, and G_3 are appropriately defined matrices obtained from the system dynamics and the target constraint. A similar formulation can be actually derived if the output is subject to linear constraints along the time horizon $[0, T]$ (see Section 2.2.1). If we set

$$\Theta = \{\theta \in (\mathbb{R}^{m_c} \times \{0,\ 1\}^{m_l})^{T+1} \times (\mathbb{R}^{r_c})^{T+1} \times (\{0,\ 1\}^{r_l})^{T+1} : G_1\theta \leq G_2 + G_3 x(0)\}$$

then, the problem reduces to determining $\theta \in \Theta$, which is a mixed integer linear feasibility problem whose combinatorial complexity is determined by the number of discrete decision variables. Suppose now that we have m MLD systems of the form (22), each one of them possibly subject to some (linear) output specification, and that there is a coupling constraint among their decision vectors θ_i, $i = 1, \ldots, m$, of the form:

$$\sum_{i=1}^{m} W_i \theta_i \leq b, \tag{23}$$

with $b \in \mathbb{R}^p$ and W_i, $i = 1, \ldots, m$, matrices of appropriate dimension, originating, e.g., from some share resources with finite capacity (budget constraint). Then, the verification problem for the m MPD systems is coupled and can be formulated as follows

$$\text{Find } \theta_i \in \Theta_i,\ i = 1, \ldots, m \text{ such that } \sum_{i=1}^{m} W_i \theta_i \leq b. \tag{24}$$

Problems of the form (24) are generally challenging to solve if the number m of systems is large mainly due to the large number of discrete decision variables. We next describe an iterative method to address this issue by solving m MILP in parallel whose complexity is determined by the number of discrete decision variables in each θ_i vector separately (divide and conquer strategy).

It is worth noticing that the same approach can be applied to address the verification of a monolithic and high dimensional MLD system, if it can be decomposed in m smaller MLD systems with a coupling constraint of the form (23). The rest of the section is extracted from [18], with slightly modified notations.

In order to solve the feasibility program (24), we look for a solution to the optimization problem

$$\begin{aligned} \min_{\theta_1,\ldots,\theta_m} \quad & \sum_{i=1}^{m} c_i'\theta_i \\ \text{subject to:} \quad & \sum_{i=1}^{m} W_i\theta_i \leq b \\ & \theta_i \in \Theta_i,\ i = 1,\ldots,m \end{aligned} \tag{25}$$

where a linear separable cost function is introduced. Note that a feasible solution for (25) is a solution to the feasibility problem (24).

Despite the advances in numerical methods for integer optimization, when the number m of subsystems is large, the presence of discrete decision variables makes the optimization problem hard to solve, and calls for some decomposition into lower scale MILPs, as suggested in [44].

A common practice to handle problems of the form of (25) consists in first dualizing the coupling constraint introducing a vector $\lambda \in \mathbb{R}^p$ of p Lagrange multipliers and solving the dual program

$$\max_{\lambda \geq 0} \ -\lambda' b + \sum_{i=1}^{m} \min_{\theta_i \in \Theta_i} (c_i' + \lambda' W_i)\theta_i, \tag{26}$$

to obtain $\lambda^\star$, and then constructing a primal solution $\theta(\lambda^\star) = [\theta_1(\lambda^\star)' \ \cdots \ \theta_m(\lambda^\star)']'$ by solving m MILPs given by:

$$\theta_i(\lambda) \in \arg \min_{\theta_i \in \text{vert}(\Theta_i)} (c_i' + \lambda' W_i)\theta_i, \tag{27}$$

where the search within the closed constraint polyhedral set Θ_i can be confined to its set of vertices $\text{vert}(\Theta_i)$ since the cost function is linear. Unfortunately, while this procedure guarantees $\theta(\lambda^\star)$ to satisfy the local constraints since $\theta_i(\lambda^\star) \in \Theta_i$ for all $i = 1,\ldots,m$, it does not guarantee the satisfaction of the coupling constraint.

A way to enforce the satisfaction of the coupling constraint is to follow the approach in [40], where the dual program (26) is solved via a particular iterative methodology, namely, the subgradient algorithm. At each iteration of the subgradient algorithm a tentative primal solution is generated by every subsystem. By appropriately averaging the tentative solutions across iterations (see [40, pag. 117]), one can obtain a solution that satisfies the joint constraint. However, when discrete decision variables are present, such solution does not necessarily satisfy also the local constraints. Specifically, letting $\text{conv}(\Theta_i)$ denote the convex hull of Θ_i, $i = 1, \dots, m$, if we apply the above procedure to (25), we obtain an optimal solution $\theta^\star_{\text{LP}}$ to the following Linear Program (LP):

$$\begin{aligned} \min_{\theta_1,\dots,\theta_m} \quad & \sum_{i=1}^{m} c_i' \theta_i \\ \text{subject to:} \quad & \sum_{i=1}^{m} W_i \theta_i \leq b \\ & \theta_i \in \text{conv}(\Theta_i), \ i = 1, \dots, m. \end{aligned} \tag{28}$$

This fact is true because the dual of the convexified problem (28) coincides with the dual of (25) and is given by (26) (see [20] for a proof). Clearly $\theta^\star_{\text{LP}} \in \text{conv}(\Theta_1) \times \cdots \times \text{conv}(\Theta_m)$ does not necessarily imply that $\theta^\star_{\text{LP}} \in \Theta_1 \times \Theta_2 \times \cdots \times \Theta_m$. Therefore the solution $\theta^\star_{\text{LP}}$ recovered using [40] satisfies the coupling constraint but not necessarily the local constraints. An alternative approach for finding an optimal solution to the primal-dual pair (28)-(26) is to exploit the column generation algorithm (see [25]). Even in this case, however, the procedure converges to a solution $\theta^\star_{\text{LP}}$ of (28), but it is not guaranteed that such a solution is feasible for the local constraints in (25).

For these reasons recovery procedures for MILPs are usually composed of two steps: a tentative solution that is not feasible for either the joint constraint or the local ones is first obtained exploiting one of the two procedures described above, and then a problem-specific heuristic is applied to recover a feasible solution for (25), see, e.g., [9, 35].

Problems in the form of (25) have been investigated in [6], where the authors studied the behavior of the duality gap (i.e., the difference between the optimal value of (25) and (26)) showing that it decreases relatively to the optimal value of (25) as the number of agents grows. The same behavior has been observed in [9]. In the recent paper [44], the authors explored the connection between the solutions $\theta^\star_{\text{LP}}$ to the linear program (28) and $x(\lambda^\star)$ recovered via (27) from the solution $\lambda^\star$ to the dual program (26). They proposed a method to recover a primal solution which is feasible for (25) by using the dual optimal solution of a modified primal problem, obtained by tightening the coupling constraint by an appropriate amount.

We now recall those parts of [44] that are relevant for the developments in this work. Let $\rho \in \mathbb{R}^p$ with $\rho \geq 0$ and consider the following pair of primal-dual problems:

$$\begin{aligned} \min_{\theta_1,\ldots,\theta_m} \quad & \sum_{i=1}^{m} c_i' \theta_i \\ \text{subject to:} \quad & \sum_{i=1}^{m} W_i \theta_i \leq b - \rho \\ & \theta_i \in \operatorname{conv}(\Theta_i),\ i = 1, \ldots, m \end{aligned} \tag{29}$$

and

$$\max_{\lambda \geq 0} \ -\lambda'(b-\rho) + \sum_{i=1}^{m} \min_{\theta_i \in \Theta_i} (c_i' + \lambda' W_i)\theta_i. \tag{30}$$

(29) constitutes a tightened version of (28), whereas (30) is the corresponding dual. For all $j = 1, \ldots, p$, let $\tilde{\rho} \in \mathbb{R}^p$ be defined as follows:

$$[\tilde{\rho}]_j = p \max_{i \in \{1,\ldots,m\}} \ \Big\{ \max_{\theta_i \in \Theta_i} [W_i]_j \theta_i - \min_{\theta_i \in \Theta_i} [W_i]_j \theta_i \Big\}, \tag{31}$$

where $[W_i]_j$ denotes the j-th row of W_i and $[\tilde{\rho}]_j$ the j-th entry of $\tilde{\rho}$.

Define $\widetilde{\mathcal{P}}_{\mathrm{LP}}$ and $\widetilde{\mathcal{D}}$ as the primal-dual pair of optimization problems that are given by setting ρ equal to $\tilde{\rho}$ in (29) and (30).

Assumption 1 (Existence and uniqueness, [44])**.**
Problems $\widetilde{\mathcal{P}}_{\mathrm{LP}}$ and $\widetilde{\mathcal{D}}$ have unique solutions $\theta^\star_{\mathrm{LP},\tilde{\rho}}$ and $\lambda^\star_{\tilde{\rho}}$.

Proposition 1 (Theorem 3.1 in [44])**.** *Let $\lambda^\star_{\tilde{\rho}}$ be the solution to $\widetilde{\mathcal{D}}$. Under Assumption 1, we have that any $\theta(\lambda^\star_{\tilde{\rho}})$ satisfying* (27)*, is feasible for* (25)*.*

The proof of Proposition 1 rests on Theorem 2.5 in [44]. Example 2.6 in [44] shows how Theorem 2.5 in [44], and therefore also Proposition 1, strongly depend on the uniqueness part of Assumption 1. Note, however, that in case $\widetilde{\mathcal{P}}_{\mathrm{LP}}$ has multiple solutions, then a small perturbation in its cost coefficients will render its solution unique, thus making Assumption 1 fulfilled again. We refer the reader to [44] for further details.

Inspired by the idea of constraint tightening in [44], we propose Algorithm 2 for the parallel computation in a finite number of iterations of an approximate solution to the optimization problem (25) that is feasible.

Algorithm 2 is a variant of the dual subgradient algorithm. As the standard dual subgradient method, it includes two main steps: step 7 in which a subgradient of the dual objective function is computed by fixing the dual variables and minimizing the Lagrangian with respect to the primal variables, and step 13 which involves a dual update step with step size equal to $\alpha(k)$, and a projection onto the non-negative orthant (in Algorithm 2 $[\,\cdot\,]_+$ denotes the

Algorithm 2 Parallel verification scheme

1: $\lambda(0) = 0$

2: $\bar{s}_i(0) = -\infty$, $i = 1, \dots, m$

3: $\underline{s}_i(0) = +\infty$, $i = 1, \dots, m$

4: $k = 0$

5: Repeat

6: For $i = 1$ to m do

7: $\theta_i(k+1) \leftarrow \arg \min_{\theta_i \in \mathrm{vert}(\Theta_i)} (c_i' + \lambda(k)' W_i)\theta_i$

8: $\bar{s}_i(k+1) = \max\{\bar{s}_i(k), W_i\theta_i(k+1)\}$, $i = 1, \dots, m$

9: $\underline{s}_i(k+1) = \min\{\underline{s}_i(k), W_i\theta_i(k+1)\}$, $i = 1, \dots, m$

10: $\rho_i(k+1) = \bar{s}_i(k+1) - \underline{s}_i(k+1)$, $i = 1, \dots, m$

11: $\rho(k+1) = p \max\{\rho_1(k+1), \dots, \rho_m(k+1)\}$

12: $\lambda(k+1)$

13: $= \Big[\lambda(k) + \alpha(k)\big(\sum_{i=1}^{m} W_i\theta_i(k+1) - b + \rho(k+1)\big)\Big]_+$

14: $k \leftarrow k+1$

15: until $\theta(k) = [\theta_1(k)' \ \cdots \ \theta_m(k)']'$ satisfies the coupling constraint

projection operator onto the p-dimensional non-negative orthant $\mathbb{R}^p_+$). The operators max and min appearing in steps 8, 9, and 11 of Algorithm 2 with arguments in $\mathbb{R}^p$ are meant to be applied component-wise. The sequence $\{\alpha(k)\}_{k\geq 0}$ is chosen so as to satisfy $\lim_{k\to\infty} \alpha(k) = 0$ and $\sum_{k=0}^{\infty} \alpha(k) = \infty$ (e.g., $\alpha(k) = \frac{1}{k}$), as requested in the standard dual subgradient method to achieve asymptotic convergence. Furthermore, in order to guarantee that the solution to step 7 of Algorithm 2 is well-defined, we impose the following assumption on (25):

Assumption 2 (Boundedness)**.** *The polyhedral sets Θ_i, $i = 1, \dots, m$, in problem* (25) *are bounded.*

If $\arg\min_{\theta_i \in \mathrm{vert}(\Theta_i)} (c_i' + \lambda(k)' W_i)\theta_i$ in step 7 is a set of cardinality larger than 1, then, a deterministic tie-break rule is applied to choose a value for $\theta_i(k+1)$.

Algorithm 2 is conceived to be implemented in a parallel scheme where, at each iteration k, every subsystem i updates its local tentative solution $\theta_i(k+1)$ and communicates $A_i x_i(k+1)$ to some central unit that is in charge of the update of the dual variable. The tentative value $\lambda(k+1)$ for the dual variable is then broadcast to all subsystems. Note that subsystems do not need to communicate to the central unit their local constraint set and cost but only their tentative solution $\theta_i(k)$.

The tentative primal solutions $\theta_i(k+1)$, $i = 1, \dots, m$, computed at step 7 are used in

Algorithm 2 by the central unit to determine the amount of tightening $\rho(k+1)$ entering step 13. The value of $\rho(k+1)$ is progressively refined through iterations based only on those values of $\theta_i \in \Theta_i$, $i = 1, \ldots, m$, that are actually considered as candidate primal solutions, and not based on the whole sets Θ_i, $i = 1, \ldots, m$. This reduces conservativeness in the amount of tightening.

A further reduction in the level of conservativeness can be achieved by assigning to $[\rho_i(k+1)]_j$ in step 11 of Algorithm 2 the (less conservative) sum of the p-largest $[\rho_i(k+1)]_j$, for all $j = 1, \ldots, p$. Further discussion is provided after the proof of Proposition 1.

Algorithm 2 terminates when $\theta(k) = [\theta_1(k)' \ \cdots \ \theta_m(k)']'$ satisfies the coupling constraint. As for the initialization of Algorithm 2, $\lambda(0)$ is set equal to 0 so that at iteration $k = 0$ each subsystem i computes its locally optimal solution

$$\theta_i(1) \leftarrow \arg \min_{\theta_i \in \mathrm{vert}(\Theta_i)} c_i' \theta_i.$$

Since $\rho(1) = 0$, if the local solutions $\theta_i(1)$, $i = 1, \ldots, m$, satisfy the coupling constraint (and they hence are optimal for the original problem (25)), then, Algorithm 2 will terminate since λ will remain 0, and the subsystems will stick to their locally optimal solutions.

Before stating the feasibility guarantees of the solution computed by Algorithm 2, we need to introduce some further quantities and assumptions.

Let us consider the sequence $\{\rho(k)\}_{k\geq 1}$, iteratively computed in Algorithm 2 (see step 11), and given by

$$[\rho(k)]_j = p \max_{i \in \{1,\ldots,m\}} \Big\{ \max_{r \leq k} [W_i]_j \theta_i(r) - \min_{r \leq k} [W_i]_j \theta_i(r) \Big\}. \tag{32}$$

Due to Assumption 2, for any $i = 1, \ldots, m$, $\mathrm{conv}(\Theta_i)$ is a bounded polyhedron. If it is also non-empty, then $\mathrm{vert}(\Theta_i)$ is a non-empty finite set (see Corollaries 2.1 and 2.2 together with Theorem 2.3 in [11, Chapter 2]). As a consequence, the sequence $\{\rho(k)\}_{k\geq 1}$ converges in finite-time to some $\bar{\rho}$ since it takes values in a finite set and is (component-wise) monotonically non-decreasing. Note that the limiting value $\bar{\rho}$ for $\{\rho(k)\}_{k\geq 1}$ satisfies $\bar{\rho} \leq \tilde{\rho}$ and $\bar{\gamma} \leq \tilde{\gamma}$ where $\tilde{\rho}$ and $\tilde{\gamma}$ are defined in (31).

Define $\overline{\mathcal{P}}_{\mathrm{LP}}$ and $\overline{\mathcal{D}}$ as the primal-dual pair of optimization problems that are given by setting ρ equal to $\bar{\rho}$ in (29) and (30).

In order to state the feasibility properties of Algorithm 2, besides Assumption 2, the following assumption is needed.

Assumption 3 (Existence and uniqueness)**.**
Problems $\overline{\mathcal{P}}_{\mathrm{LP}}$ and $\overline{\mathcal{D}}$ have unique solutions $\bar{\theta}^\star_{\mathrm{LP}}$ and $\bar{\lambda}^\star$.

Note that Assumption 3 is similar to Assumptions 1. However, owing to the fact that $\bar{\rho} \leq \tilde{\rho}$, imposing Assumption 3 in place of Assumption 1 makes Algorithm 2 applicable to a larger class of problems with respect to the approach in [44].

We are now in a position to state the finite-time feasibility result.

Theorem 1 (Finite-time feasibility)**.** *Under Assumptions 2 and 3, there exists a finite iteration index K such that, for all $k \geq K$, $\theta(k) = [\theta_1(k)' \ \cdots \ \theta_m(k)']'$, where $\theta_i(k)$, $i = 1, \ldots, m$, are computed by Algorithm 2, is a feasible solution for problem* (25)*, i.e., $\sum_{i=1}^{m} W_i \theta_i(k) \leq b$, $k \geq K$ and $\theta_i(k) \in \Theta_i$, $i = 1, \ldots, m$.*

Note that if $\overline{\mathcal{P}}_{\mathrm{LP}}$ is not feasible for the resulting $\bar{\rho}$, then $\sum_{i=1}^{m} W_i \theta_i(k+1) - b + \rho(k+1)$ is bounded below by some positive constant for a sufficiently high k given that $\rho(k+1)$ converges to $\bar{\rho}$. Since $\sum_{k=0}^{\infty} \alpha(k) = \infty$, step 13 of Algorithm 2 will then produce a $\{\lambda(k)\}_{k\geq 0}$ sequence diverging towards $+\infty$. Therefore, observing a component of $\lambda(k)$ which diverges as k increases is an indication that the existence part of Assumption 3 is not satisfied.

The proof of Theorem 1 is based on some preliminary results that are presented next.

Proposition 2 (Dual asymptotic convergence)**.** *Under Assumptions 2 and 3, the Lagrange multiplier sequence $\{\lambda(k)\}_{k\geq 0}$ generated by Algorithm 2 converges to an optimal solution of $\overline{\mathcal{D}}$.*

Proof. As discussed after equation (32), there exists a $K \in \mathbb{N}$ such that for all $k \geq K$ we have that the tightening coefficient $\rho(k)$ computed in Algorithm 2 becomes constant and equal to $\bar{\rho}$. Therefore, for any $k \geq K$, Algorithm 2 reduces to the following two steps

$$\theta_i(k+1) \in \arg \min_{\theta_i \in \mathrm{vert}(\Theta_i)} (c_i' + \lambda(k)' W_i) \theta_i \tag{33}$$

$$\lambda(k+1) = \left[\lambda(k) + \alpha(k) \left(\sum_{i=1}^{m} W_i \theta_i(k+1) - b + \bar{\rho} \right) \right]_+ \tag{34}$$

which constitute a gradient ascent iteration for $\overline{\mathcal{D}}$. According to [10], the sequence $\{\lambda(k)\}_{k\geq 0}$ generated by the iterative procedure (33)-(34) is guaranteed to converge to the (unique under Assumption 3) optimal solution of $\overline{\mathcal{D}}$.

Note that this result requires only uniqueness of the optimal solution of $\overline{\mathcal{D}}$. Uniqueness of the optimal solution to $\overline{\mathcal{P}}_{\mathrm{LP}}$ is not necessary. □

Lemma 2.8 (Robustness against cost perturbation)**.** *Let P be a non-empty bounded polyhedron. Consider the linear program $\min_{\theta \in P} (c' + \delta')\theta$, where δ is a perturbation in the cost coefficients. Define the set of optimal solutions as $\Theta(\delta)$. There always exists an $\varepsilon > 0$ such that for all δ satisfying $\|\delta\| < \varepsilon$, we have $\Theta(\delta) \subseteq \Theta(0)$.*

Proof. Let $u(\delta) = \min_{\theta \in P}(c' + \delta')\theta$. Since P is a bounded polyhedron, the minimum is always attained and $u(\delta)$ is finite for any value of δ. The set $\Theta(\delta)$ can be defined as

$$\Theta(\delta) = \{\theta \in P : (c' + \delta')\theta \leq u(\delta)\}, \tag{35}$$

which is a non-empty polyhedron. As such, it can be described as the convex hull of its vertices (see Theorem 2.9 in [11, Chapter 2]), which are also vertices of P (Theorem 2.7 in [11, Chapter 2]).

Let $V = \text{vert}(P)$ and $V_\delta = \text{vert}(\Theta(\delta)) \subseteq V$. Consider $\delta = 0$. If $V_0 = V$, then, given the fact that, for any δ, $\Theta(\delta)$ is the convex hull of V_δ and $V_\delta \subseteq V = V_0$, we have trivially that $\Theta(\delta) \subseteq \Theta(0)$, for any δ. Suppose now that $V_0 \subset V$. For any choice of $\theta^\star \in V_0$ and $\theta \in V \setminus V_0$, we have that $c'\theta^\star < c'\theta$, or equivalently $c'(\theta^\star - \theta) < 0$. Pick

$$\varepsilon = \min_{\substack{\theta^\star \in V_0 \\ \theta \in V \setminus V_0}} -\frac{c'(\theta^\star - \theta)}{\|\theta^\star - \theta\|} \tag{36}$$

and let $(\bar{\theta}^\star, \bar{\theta})$ be the corresponding minimizer. By construction, (36) is well defined since $\bar{\theta}^\star$ is different from $\bar{\theta}$. Since $c'(\theta^\star - \theta) < 0$ for any $\theta^\star \in V_0$ and $\theta \in V \setminus V_0$, we have that $\varepsilon > 0$. Moreover, for any $\theta^\star \in V_0$ and $\theta \in V \setminus V_0$, if δ satisfies $\|\delta\| < \varepsilon$, then

$$\begin{aligned}
(c' + \delta')(\theta^\star - \theta) = c'(\theta^\star - \theta) + \delta'(\theta^\star - \theta) &\leq c'(\theta^\star - \theta) + \|\delta\| \, \|\theta^\star - \theta\| \\
&< c'(\theta^\star - \theta) + \varepsilon \, \|\theta^\star - \theta\| \\
&\leq c'(\theta^\star - \theta) + \left(-\frac{c'(\theta^\star - \theta)}{\|\theta^\star - \theta\|}\right) \|\theta^\star - \theta\| \\
&= c'(\theta^\star - \theta) - c'(\theta^\star - \theta) = 0,
\end{aligned} \tag{37}$$

where the first inequality is given by the fact that $u'v \leq |u'v|$ together with the Cauchy–Schwarz inequality $|u'v| \leq \|u\| \, \|v\|$, the second inequality is due to δ satisfying $\|\delta\| < \varepsilon$, and the third inequality is given by the definition of ε in (36).
By (35) and the definition of $u(\delta)$, for any point θ_δ in the set V_δ, we have that $(c' + \delta')\theta_\delta \leq (c' + \delta')\theta$, for all $\theta \in V$, and therefore $(c' + \delta')\theta_\delta \leq (c' + \delta')\theta^\star$ for any $\theta^\star \in V_0 \subset V$. By (37), whenever $\|\delta\| < \varepsilon$, we have that $(c' + \delta')\theta^\star < (c' + \delta')\theta$ for any choice of $\theta^\star \in V_0$ and $\theta \in V \setminus V_0$, therefore $(c' + \delta')\theta_\delta < (c' + \delta')\theta$ for any $\theta \in V \setminus V_0$. Since the inequality is strict, we have that $\theta_\delta \notin V \setminus V_0$, which implies $\theta_\delta \in V_0$. Since this holds for any $\theta_\delta \in V_\delta$, we have that $V_\delta \subseteq V_0$. Finally, given the fact that, for any δ, $\Theta(\delta)$ is the convex hull of V_δ and $V_\delta \subseteq V_0$, we have $\Theta(\delta) \subseteq \Theta(0)$, thus concluding the proof. □

Exploiting Lemma 2.8, we shall show next that each $\{\theta_i(k)\}_{k \geq 1}$ sequence, $i = 1, \ldots, m$, converges in finite-time to some set. Note that, for the subsequent result, only uniqueness of the optimal solution of $\overline{\mathcal{D}}$ is required.

Proposition 3 (Primal finite-time set convergence)**.** *Under Assumptions 2 and 3, there exists a finite K such that for all $i = 1, \dots, m$ the tentative primal solution $\theta_i(k)$ generated by Algorithm 2 satisfies*

$$\theta_i(k) \in \arg \min_{\theta_i \in \text{vert}(\Theta_i)} (c_i' + \bar{\lambda}^{\star\prime} W_i)\theta_i, \quad k \geq K, \tag{38}$$

where $\bar{\lambda}^\star$ is the limit value of the Lagrange multiplier sequence $\{\lambda(k)\}_{k\geq 0}$.

Proof. Consider subsystem i, with $i \in \{1, \dots, m\}$. We can characterize the solution $\theta_i(k)$ in step 7 of Algorithm 2 by performing the minimization over $\text{conv}(\Theta_i)$ instead of $\text{vert}(\Theta_i)$ since the problem is linear and by enlarging the set $\text{vert}(\Theta_i)$ to $\text{conv}(\Theta_i)$ we still obtain all minimizers that belong to $\text{vert}(\Theta_i)$. Adding and subtracting $\bar{\lambda}^{\star\prime} W_i \theta_i$ to the cost, we then obtain

$$\theta_i(k) \in \arg \min_{\theta_i \in \text{conv}(\Theta_i)} (c_i' + \bar{\lambda}^{\star\prime} W_i + (\lambda(k-1) - \bar{\lambda}^\star)' W_i)\theta_i. \tag{39}$$

Set $\delta_i(k-1)' = (\lambda(k-1) - \bar{\lambda}^\star)' W_i$, and let $\Theta_i(\delta_i(k-1))$ be the set of minimizers of (39) as a function of $\delta_i(k-1)$. By Lemma 2.8, we know that there exists an $\varepsilon_i > 0$ such that if $\|\delta_i(k-1)\| < \varepsilon_i$, then $\Theta_i(\delta_i(k-1)) \subseteq \Theta_i(0)$.

Since, by Proposition 2, the sequence $\{\lambda(k)\}_{k\geq 0}$ generated by Algorithm 2 converges to $\bar{\lambda}^\star$, by definition of limit, we know that there exists a K_i such that $\|\delta_i(k-1)\| = \left\|(\lambda(k-1) - \bar{\lambda}^\star)' W_i\right\| < \varepsilon_i$ for all $k \geq K_i$. Therefore, for every $k \geq K = \max\{K_1, \dots, K_m\}$, we have that $\theta_i(k) \in \Theta_i(0) = \arg\min_{\theta_i \in \text{conv}(\Theta_i)} (c_i' + \bar{\lambda}^{\star\prime} W_i)\theta_i$, $i = 1, \dots, m$. This property jointly with the fact that $\theta_i(k) \in \text{vert}(\Theta_i)$, $i = 1, \dots, m$, leads to (38), thus concluding the proof. □

We can now finally prove Theorem 1.

Proof of Theorem 1. Theorem 2.5 of [44] establishes a relation between the solution $\bar{\theta}^\star_{\text{LP}}$ of $\overline{\mathcal{P}}_{\text{LP}}$ and the one recovered in (27) from the optimal solution $\bar{\lambda}^\star$ of the dual optimization problem $\overline{\mathcal{D}}$. Specifically, it states that there exists a set of indices $I \subseteq \{1, \dots, m\}$ of cardinality at least $m - p$, such that $[\bar{\theta}^\star_{\text{LP}}]^{(i)} = \theta_i(\bar{\lambda}^\star)$ for all $i \in I$, where $[\bar{\theta}^\star_{\text{LP}}]^{(i)}$ is the subvector of $\bar{\theta}^\star_{\text{LP}}$ corresponding to the i-th subsystem. Therefore, following the proof of Theorem 3.1 in [44], we have that

$$\begin{aligned}
\sum_{i=1}^{m} W_i \theta_i(\bar{\lambda}^\star) &= \sum_{i \in I} W_i \theta_i(\bar{\lambda}^\star) + \sum_{i \in I^c} W_i \theta_i(\bar{\lambda}^\star) \\
&= \sum_{i \in I} W_i [\bar{\theta}^\star_{\text{LP}}]^{(i)} + \sum_{i \in I^c} W_i \theta_i(\bar{\lambda}^\star) \\
&= \sum_{i=1}^{m} W_i [\bar{\theta}^\star_{\text{LP}}]^{(i)} + \sum_{i \in I^c} W_i \left(\theta_i(\bar{\lambda}^\star) - [\bar{\theta}^\star_{\text{LP}}]^{(i)}\right)
\end{aligned}$$

$$\leq b - \bar{\rho} + p \max_{i=1,\dots,m} \{W_i \theta_i(\bar{\lambda}^\star) - W_i[\bar{\theta}^\star_{\mathrm{LP}}]^{(i)}\}, \tag{40}$$

where $I^c = \{1, \dots, m\} \setminus I$, and $b - \bar{\rho}$ constitutes an upper bound for $\sum_{i=1}^m W_i[\bar{\theta}^\star_{\mathrm{LP}}]^{(i)}$ given that $\bar{\theta}^\star_{\mathrm{LP}}$ is feasible for $\overline{\mathcal{P}}_{\mathrm{LP}}$.

According to [40, p. 117], the component $[\theta^\star_{\mathrm{LP}}]^{(i)}$ of the (unique, under Assumption 3) solution $\bar{\theta}^\star_{\mathrm{LP}}$ to $\overline{\mathcal{P}}_{\mathrm{LP}}$ is the limit point of the sequence $\{\tilde{\theta}_i(k)\}_{k\geq 1}$, defined as

$$\tilde{\theta}_i(k) = \frac{\sum_{r=1}^{k-1} \alpha(r)\theta_i(r+1)}{\sum_{r=1}^{k-1} \alpha(r)}.$$

By linearity, for all $k \geq 0$, we have that

$$W_i\tilde{\theta}_i(k) = \frac{\sum_{r=1}^{k-1} \alpha(r) W_i\theta_i(r+1)}{\sum_{r=1}^{k-1} \alpha(r)} \geq \min_{r\leq k} W_i\theta_i(r) = \underline{s}_i(k) \geq \underline{s}_i,$$

where the first inequality is due to the fact that all $\alpha(k)$ are positive and the second equality follows from step 9 of Algorithm 2. In the final inequality, $\underline{s}_i(k)$ is lower bounded by $\underline{s}_i$, that denotes the limiting value of the non-increasing finite-valued sequence $\{\underline{s}_i(k)\}_{k\geq 0}$. Note that all inequalities have to be intended component-wise. By taking the limit for $k \to \infty$, we also have that

$$W_i[\bar{\theta}^\star_{\mathrm{LP}}]^{(i)} \geq \underline{s}_i. \tag{41}$$

By Proposition 3, there exists a finite iteration index K such that $\theta_i(k)$ satisfies (38). Since (40) holds for any choice of $\theta_i(\bar{\lambda}^\star)$ which minimizes $(c_i' + \bar{\lambda}^{\star\prime} W_i)\theta_i$ over $\mathrm{vert}(\Theta_i)$, if $k \geq K$, then we can choose $\theta_i(\bar{\lambda}^\star) = \theta_i(k)$. Therefore, for all $k \geq K$, (40) becomes

$$\begin{aligned}
\sum_{i=1}^m W_i\theta_i(k) &\leq b - \bar{\rho} + p \max_{i=1,\dots,m} \{W_i\theta_i(k) - W_i[\theta^\star_{\mathrm{LP}}]^{(i)}\} \\
&\leq b - \bar{\rho} + p \max_{i=1,\dots,m} \left\{ \max_{r\leq k} W_i\theta_i(r) - W_i[\theta^\star_{\mathrm{LP}}]^{(i)} \right\} \\
&= b - \bar{\rho} + p \max_{i=1,\dots,m} \left\{ \bar{s}_i(k) - W_i[\theta^\star_{\mathrm{LP}}]^{(i)} \right\} \\
&\leq b - \bar{\rho} + p \max_{i=1,\dots,m} \{\bar{s}_i - \underline{s}_i\} \\
&= b,
\end{aligned} \tag{42}$$

where the second inequality is obtained by taking the maximum up to k, the first equality is due to step 8 of Algorithm 2, the third inequality is due to the fact that $\bar{s}_i$ is the limiting value of the non-decreasing finite-valued sequence $\{\bar{s}_i(k)\}_{k\geq 1}$ together with (41), and the last equality comes from the definition of $\rho(k) = p \max\{\rho_1(k), \dots, \rho_m(k)\}$ where $\rho_i(k) = \bar{s}_i(k) - \underline{s}_i(k)$.

From (42) we have that, for any $k \geq K$, the iterates $\theta_i(k)$, $i = 1, \dots, m$, generated by Algorithm 2 provide a feasible solution for (25), thus concluding the proof. □

As previously mentioned, we can make Algorithm 2 less conservative by assigning to $[\rho_i(k+1)]_j$ in step 11 of Algorithm 2 the sum of the p-largest $[\rho_i(k+1)]_j$, for all $j = 1, \ldots, p$. To adapt the proof, it suffice to note that the j-th component of $p \max_{i=1,\ldots,m}\{W_i\theta_i(\bar{\lambda}^\star) - W_i[\bar{\theta}^\star_{\mathrm{LP}}]^{(i)}\}$ in (40) can be substituted with the sum of the p-largest values in the set $\{[W_i]_j\theta_i(\bar{\lambda}^\star) - [W_i]_j[\bar{\theta}^\star_{\mathrm{LP}}]^{(i)}\}_{i=1}^m$, and the following derivations will remain unchanged.

3 Incremental verification in interaction with online controller adaptation

3.1 Incremental Computation of Reachable Sets for Anytime Verification

In this section, we propose a procedure to formally verify the safety of autonomous vehicles online, i.e., during operation, that considers the uniqueness of each traffic situation. A challenging aspect of online verification is the varying number of surrounding traffic participants, which causes significant variations in computational demand. To guarantee timely safe motion plans, we propose an anytime approach that provides rapid conservative verification results based on coarse model abstractions, which are refined continually if computation time is available. Reachability analysis, which over-approximates all possible behaviors of other traffic participants, is performed for each abstraction. The interested reader is referred to [22], where the following text was originally published.

3.1.1 Preliminaries

In this subsection, we introduce some general notation and the concepts of reachable and occupancy sets. Please note that throughout this section, we assume that all safety-relevant traffic participants are detected by the sensors of the ego-vehicle.

Notation Let $\mathbb{R}^n$ represent the n-dimensional Euclidean space. Given an n-dimensional vector a of any set $\mathcal{A}$ or a list a of length $|a| = n$, a_i represents its i^{th} component or element for $i \in \{1, \dots, n\}$, and $\mathcal{P}(\mathcal{A})$ denotes the power set of $\mathcal{A}$, i.e., the set of all subsets of $\mathcal{A}$. The Minkowski addition of two sets $\mathcal{A}$ and $\mathcal{B}$ is defined by $\mathcal{A} \oplus \mathcal{B} = \{a + b \mid a \in \mathcal{A},\ b \in \mathcal{B}\}$.

The set of Booleans $\mathbb{B}$ comprises two elements, i.e., true $\top$ and false $\bot$. Let $\wedge$ and $\vee$ denote logical conjunction and disjunction, respectively. The logical equality and nonequality are denoted respectively by $\equiv$ and $\not\equiv$.

The sensor measurements are updated only at discrete time steps t_k for $k \in \mathbb{Z}_{\geq 0}$. To simplify parallelization, we consider only a fixed step size, i.e., $t_{k+1} - t_k = \Delta t$, as shown in Fig. 13. Moreover, the constant receding prediction horizon is denoted by $h \in \mathbb{Z}_{>0}$, which corresponds to the number of time intervals evaluated by prediction at each initial time step t_k.

Reachable Set Typically, an exact mathematical model M^{exact} of another traffic participant is not known by the ego-vehicle unless transmitted via vehicle-to-vehicle communication.

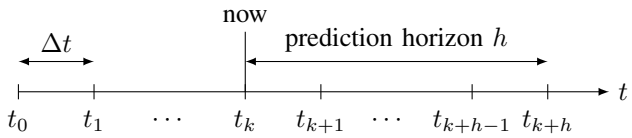


Figure 13: Fixed receding prediction horizon h at current initial time step t_k, and constant step size Δt.

Therefore, we use nondeterministic dynamic models that capture real physical behavior. All considered models of different complexity for another traffic participant are contained in a list M of length $|M|$.

The nominal behavior of model M_i for $i \in \{1, \dots, |M|\}$ is given by the following ordinary differential equation:

$$\dot{x}^{(i)}(t) = f^{(i)}\big(x^{(i)}(t), u^{(i)}(t)\big), \tag{43}$$

where $x^{(i)}(t) \in \mathbb{R}^{n_x^{(i)}}$ and $u^{(i)}(t) \in \mathbb{R}^{n_u^{(i)}}$ denote the state and input, respectively. The inputs, e.g., steering rate and acceleration, are uncertain but bounded by the set $\mathcal{U}^{(i)} \subset \mathbb{R}^{n_u^{(i)}}$ for all times, i.e., $\forall t\colon u^{(i)}(t) \in \mathcal{U}^{(i)}$ which is denoted by $u^{(i)}(\cdot) \in \mathcal{U}^{(i)}$. Based on new sensor measurements, the uncertain initial state set $\mathcal{X}_0^{(i)}(t_k) \subset \mathbb{R}^{n_x^{(i)}}$ is updated at each time step t_k, and the corresponding initial state is $x_0^{(i)}(t_k) \in \mathcal{X}_0^{(i)}(t_k)$. The solution of (43) beginning from initial time step t_k is denoted by $\xi^{(i)}\big(t, x_0^{(i)}(t_k), u^{(i)}(\cdot)\big)$.

The exact reachable set, i.e., the set of states $x^{(i)}$ that are reachable, based on model M_i, initial time step t_k, and prediction time interval $[t_{k+j-1},\ t_{k+j}]$ is

$$\begin{aligned}\mathcal{R}^e(M_i, t_k, t_{k+j}) = \big\{\xi^{(i)}\big(t, x_0^{(i)}(t_k), u^{(i)}(\cdot)\big) \;\big|\; t \in [t_{k+j-1},\ t_{k+j}],\\ x_0^{(i)}(t_k) \in \mathcal{X}_0^{(i)}(t_k), u^{(i)}(\cdot) \in \mathcal{U}^{(i)}\big\},\end{aligned} \tag{44}$$

where $i \in \{1, \dots, |M|\}$ and $j \in \{1, \dots, h\}$. Typically, (44) cannot be computed exactly [29]. Thus, we use over-approximations $\mathcal{R} \supseteq \mathcal{R}^e$ and want $\mathcal{R}$ to enclose $\mathcal{R}^e$ as tightly as possible. In the following, we assume that tight over-approximations are provided for all models.

Model M_i is called an abstraction of the unknown model M^{exact} of another traffic participant if the exact reachable set of M_i over-approximates the set of M^{exact}, i.e., for all j and k it holds that $\mathcal{R}^e(M^{\text{exact}}, t_k, t_{k+j}) \subseteq \mathcal{R}^e(M_i, t_k, t_{k+j})$. In this work, we assume all models M_i are abstractions, i.e., conformant with the real physical system. If this assumption is invalid, more uncertainty must be added to the nondeterministic model abstractions, as done in reachset conformance [37, 38].

Occupancy Set We introduce the mapping

$$\Pi\big(x^{(i)}\big) : \mathbb{R}^{n_x^{(i)}} \to \mathcal{P}\big(\mathbb{R}^2\big),$$

which projects the state $x^{(i)}$ of a traffic participant to its set of occupied X-Y-positions. For a given set $\mathcal{A}$, the projection is applied element-wise, i.e., $\Pi(\mathcal{A}) = \{\Pi(a) \mid a \in \mathcal{A}\}$. The predicted occupancy set of another traffic participant based on model M_i, initial time step t_k, and prediction interval $[t_{k+j-1},\ t_{k+j}]$ is denoted by

$$\mathcal{O}_j^k(M_i) = \Pi\big(\mathcal{R}(M_i, t_k, t_{k+j})\big), \tag{45}$$

i.e., given by projecting the reachable set $\mathcal{R}$ to the two-dimensional set of occupied X-Y-positions. The relation

$$\mathcal{O}_j^k(M^{\text{exact}}) \subseteq \bigcap_{i=1}^{|M|} \mathcal{O}_j^k(M_i) \tag{46}$$

allows us to predict the occupancy set of another traffic participant efficiently by intersecting the occupancies of $|M|$ different abstractions [3, Prop. 5.1]. Thus, the over-approximation becomes tighter each time a new model is added.

Similar to (45), the occupancy set of the ego-vehicle based on the known reference trajectory at time step t_k and prediction interval $[t_{k+j-1},\ t_{k+j}]$ is denoted by $\mathcal{E}_j^k$. The uncertainties due to a non-perfect tracking controller and the dimensions of the ego-vehicle are included in the set $\mathcal{E}_j^k$ [3].

Example 1. *A very simple model M_1 can be obtained by allowing infinite acceleration and assuming maximum velocity $v_{\max}$. Under this model, it is possible to compute the projected exact reachable set $\Pi(\mathcal{R}^e)$ of a point mass as a circular disk with radius $r = (t_{k+j} - t_k)v_{\max}$ corresponding to the prediction interval $[t_{k+j-1},\ t_{k+j}]$. A simple over-approximation $\Pi(\mathcal{R})$ is a square with length $2r$. Finally, to obtain the occupancy set of the considered traffic participant, the vehicle dimensions must be added via Minkowski addition.* □

3.1.2 Safety Verification of Autonomous Vehicles

In this subsection, we provide an overview of our formal safety verification procedure that uses set-based prediction of other traffic participants [5, 28]. In addition, the concept of fail-safe motion planning is presented [32].

Set-based Formal Verification Our formal verification method in Algorithm 3 is executed in parallel for each surrounding traffic participant and has two return values. The first output of Algorithm 3 is Boolean true $\top$ if there exists a possible collision for the ego-vehicle with the considered traffic participant, otherwise Boolean false $\bot$. To this end, we introduce the function `any`, which returns $\top$ if any element of the considered input vector $c \in \mathbb{B}^h$ is $\top$, otherwise $\bot$. At time step t_k, the list of occupancies of another traffic participant

and the ego-vehicle for the prediction horizon h are denoted by $\mathcal{O}^k = \left[\mathcal{O}_1^k, \mathcal{O}_2^k, \ldots, \mathcal{O}_h^k\right]$ and $\mathcal{E}^k = \left[\mathcal{E}_1^k, \mathcal{E}_2^k, \ldots, \mathcal{E}_h^k\right]$, respectively. The list $\mathcal{O}^k$ is the second output of Algorithm 3.

Algorithm 3 Standard Safety Verification

1: **function** standardVerification($\mathcal{E}^k, M, k, h$)
2: updateParameters( )
3: **for all** $j \in \{1, \ldots, h\}$ **do**
4: $\mathcal{O}_j^k \leftarrow \bigcap_{i=1}^{|M|} \mathcal{O}_j^k(M_i)$
5: $c_j \leftarrow$ checkCollision$\left(\mathcal{O}_j^k, \mathcal{E}_j^k\right)$
6: **return** any(c), $\mathcal{O}^k$

Throughout this section, we use the following three model abstractions for other traffic participants:

- an infinite-acceleration-based model M_1 (Section 3.1.1);
- a finite-acceleration-based model M_2 [5]; and
- a lane-following model M_3 [5].

Although $\mathcal{O}_j^k(M_2) \subseteq \mathcal{O}_j^k(M_1)$ holds for arbitrary k and j, we do not require the occupancy set of a model to be the subset of another one or vice versa, e.g., $\mathcal{O}_j^k(M_2) \nsubseteq \mathcal{O}_j^k(M_3)$ and $\mathcal{O}_j^k(M_3) \nsubseteq \mathcal{O}_j^k(M_2)$ generally hold.

The parameters of these models are primarily based on traffic rules and physical constraints. For example, M_1 and M_2 assume that another traffic participant does not exceed maximum velocity $v_{\max}$, e.g., given by an exact or relaxed speed limit. Moreover, it is checked whether the other traffic participant obeys traffic rules, such as staying in their own lane. If a violation is detected by the ego-vehicle, the corresponding parameter is adapted or removed, e.g., by increasing the individual speed limit or disabling the assumption that the other traffic participant will follow lanes in the future. Otherwise, the abstractions become nonconformant with the real system. The described parameter updating is handled by the function updateParameters, which is called first in Algorithm 3.

Second, our set-based verification procedure predicts the occupancies of other traffic participants at time step t_k for all h consecutive prediction intervals via reachability analysis, as shown in Fig. 14. In line 4 of Algorithm 3, the overall occupancy set $\mathcal{O}_j^k$ for another traffic participant at time step t_k and prediction interval $[t_{k+j-1},\ t_{k+j}]$ is computed. Based on (46), in order to reduce the over-approximation error, the occupancy sets $\mathcal{O}_j^k(M_i)$ of all $|M|$ models are intersected.

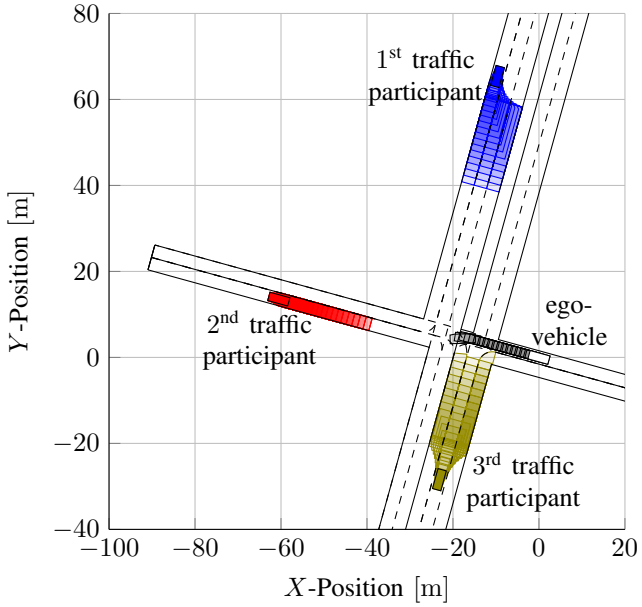


Figure 14: Predicted future occupancy sets of all surrounding traffic participants and ego-vehicle. The time step size is $\Delta t = 0.1$s, and the prediction horizon is $h = 17$.

Third, collision checks are performed for each prediction interval in line 5 of Algorithm 3. The set-based method $\mathsf{checkCollision}\left(\mathcal{O}_j^k, \mathcal{E}_j^k\right)$ returns Boolean true $\top$ if $\mathcal{O}_j^k \cap \mathcal{E}_j^k \not\equiv \emptyset$, otherwise false $\bot$. If no intersection is detected for any of the h prediction intervals, the motion plan of the ego-vehicle is formally verified as safe with respect to the considered traffic participant. Otherwise, the ego-vehicle must modify the intended trajectory or perform a fail-safe maneuver to ensure safety, as explained in the following.

Fail-safe Motion Planning The consideration of all possible future behaviors increasingly restricts the solution space of the ego-vehicle's trajectory the larger the prediction horizon h is chosen. Thus, the formal set-based verification procedure in Section 3.1.2 is primarily used to verify maneuvers with short time horizons. Nevertheless, there exist non-formal long-term trajectories that are initially not safe for all parts of the maneuver while considering all possible future behaviors of the other traffic participants. However, such motion plans can become safe because uncertainty about the other traffic participants' future maneuvers is reduced significantly as time proceeds.

Thus, we use an off-the-shelf trajectory planner to compute a long-term reference motion plan based on the most likely maneuvers of the other surrounding traffic participants, as shown in Fig. 15. Our presented safety verification method is only applied to the first part of the computed motion plan and a consecutive fail-safe maneuver. If formally verified, this part of the long-term plan can be executed safely; then, the next part along the trajectory

is checked for potential collisions. However, the previously verified fail-safe maneuver, e.g., realizing a sufficient distance behind another traffic participant or a standstill, is executed if the verification fails. Please note that the existence of fail-safe trajectories can be proven by induction [32]. Similar to our approach, the braking inevitable collision state concept ensures that the ego-vehicle is always in a legally safe state when a collision occurs, i.e., in a state where it is not causing a collision [14].

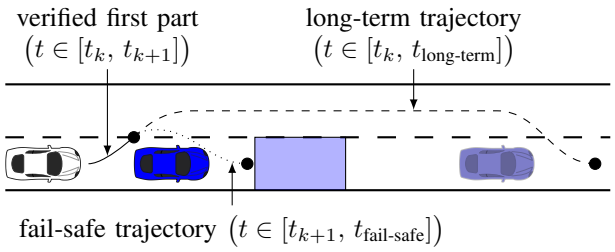


Figure 15: Comparison of long-term and fail-safe trajectory planning of the white ego-vehicle, which wants to overtake the blue traffic participant. The predicted occupancy set of the other vehicle at $t_{\text{fail-safe}}$ and the most likely position at $t_{\text{long-term}}$ are shown by the transparent blue rectangle and vehicle, respectively.

3.1.3 Anytime Safety Verification

In Section 3.1.3, we propose a novel anytime safety verification scheme that attempts to verify that the trajectory of the ego-vehicle is collision-free as quickly as possible. While previous works provide a formal concept, none of these approaches are anytime capable, i.e., the algorithm can be interrupted at any time after completing a short start-up phase, and the quality of the results improves as more computation time is available [47]. To design an efficient algorithm, we

- reuse the list of predicted occupancies $\mathcal{O}^{k-1}$ obtained at the previous time step t_{k-1} (Section 3.1.3),
- order the list of models M based on computational complexity and perform collision checks immediately after a new occupancy set has been computed (Section 3.1.3), and
- refine the predicted occupancies $\mathcal{O}_j^k$ for as long as computation time is available (Section 3.1.3).

Anytime Algorithm Our anytime safety verification procedure is presented in Algorithm 4. It has the same inputs and outputs as Algorithm 3 with one exception, i.e., we use the occupancy

list of the other traffic participant of the previous time step $\mathcal{O}^{k-1}$ for $k \in \mathbb{Z}_{>0}$ as an additional input.

Algorithm 4 Anytime Safety Verification

1: **function** anytimeVerification$(\mathcal{E}^k, M, k, h, \mathcal{O}^{k-1})$
2: **if** updateParameters$(\,) \equiv \top$ **then**
3: $\mathcal{O}^k \leftarrow$ lazyUpdate$(\mathcal{O}^{k-1})$
4: $\mathcal{O}_h^k \leftarrow \mathbb{R}^2$
5: **else**
6: **for all** $j \in \{1, \ldots, h\}$ **do**
7: $\mathcal{O}_j^k \leftarrow \mathbb{R}^2$
8: **for all** $j \in \{1, \ldots, h\}$ **do**
9: $m_j \leftarrow 0$
10: $c_j \leftarrow$ checkCollision$\big(\mathcal{O}_j^k, \mathcal{E}_j^k\big)$
11: **while** $\big(c_j \equiv \top\big) \wedge \big(m_j < |M|\big)$ **do**
12: $m_j \leftarrow m_j + 1$
13: $\mathcal{O}_j^k \leftarrow \mathcal{O}_j^k \cap \mathcal{O}_j^k(M_{m_j})$
14: $c_j \leftarrow$ checkCollision$\big(\mathcal{O}_j^k, \mathcal{E}_j^k\big)$
15: **for all** $j \in \{1, \ldots, h\}$ **do**
16: **for all** $i \in \{m_j + 1, \ldots, |M|\}$ **do**
17: $\mathcal{O}_j^k \leftarrow \mathcal{O}_j^k \cap \mathcal{O}_j^k(M_i)$
18: **return** any(c), $\mathcal{O}^k$

First, we check in line 2 of Algorithm 4 if the updated model parameters at time step t_k have changed compared to those at t_{k-1} based on new sensor data. If altered, the occupancy sets obtained at the previous time step t_{k-1} are based on models M_i that are possibly no longer conformant with the real system. Therefore, we modify the function updateParameters compared to Algorithm 3 by adding a return value that is Boolean false $\perp$ if the model parameters have changed or $t = t_0$, otherwise true $\top$. If the return value is $\top$, our procedure reuses the list of occupancies $\mathcal{O}^{k-1}$ obtained at the previous time step t_{k-1} to quickly obtain over-approximations of $\mathcal{O}_j^k$ for $j \in \{1, \ldots, h-1\}$, as described in Section 3.1.3. Otherwise, all h occupancies are initialized with $\mathbb{R}^2$ in line 7 of Algorithm 4.

In lines 8 to 14, we attempt to verify that no collision occurs for any prediction interval as quickly as possible. This is achieved by ordering the list of models M and performing collision checks (line 14) immediately after obtaining new sets $\mathcal{O}_j^k(M_{m_j})$, as described in

Section 3.1.3. Similar to Algorithm 3, the Boolean collision vector $c \in \mathbb{B}^h$ stores the formal verification result for all h prediction intervals. Furthermore, the variable $m_j \in \mathbb{Z}_{\geq 0}$ for $j \in \{1, \ldots, h\}$ corresponds to the number of models required to verify safety for the prediction interval $[t_{k+j-1},\ t_{k+j}]$.

If more computation time is available, the remaining abstractions are used additionally to refine the occupancies $\mathcal{O}_j^k$ in lines 15 to 17, as described in Section 3.1.3. Finally, our anytime method returns the safety verification result $\mathsf{any}(c)$ in addition to the list $\mathcal{O}^k$ in line 18 of Algorithm 4.

Reuse of Occupancy Lists We can quickly predict future occupancies of another traffic participant at time step t_k for $k \in \mathbb{Z}_{>0}$ by reusing the list $\mathcal{O}^{k-1}$ obtained at the previous time step t_{k-1}. As a result, we only need to compute the occupancy set $\mathcal{O}_h^k$ corresponding to the last prediction interval $[t_{k+h-1},\ t_{k+h}]$ while using elements of $\mathcal{O}^{k-1}$ as over-approximations corresponding to the other intervals, as described subsequently.

Proposition 6. *At time step t_k for $k \in \mathbb{Z}_{>0}$, the relation $\mathcal{O}_{j-1}^k(M_i) \subseteq \mathcal{O}_j^{k-1}(M_i)$ holds for all models M_i and $j \in \{2, \ldots, h\}$.* □

Proof. This relation is valid because the considered time intervals are the same, i.e., it holds that $[t_{k+(j-1)-1},\ t_{k+(j-1)}] \equiv [t_{(k-1)+j-1},\ t_{(k-1)+j}]$. In addition, the prediction uncertainty for the identical interval is reduced after each time step because more information about the other traffic participant has been gathered. □

As mentioned previously, the list of occupancies of another traffic participant at time step t_k is given by $\mathcal{O}^k = \left[\mathcal{O}_1^k, \mathcal{O}_2^k, \mathcal{O}_3^k, \ldots, \mathcal{O}_{h-1}^k, \mathcal{O}_h^k\right]$. Then, the lazy update function, which is called in line 3 of Algorithm 4 with $\mathcal{O}^{k-1}$ as input, is defined by

$$\mathsf{lazyUpdate}(\mathcal{O}^k) = \left[\mathcal{O}_2^k, \mathcal{O}_3^k, \ldots, \mathcal{O}_{h-1}^k, \mathcal{O}_h^k, \mathcal{O}_1^k\right],$$

i.e., the method performs a circular shift.

Based on Proposition 6, by executing $\mathsf{lazyUpdate}(\mathcal{O}^{k-1})$, we quickly obtain an over-approximative result for all prediction time intervals $[t_{k+j-1},\ t_{k+j}]$ with $j \in \{1, \ldots, h-1\}$ at initial time step t_k. Therefore, only the element $\mathcal{O}_h^k$ must be computed based on new sensor measurements at t_k to obtain a valid over-approximative list $\mathcal{O}^k$.

Example 2. *In the upper plot of Fig. 16, all occupancy sets at time step t_0 for $h = 3$ and the rightward moving vehicle are illustrated. Based on Proposition 6, we exploit the fact that $\mathcal{O}_1^1 \subseteq \mathcal{O}_2^0$ and $\mathcal{O}_2^1 \subseteq \mathcal{O}_3^0$ hold to quickly obtain an over-approximative result for the first two*

prediction intervals at the subsequent time step t_1, as shown in the lower plot of Fig. 16. Thus, only the occupancy set $\mathcal{O}_3^1$ must be computed at time step t_1. □

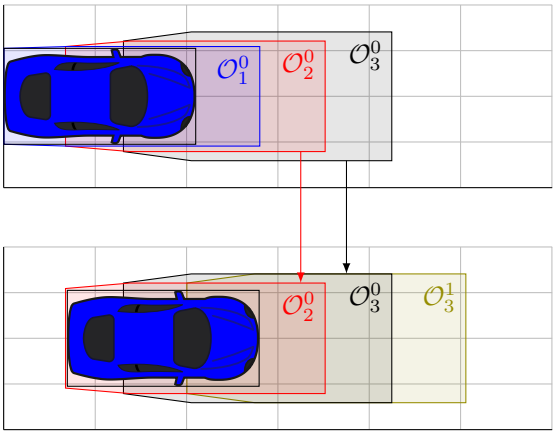


Figure 16: Occupancy sets $\mathcal{O}_2^0$ and $\mathcal{O}_3^0$ computed at time step t_0 (upper plot) are reused at t_1 (lower plot) to over-approximate $\mathcal{O}_1^1$ and $\mathcal{O}_2^1$, respectively. Only the set $\mathcal{O}_3^1$ is computed at time step t_1.

Fast Safety Verification In lines 8 to 14 of Algorithm 4, we attempt to verify that the motion plan of the ego-vehicle is safe for $t \in [t_k,\ t_{k+h}]$ as quickly as possible. First, the set $\mathcal{O}_j^k$ for $j \in \{1, \dots, h-1\}$, which is possibly over-approximated by a reused set as explained in Section 3.1.3, is checked for collision with the ego-vehicle using $\mathcal{E}_j^k$. If the trajectory of the ego-vehicle is unchanged, i.e., $\mathcal{E}_j^k \subseteq \mathcal{E}_{j+1}^{k-1}$, and we can reuse $\mathcal{O}_{j+1}^{k-1}$, the collision check in line 10 always returns $\bot$ and can thus be omitted. However, if a collision is detected in line 10 for a reused set and a changed motion plan, it is unclear whether this is a true or spurious collision due to the reuse of over-approximations. In this case, we verify safety for the first $h-1$ prediction intervals exactly as done for $[t_{k+h-1},\ t_{k+h}]$, which is described in the following.

To speed up the safety verification, we order the list of models M such that M_i has lower computational complexity than M_{i+1} for all $i \in \{1, \dots, |M|-1\}$. As a complexity measure, we use the number of floating point operations required to obtain the corresponding occupancy set. Then, we compute $\mathcal{O}_h^k(M_1)$ corresponding to the simplest abstraction M_1 and intersect this set with the overall occupancy $\mathcal{O}_h^k$ in line 13 of Algorithm 4. Subsequently, a collision check is performed in line 14. If a collision is detected for model M_1, as illustrated in Fig. 17, we compute $\mathcal{O}_h^k(M_2)$ for the second abstraction M_2, intersect it with the overall set $\mathcal{O}_h^k$ to reduce the over-approximation based on (46), and perform a collision check. This procedure is repeated until safety, i.e., $c_h \equiv \bot$, is eventually verified for model M_{m_h} with $m_h \in \{1, \dots, |M|\}$, as illustrated in Fig. 18. Therefore, we formally verify the motion plan of the ego-vehicle as safe using as few model abstractions as possible, beginning with the coarsest ones.

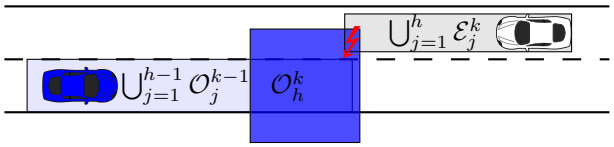


Figure 17: Results: $c_h \equiv \top$ and $\mathcal{O}_h^k \equiv \mathcal{O}_h^k(M_1)$.

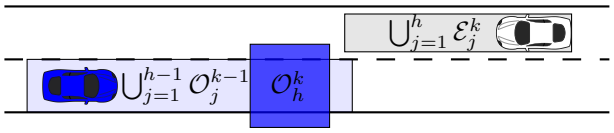


Figure 18: Results: $c_h \equiv \bot$ and $\mathcal{O}_h^k \equiv \bigcap_{i=1}^{m_h} \mathcal{O}_h^k(M_i)$.

Figure 19: Use of m_h models to verify safety for the last prediction interval $[t_{k+h-1},\ t_{k+h}]$, i.e., to show that $\mathcal{O}_h^k \cap \mathcal{E}_h^k \equiv \emptyset$. The first $h-1$ occupancies are over-approximated by the collision-free reused sets obtained at time step t_{k-1}.

The procedure above produces different verification results, i.e., different collision vectors $c \in \mathbb{B}^h$, for the same input data depending on the amount of available computation time. Nevertheless, our interruptible Algorithm 4 can formally verify the safety of the ego-vehicle's trajectory for $t \in [t_k,\ t_{k+h}]$ much faster than Algorithm 3, as shown in Section 3.1.4. In case we cannot verify an intended motion plan in time, we execute the verified fail-safe maneuver, as explained in Section 3.1.2.

Occupancy Set Refinements In lines 15 to 17 of Algorithm 4, our anytime procedure continues computing the occupancy sets $\mathcal{O}_j^k(M_i)$ based on the more complex models M_i for $i \in \{m_j + 1, \ldots, |M|\}$ and the sensor data obtained at t_k, even though the collision vector c no longer changes. This is done to reduce the over-approximation of the occupancy sets for future reuse of these sets, i.e., at initial time steps $t_{k+\tilde{k}}$ for $\tilde{k} \in \{1, \ldots, h\}$. Thus, if more computation time is available, the other abstractions are additionally used to refine the overall occupancies $\mathcal{O}_j^k$ for all $j \in \{1, \ldots, h\}$. Finally, after all occupancy sets are refined, as illustrated in Fig. 20, Algorithm 4 returns the formal verification result $\mathsf{any}(c)$ and the list of computed occupancies $\mathcal{O}^k$, which are identical to the two outputs of Algorithm 3.

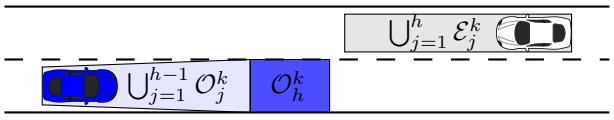


Figure 20: Refined occupancy sets.

3.1.4 Examples

In this subsection, we compare the performance of the formal safety verification Algorithms 3 and 4 on two benchmarks. Our proposed anytime procedure has been integrated into the open-source MATLAB® tool SPOT [28], which represents occupancies by polygons and implements Algorithm 3. Since collision detection involving polygons is relatively slow, we plan to speed it up using bounding volume hierarchies [26, 17] and pre-computed collision checks [36]. To generate a long-term trajectory, as described in Section 3.1.2, we use the sampling-based approach in [46]. All computations are run on a single thread of an Intel® Core™ i7-7820HQ with 32GB RAM.

To determine the computational speed-up potential, we terminate Algorithm 4 as soon as the motion plan is verified. To easily reproduce our results, we use the freely available motion planning benchmark suite CommonRoad[1] [4], since performance comparisons are highly dependent on the specific traffic scenario. Each benchmark is specified by a unique identifier and contains detailed information about the ego-vehicle, road network, and other traffic participants.

PM1:MW1:DEU_Muc-3_1_T-1 Benchmark To visualize the computed occupancy sets for two consecutive time steps, we compare the two verification algorithms using the CommonRoad benchmark PM1:MW1:DEU_Muc-3_1_T-1. The considered traffic scenario comprises an uncontrolled intersection with three other traffic participants and specifies that the ego-vehicle makes the left turn. The initial configuration and the occupancies computed at time step t_0 are shown in Fig. 14. The step size is $\Delta t = 0.1$s, and the prediction horizon is $h = 17$, i.e., we predict the occupancies for all surrounding vehicles for the next 1.7s.

The predicted occupancy sets computed by our interruptible Algorithm 4 at time step t_1 are shown in Fig. 21. As described in Section 3.1.2, we use models M_1, M_2, and M_3, which are ordered by computational complexity. In addition to reusing the occupancies obtained at t_0, it is sufficient to consider only the simplest model M_1 for the first and second vehicles in order to guarantee safety. However, for the third traffic participant, we have to use all three models to formally verify the motion plan.

By averaging the results over 10 simulation runs, we obtain computational speed-ups of Algorithm 4 compared to Algorithm 3 of 33.6, 28.4, and 3.4 for the first, second, and third traffic participants, respectively. This results in an overall speed-up of 7.9 and takes 12ms in total.

[1] commonroad.in.tum.de

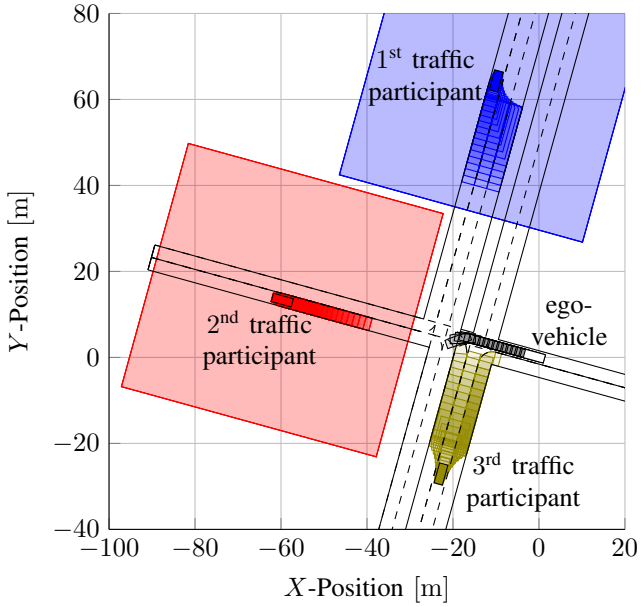


Figure 21: Occupancy sets of all surrounding traffic participants and ego-vehicle computed by interrupted Algorithm 4 for CommonRoad benchmark PM1:MW1:DEU_Muc-3_1_T-1 at time step t_1.

There are multiple reasons why our proposed anytime method is not even faster. For example, the collision check, which is currently computationally expensive due to the intersection of polygons, is performed each time after a new set $\mathcal{O}_h^k(M_i)$ is intersected with the overall $\mathcal{O}_h^k$ in line 13 of Algorithm 4. In contrast, Algorithm 3 performs a single collision check for only the final occupancy set $\mathcal{O}_h^k$. Thus, if computing the occupancy set for M_{i+1} has lower complexity than performing the collision detection for M_i, it may be beneficial to skip this check to optimize, e.g., the expected overall computation time. More importantly, some computations, e.g., obtaining the reachable lanes for model M_3, must be performed regardless of whether the result is only used for the last prediction time interval $[t_{k+h-1},\ t_{k+h}]$ or for all h intervals. However, the complexity of these computations will be reduced in future implementations.

PM1:MW1:DEU_A9-2_1_T-1 Benchmark The second vehicular traffic example is given by the CommonRoad benchmark PM1:MW1:DEU_A9-2_1_T-1. It features a three-lane highway, where the ego-vehicle is initially located in the middle lane and must perform a lane change to the right one, as shown in Fig. 22. Furthermore, this scenario includes two other traffic participants.

Similar to the previous benchmark in Section 3.1.4, the step size is $\Delta t = 0.1$s, and the prediction horizon is $h = 17$. By averaging the results over 10 simulation runs for the whole lane change maneuver, we obtain computational speed-ups of Algorithm 4 compared to

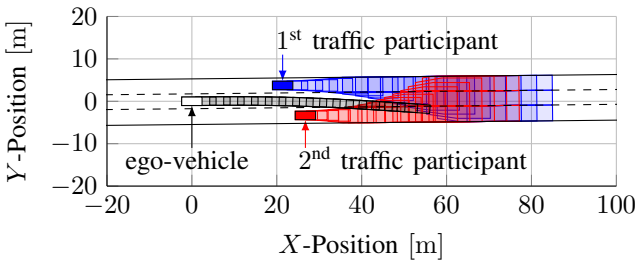


Figure 22: Initial occupancies of all traffic participants and ego-vehicle for CommonRoad benchmark PM1:MW1:DEU_A9-2_1_T-1.

Algorithm 3 of 43.8 and 50.4 for the first and second traffic participants, respectively. This results in an overall speed-up of 47.4 and takes 3ms in total. In contrast to the previous benchmark, it is unnecessary to consider the most complex model M_3 for either of the two other traffic participants in order to guarantee safety, which is the primary reason why a higher overall speed-up is obtained.

3.2 Recursive feasibility and stability of predictive controllers for systems with changing environments

This subsection addresses the questions of how and when the crucial properties of stability and recursive feasibility can be ensured for controlled systems which evolve in dynamic environments. In particular, we consider the setting that the changing environment can be cast into time-varying state constraints to be observed by the controlled system. This setting arises, e.g., in autonomous driving, when an autonomously controlled vehicle has to adjust its path to the complement of the space occupied by other traffic participants [16], or in human-robot cooperation, when the robot controller has to ensure that a robotic manipulator has to circumvent regions momentarily blocked by a human operator [27].

When MPC is used in such cases, the starting point is that in any time instant of an iterative procedure in discrete time, the state constraints of the system to be controlled are predicted over a given future time horizon. These constraints can be obtained by reachable set computations for the environment, e.g. by encoding the regions of a street topology that are potentially occupied by another car, as presented in Section 3.1. These reachable set computations can be executed either offline or online. The state constraints to be considered for the system to be controlled can then be determined as convex subsets of the complement of the reachable sets of all relevant entities of the environment. The requirement of convexity for the state constraints is straightforwardly used to simplify the computation of control strategies in real-time (and in response to changes of the environment).

When adapting the MPC strategy in any time instant of a discrete-time scheme to varying constraints, obviously the questions arises, whether it is possible to always find a feasible solution for the control problem – or phrased differently, it has to be determined which changes of the environment are permissible to ensure the existence of a feasible control strategy, especially when the change of the environment cannot be exactly predicted in advance. The corresponding property is known as *recursive feasibility* in predictive control [30], and it is one important subject of this work. The second property to be investigated is that of *stability*, thus the question of whether the system (subject to the constraints) is certainly driven into a goal set (or towards a reference state) by the predictive controller [34].

While for time-invariant constraints, recursive feasibility and stability have been studied for different settings and definitions, only very little work addresses these properties for time-varying constraints: The work in [45] focuses on MPC with time varying input constraints, where the pattern of how the constraints change is assumed to be known a-priori. Techniques of explicit MPC rely on state-space partitioning which has to be provided in offline computations [2] – considering all configurations (and thus different partitions) which may occur in applications like autonomous driving seems not realistic. The work in [33] introduced the method to homotheticly change the terminal region in order to provide stability guarantees despite changes of the state and input constraints. However, recursive feasibility was not addressed in that work, while being an important pre-requisite of stability of predictive controllers [30].

The following exposition starts from a brief review on how to ensure recursive feasibility and stability of MPC when the state constraints are time-invariant. Then, we extend the discussion to the cases when: (a) there is no suitable model to describe the change of the environment precisely, but where the maximal change of the environment within one sampling time is known to be bounded; (b) a model of the change of the environment exists with bounded uncertainties. In both cases, the change of the environment can be understood as incremental between two successive sampling times, and the bounds can be obtained from reachability computations. We show that under moderate and realistic assumptions, recursive feasibility and asymptotic stability of MPC can be preserved for the two cases.

Time-Invariant State Constraints

To model the dynamics of the system to be controlled, consider the following nonlinear discrete-time difference equations:

$$x_{k+1} = f(x_k, u_k), \tag{47}$$

with state vector $x_k \in \mathbb{R}^{n_x}$ and input vector $u_k \in \mathbb{R}^{n_u}$. At each time step $k \in \mathbb{N} \cup \{0\}$, the system evolution is subject to convex state constraints $X_k = \{x \,|\, C \in \mathbb{R}^{n_c \times n_x}, d_k \in \mathbb{R}^{n_c}, x \in \mathbb{R}^{n_x} : C \cdot x \leq d_k\}$. This represents constraints with changing positions of bounding hyperplanes of the polytope X_k, while the orientation of the hybperplane remains unchanged. The input vector u_k is bounded to a time-invariant set $U \in \mathbb{R}^{n_u}$.

Before indeed considering time-varying state constraints, let us first review the asymptotic stability and recursive feasibility of standard MPC problems with time-invariant constraints, i.e. we first let $\phi_{u,k} = \{u_{k|k}, u_{k+1|k}, \ldots, u_{k+H-1|k}\}$ and $\phi_{x,k} = \{x_{k+1|k}, x_{k+2|k}, \ldots, x_{k+H|k}\}$ denote the prediction of the input and state sequences over a prediction horizon of H steps, and assume $X_k := \overline{X}$, for all $k \in \mathbb{N} \cup \{0\}$. To model state-dependent, input-dependent, and terminal costs, we assume a standard quadratic form of the cost functional $\mathcal{J}(x_k)$:

$$\begin{aligned} \mathcal{J}(x_k) = \sum_{j=0}^{H-1} & \underbrace{(x_{k+j|k}^{\mathrm{T}} Q x_{k+j|k} + u_{k+j|k}^{\mathrm{T}} R u_{k+j|k})}_{\textit{step cost } L(x_{k+j|k}, u_{k+j|k})} \\ & + \underbrace{x_{k+H|k}^{\mathrm{T}} Q_f x_{k+H|k}}_{\textit{terminal cost } F(x_{k+H|k})}, \end{aligned} \tag{48}$$

in which Q, R, and Q_f are chosen as positive-definite weighting matrices. Furthermore, let a terminal set $X_f \subseteq \overline{X}$ be selected. The problem to be solved in step k can then be defined to:

Problem 1.

$$\min_{\phi_{u,k}} \mathcal{J}(x_k)$$

$$\begin{aligned} \textit{s.t.: } & u_{k+j|k} \in U, j \in \{0, \ldots, H-1\}; && \text{(49a)} \\ & x_{k+j|k} \in \overline{X}, j \in \{1, \ldots, H-1\}; && \text{(49b)} \\ & x_{k+H|k} \in X_f. && \text{(49c)} \end{aligned}$$

When using the standard receding-horizon scheme of MPC, only the first step input signal $u_{k|k}$ of the solution $\phi_{u,k}^*$ of Problem 1 in time k is applied, then the next state x_{k+1} is measured, and the solution of Problem 1 is repeated for the updated data in $k+1$.

We next refer to the context of the recursive feasibility of the MPC strategy, which is similarly defined as in [30]:

Definition 3. *(Recursive Feasibility) Given a compact set $\mathcal{F}$ of possible initialization x_0 of the system (47), the MPC controller established by solving Problem 1 in any step k is recursively feasible if and only if for any $x_0 \in \mathcal{F}$, a feasible solution to Problem 1 for $k = 0$ implies the existence of a feasible solution to the problem for any $k \in \{0, 1, 2, \cdots\}$.*

Next, the asymptotic stabilization of the system (47) by the MPC controller obtained from solving Problem 1 is defined similarly to [39]:

Definition 4. *(Asymptotic Stability) If there exists a Lyapunov function $V : \mathbb{R}^{n_x} \to \mathbb{R}$, $V(0) = 0$, on $\overline{X}$, such that for all $k \in \{0, 1, 2, \cdots\}$, the system (47) under control of the solution to Problem 1 satisfies $V(x_{k+1}) < V(x_k)$, then the controlled system is asymptotically stabilized to the origin.*

As discussed in [34], the asymptotic stability according to Def. 2 can be ensured by imposing additional assumptions on the terminal set X_f, namely:

Assumption 1. *The terminal set $X_f \subseteq \overline{X}$ is closed, and $0 \in X_f$ applies.*

Assumption 2. *A terminal controller κ_f exists such that $\kappa_f \cdot \overline{x} \in U$ for all $\overline{x} \in X_f$, and $f(\overline{x}, \kappa_f \cdot \overline{x}) \in X_f$ for all $\overline{x} \in X_f$, i.e., X_f is a control invariant set of the system.*

Assumption 3. *The condition $F(f(\overline{x}, \kappa_f \cdot \overline{x})) - F(\overline{x}) + L(\overline{x}, \kappa_f \cdot \overline{x}) \leq 0$ with L according to (48) applies for all $\overline{x} \in X_f$.*

Then, the following lemma applies:

Lemma 3.1. *If the Assumptions 1, 2, and 3 hold, then the solution to Problem 1 in any step k leads to a state x_{k+1} for which Problem 1 again leads to a feasible solution, and the controlled system is asymptotically stabilized over k.*

Proof. To start with recursive feasibility, we first assume that the state sequence $\phi^*_{x,k} = \{x^*_{k+1|k}, x^*_{k+2|k}, \ldots, x^*_{k+H|k}\}$ is the optimal solution to Problem 1 in time k. Since $x^*_{k+H|k} \in X_f$ according to (49c) applies, there must exist a new state $x_{k+H+1|k} = f(x^*_{k+H|k}, \kappa_f \cdot x^*_{k+H|k}) \in X_f$ and $\kappa_f \cdot x^*_{k+H|k} \in U$ according to Assumption 2. Furthermore, each intermediate state in the sequence $\phi^*_{x,k}$ satisfies $x^*_{k+j|k} \in \overline{X}$, $\forall j \in \{1, \cdots, H\}$. Thus after moving to state $x^*_{k+1|k}$ at step $k+1$, a new candidate state sequence $\phi^{cd}_{x,k+1} = \{x^*_{k+2|k}, \ldots, x^*_{k+H|k}, x_{k+H+1|k}\}$ does also satisfy all state constraints of Problem 1 in $k+1$, and recursive feasibility according to Def. 1 follows from induction.

As for asymptotic stability, the state sequence $\phi^{cd}_{x,k+1}$ in step $k+1$ leads to costs $\mathcal{J}^{cd}(x_{k+1})$. This value constitutes an upper bound of the optimal cost: $\mathcal{J}^{cd}(x_{k+1}) \geq \mathcal{J}^*(x_{k+1})$. Furthermore, the cost difference between $\mathcal{J}^{cd}(x_{k+1})$ and $\mathcal{J}^*(x_k)$ can be calculated from:

$$\begin{aligned}\mathcal{J}^{cd}(x_{k+1}) - \mathcal{J}^*(x_k) &= F(f(x^*_{k+H|k}, \kappa_f \cdot x^*_{k+H|k})) \\ &- F(x^*_{k+H|k}) + L(x^*_{k+H|k}, \kappa_f \cdot x^*_{k+H|k}) - L(x_k, u^*_{k|k}). \end{aligned} \tag{50}$$

According to Assumption 3, the sum of the first three terms on the right-hand side of (50) is non-positive, thus $\mathcal{J}^{cd}(x_{k+1}) - \mathcal{J}^*(x_k) \leq -L(x_k, u^*_{k|k})$, implying also: $\mathcal{J}^*(x_{k+1}) - \mathcal{J}^*(x_k) \leq -L(x_k, u^*_{k|k})$. As the step cost L defined in (48) is always strictly positive outside of the origin, $\mathcal{J}^*$ decreases monotonically. If $\mathcal{J}^*$ is taken as the Lyapunov function according to Def. 2, then asymptotic stability of the controlled system according to Def. 2 is obtained. □

Bounded Changes of State Constraints

After review of the time-invariant case, let us now turn back to the case of time-varying state constraints. Note that even if the exact changes of the constraints cannot be predicted, it still has to be ensured that the successor state obtained from the MPC strategy remains feasible, and the property of asymptotic stability should still apply. Two different scenarios are considered in which the state constraints are not precisely known, and sufficient conditions are proposed to ensure that the desired properties of the MPC strategy hold.

First, we consider the case that no explicit model to predict the change of the constraints between two subsequent steps is available, but only an upper bound of the change. As indicated before, assume that the state constraint changes only with respect to the right hand sides of the inequalities defining $X_{k+1} = \{x \mid C \cdot x \leq d_{k+1}\}$, while the matrix C remains unchanged. This is useful if a translation from X_k to X_{k+1} is sufficient to model the available subset of the state-space, e.g., if an obstacle to the change of state x_k moves, and the constraint X_k is adapted by changing the vector d_k accordingly (without changing the orientation of

X_k).

Let the maximal change of any component of the vector d_{k+1} compared to d_k be bounded by:

$$|d_{k+1}(i) - d_k(i)| \leq w_{i,max},\ w_{i,max} \in \mathbb{R}^{\geq 0}, \tag{51}$$

for all $i \in \{1, \cdots, n_c\}$. The value of $w_{i,max}$ can be obtained, e.g., by evaluating the physical limits of the entity which constitutes the changing environment (e.g. the maximal acceleration of a vehicle, interacting with autonomous car to be controlled).

Based on this information, one can obtain a conservative estimation of the change of the environment over the horizon by using the prediction:

$$X_{k+j|k} = \{x \,|\, C \cdot x \leq d_k - j \cdot d_{max}\},\ \forall\, j \in \{1, \cdots, H\}, \tag{52}$$

with $d_{max} = [w_{1,max}, \cdots, w_{n_c,max}]^{\mathrm{T}} \in \mathbb{R}^{n_c}$. The set $X_{k+j|k}$ represents a conservative estimation (obtained at time k) of the future constraint X_{k+j} being indeed available for trajectory planning. We define the set:

$$\phi_X = \{X_{k+1|k}, X_{k+2|k}, \ldots, X_{k+H|k}\}. \tag{53}$$

of conservatively predicted state constraints, see also Fig. 23 for an illustration.

Assumption 4. *The set $X_{k+H|k}$ is not empty and contains the terminal set X_f, $0 \in X_f$, for all $k \in \{0, 1, 2, \cdots\}$.*

Note that as long as the terminal set X_f is contained in $X_{k+H|k}$ for all $k \in \{0, 1, 2, \cdots\}$, then it will remain to be a control invariant set [12] of the system by employing the terminal controller κ_f, despite the change of the state constraints. In other words, if the Assumption 4 holds, then we do not have to re-determine the terminal set X_f with respect to the change of the environment.

Lemma 3.2. *If the condition $x_{k+j} \in X_{k+j|k}$ is satisfied for all $j \in \{1, \cdots, H\}$, then $x_{k+j} \in X_{k+j}$ applies, too.*

Proof. According to (51), the relation $|d_{k+j}(i) - d_k(i)| \leq j \cdot w_{i,max}$ applies for all $j \in \{1, \cdots, H\}$ and for all $i \in \{1, \cdots, n_c\}$. This, implies also the relation $|d_{k+j} - d_k| \leq j \cdot d_{max}$ according to the definition of d_{max}. Thus, $d_k - j \cdot d_{max} \leq d_{k+j}$ holds true and implies $X_{k+j|k} \subseteq X_{k+j}$. Accordingly, $x_{k+j} \in X_{k+j|k}$ requires that $x_{k+j} \in X_{k+j}$. □

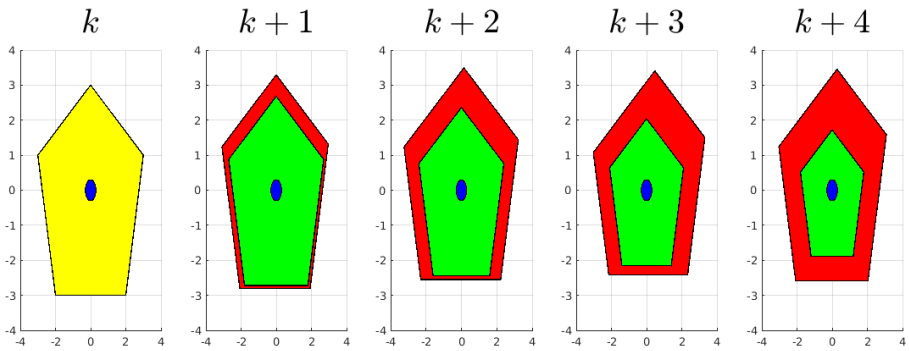


Figure 23: Conservative inner approximation of X_{k+j} through $X_{k+j|k}$ for all $j \in \{1, \cdots, 4\}$ (yellow: X_k, red: X_{k+j}, green: $X_{k+j|k}$, blue: X_f).

Lemma 3.3. *Given $j \in \{1, \cdots, H\}$, then for any $j_1 \in \mathbb{N} \cup \{0\}$ with $0 \leq j_1 \leq j$, it applies that $X_{k+j|k} \subseteq X_{k+j|k+j_1}$.*

Proof. According to (52), for j and j_1 with $0 \leq j_1 \leq j \leq H$, the predicted constraints $X_{k+j|k}$ and $X_{k+j|k+j_1}$ take the form of:

$$X_{k+j|k} = \{x \,|\, C \cdot x \leq d_k - j \cdot d_{max}\},$$
$$X_{k+j|k+j_1} = \{x \,|\, C \cdot x \leq d_{k+j_1} - (j - j_1) \cdot d_{max}\}.$$

Then, according to (51), vector $d_{k+j_1} \geq d_k - j_1 \cdot d_{max}$, which implies:

$$d_{k+j_1} - (j - j_1) \cdot d_{max} \geq d_k - j_1 \cdot d_{max} - (j - j_1) \cdot d_{max},$$
$$d_{k+j_1} - (j - j_1) \cdot d_{max} \geq d_k - j \cdot d_{max},$$

i.e. the right hand side of the inequality for $X_{k+j|k}$ is not larger than that for $X_{k+j|k+j_1}$. Furthermore, as the matrix C in both constraints $X_{k+j|k}$ and $X_{k+j|k+j_1}$ is the same, the relation $X_{k+j|k} \subseteq X_{k+j|k+j_1}$ applies. □

Note the Lemma 3.2 and 3.3 together establish the following facts:

- The constraint $X_{k+j|k}$ is a conservative (inner) estimation of the true set X_{k+j} by taking all possible realizations of the changes of the environment into account.
- The relation $X_{k+j+1|k} \subseteq X_{k+j|k}$ applies according to (52), meaning that the estimation is increasingly more conservative over j; Additionally, it also implies with $X_{k+H|k} \neq \emptyset$ from Assumption 4, that $X_{k+j|k} \neq \emptyset$ applies for all $j \in \{1, \cdots, H\}$.
- The estimation of the true constraint X_{k+j} based on set X_{k+j_1} with $j_1 \leq j$, is less conservative than based on set X_k, according to Lemma 3.3.

Now, the following optimization problem is defined for step k with use of the predicted state constraints:

Problem 2.

$$\min_{\phi_{u,k}} \mathcal{J}(x_k)$$

$$s.t.: u_{k+j|k} \in U, j \in \{0, \dots, H-1\}; \tag{54a}$$

$$x_{k+j|k} \in X_{k+j|k}, j \in \{1, \dots, H-1\}; \tag{54b}$$

$$x_{k+H|k} \in X_f. \tag{54c}$$

Lemma 3.4. *If the Assumptions 2, 3, and 4 hold, then the solution to Problem 2 for any $k \in \{0, 1, 2, \cdots\}$ will establish recursive feasibility and the system (47) is asymptotically stabilized into the origin.*

Proof. First, since $x_{k+1|k} \in X_{k+1|k}$ is ensured by constraint (54b) in Problem 2, and $X_{k+1|k} \subseteq X_{k+1}$ applies according to Lemma 3.2, the state x_{k+1} resulting from solving Problem 2 is guaranteed to be contained in constraint X_{k+1} despite the uncertainties.
The state constraints in Problem 2 in the step $k+1$ are:

$$x_{k+1+(j)|k+1} \in X_{k+1+(j)|k+1}, j \in \{1, \dots, H-1\}; \tag{55a}$$

$$x_{k+1+(H)|k+1} \in X_f. \tag{55b}$$

Similar as in the proof of Lemma 3.1, the state sequence $\phi^*_{x,k} = \{x^*_{k+1|k}, x^*_{k+2|k}, \dots, x^*_{k+H|k}\}$ denotes the optimal solution of Problem 2 in k, and a candidate state sequence $\phi^{cd}_{x,k+1} = \{x^*_{k+2|k}, \dots, x^*_{k+H|k}, x_{k+H+1|k}\}$ is obtained based on $\phi^*_{x,k}$, with $x_{k+H+1|k} = f(x^*_{k+H|k}, \kappa_f \cdot x^*_{k+H|k}) \in X_f$.

Note that for the first $H-2$ states in the candidate sequence $\phi^{cd}_{x,k+1}$, it applies that $x^*_{k+j+1|k} \in X_{k+j+1|k}$ for all $j \in \{1, \cdots, H-2\}$ according to constraint (54b). Based on Lemma 3.3, $X_{k+j+1|k} \subseteq X_{k+1+(j)|k+1}$ holds for all $j \in \{1, \cdots, H-2\}$, implying $x^*_{k+j+1|k} \in X_{k+j+1|k} \subseteq X_{k+1+(j)|k+1}$, for all $j \in \{1, \cdots, H-2\}$.

Furthermore, the penultimate state $x^*_{k+H|k}$ in $\phi^{cd}_{x,k+1}$ is contained in X_f according to constraint (54c). Then, since $X_f \subseteq X_{k+H|k} \subseteq X_{k+1+(H-1)|k+1}$ applies according to Assumption 4 and Lemma 3.3, state $x^*_{k+H|k}$ is also contained in $X_{k+1+(H-1)|k+1}$.

Finally, the last state $x_{k+H+1|k}$ in $\phi^{cd}_{x,k+1}$ satisfies $x_{k+H+1|k} \in X_f$ according to Assumption 2.

Thus, the candidate sequence $\phi^{cd}_{x,k+1}$ satisfies all the constraints of Problem 2 in step $k+1$, and as the corresponding input sequence satisfies the input constraint according to Assumption 2 and (54a), recursive feasibility of the MPC strategy according to Def. 1 is guaranteed.

The proof to the *asymptotic stability* follows a path similar as in the time-invariant case, since the relation $\mathcal{J}^{cd}(x_{k+1}) - \mathcal{J}^*(x_k) \leq -L(x_k, u^*_{k|k})$ still applies, ensuring $\mathcal{J}^*(x_{k+1}) - \mathcal{J}^*(x_k) \leq -L(x_k, u^*_{k|k})$. Thus, the monotonic decrease of the cost of the Lyapunov function applies, leading to asymptotic stability of the controlled system according to Def. 2. □

Modelled constraint variation with uncertainties

In contrast to the previous case, in which the constrained set were shrinking in all directions over the prediction, we here model the changes of the constrained state set such that a translation towards the goal is possible, i.e., we consider a more general case, and still the prediction of the constrained sets are subject to uncertainties.

Now, we assume that a model $\mathcal{M}$ for the prediction of the change of the state constraints exists, but it contains uncertainties, which may accumulate over the steps of the prediction horizon.

We assume that in the current step k, the state constraints still have the form $X_k = \{x \,|\, C \cdot x \leq d_k\}$, but the predictions $\hat{X}_{k+j|k} = \{x \,|\, \hat{d}_{k+j|k} \in \mathbb{R}^{n_c} : C \cdot x \leq \hat{d}_{k+j|k}\}$, for $j \in \{1, \ldots, H\}$ are iteratively computed from:

$$\hat{d}_{k+j+1|k} := \mathcal{M}(\hat{d}_{k+j|k}),\ \forall\, 0 \leq j \leq H-1, \tag{56}$$

where $\hat{d}_{k+0|k} := d_k$, and $\mathcal{M} : \mathbb{R}^{n_c} \to \mathbb{R}^{n_c}$ denotes a model for the variation of the vector $\hat{d}_{k+j|k}$. The prediction of the state constraint in the next step is derived from the state constraint in previous step, while C again remains unchanged during the prediction.

In addition, we require that for two different state constraints, if their maximal difference is bounded, the difference of the predicted constraints in the next step obtained from the model $\mathcal{M}$ is also bounded and bounded by the same value. Such requirement is summarized as in the following:

Assumption 5. *For any two vectors d^a_k, $d^b_k \in R^{n_c}$, with $d^a_k \neq d^b_k$, if $|d^a_k(i) - d^b_k(i)| \leq \gamma_i$, $\gamma_i \geq 0$ applies for all $i \in \{1, \cdots, n_c\}$, then $|\mathcal{M}(d^a_k)(i) - \mathcal{M}(d^b_k)(i)| \leq \gamma_i$ also applies.*

In order to formulate the deviation between accumulated uncertainties of the predicted constraint $\hat{X}_{k+j|k}$ and the true constraint $X_{k+j} = \{x \,|\, C \cdot x \leq d_{k+j}\}$, let each component of

the vector $\hat{d}_{k+j|k}$ satisfy the following property:

$$|\hat{d}_{k+j|k}(i) - d_{k+j}(i)| \leq j \cdot \hat{w}_{i,max},\ \hat{w}_{i,max} \in \mathbb{R}^{\geq 0}, \tag{57}$$

for all $i \in \{1, \cdots, n_c\}$. It implies that the uncertainty over the prediction may linearly increase over j. Similarly to (51) in the last section, the requirement (57) is reasonable since the upper bound of $\hat{w}_{i,max}$ can be determined offline from experiments.

To consider the maximally possible uncertainty of the predicted constraint $\hat{X}_{k+j|k}$, a tightened constraint $\tilde{X}_{k+j|k} = \{x \,|\, \tilde{d}_{k+j|k} \in \mathbb{R}^{n_c} : C \cdot x \leq \tilde{d}_{k+j|k}\}$ is determined according to:

$$\tilde{d}_{k+j|k} := \hat{d}_{k+j|k} - j \cdot \hat{d}_{max}, \tag{58}$$

with vector $\hat{d}_{max} = [\hat{w}_{1,max}, \cdots, \hat{w}_{n_c,max}]^{\mathrm{T}} \in \mathbb{R}^{n_c}$ (see Fig. 24).

Assumption 6. *Let the set $\tilde{X}_{k+j|k}$ be non-empty for all $k \in \{0, 1, 2, \cdots\}$ and $j \in \{1, \ldots, H\}$, and the terminal set X_f be included in $\tilde{X}_{k+H|k}$, $0 \in X_f$.*

Similar to Assumption 4, if the Assumption 6 holds, then the terminal set X_f has not to be redetermined with respect to the change of the environment.

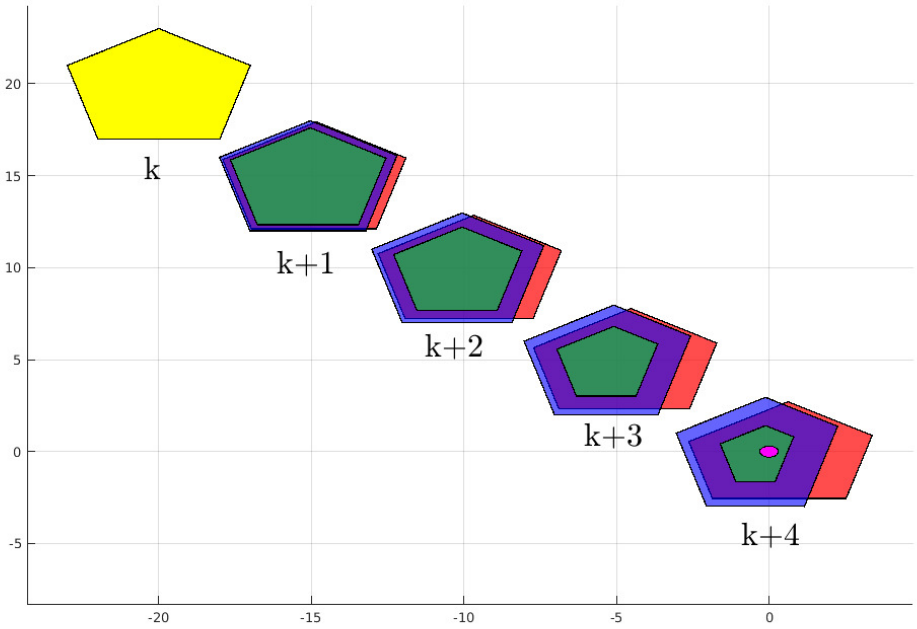


Figure 24: Tightening the predicted set $\hat{X}_{k+j|k}$ according to the knowledge of $\hat{d}_{max}$ for all $j \in \{1, \cdots, 4\}$ (yellow: X_k, blue: $\hat{X}_{k+j|k}$, green: $\tilde{X}_{k+j|k}$, red: X_{k+j}, magenta: X_f).

Lemma 3.5. *If $\tilde{X}_{k+j|k}$ is obtained from (58), where $\hat{d}_{k+j|k}$ follows from model $\mathcal{M}$ according to (56), and if $x_{k+j} \in \tilde{X}_{k+j|k}$ for all $j \in \{1, \cdots, H\}$, then $x_{k+j} \in X_{k+j}$ applies too.*

Proof. According to (57) and the definition of $\hat{d}_{max}$, the relation $\hat{d}_{k+j|k} - d_{k+j} \leq j \cdot \hat{d}_{max}$ applies. Furthermore, since $\tilde{d}_{k+j|k} = \hat{d}_{k+j|k} - j \cdot \hat{d}_{max}$ holds, the relation $\tilde{d}_{k+j|k} \leq d_{k+j}$ applies, implying $\tilde{X}_{k+j|k} \subseteq X_{k+j}$. Accordingly, for all $x_{k+j} \in \tilde{X}_{k+j|k}$, relation $x_{k+j} \in X_{k+j}$ must hold, too. □

Lemma 3.6. *Given the situation in Lemma 3.5 and $j \in \{1, \cdots, H\}$, then for any $j_1 \in \mathbb{N} \cup \{0\}$ with $0 \leq j_1 \leq j$, it applies that $\tilde{X}_{k+j|k} \subseteq \tilde{X}_{k+j|k+j_1}$.*

Proof. According to (57), it applies for the constraint X_{k+j_1} with $0 \leq j_1 \leq j \leq H$, that:

$$|\hat{d}_{k+j_1|k} - d_{k+j_1}| \leq j_1 \cdot \hat{d}_{max}.$$

Then, according to Assumption 5, for the prediction of the constraint X_{k+j} by starting once from the constraint $\hat{X}_{k+j_1|k}$, and once starting from the constraint X_{k+j_1}, the difference is bounded by:

$$|\hat{d}_{k+j|k} - \hat{d}_{k+j|k+j_1}| \leq j_1 \cdot \hat{d}_{max}.$$

Now, according to (58), substituting $\hat{d}_{k+j|k}$ by $\tilde{d}_{k+j|k} + j \cdot \hat{d}_{max}$, and substituting $\hat{d}_{k+j|k+j_1}$ by $\tilde{d}_{k+j|k+j_1} + (j - j_1) \cdot \hat{d}_{max}$ leads to the relation:

$$\begin{aligned}&(\tilde{d}_{k+j|k} + j \cdot \hat{d}_{max}) - (\tilde{d}_{k+j|k+j_1} + (j - j_1) \cdot \hat{d}_{max}) \\ &\leq j_1 \cdot \hat{d}_{max},\end{aligned}$$

and thus to:

$$\tilde{d}_{k+j|k} - \tilde{d}_{k+j|k+j_1} \leq 0.$$

With $\tilde{d}_{k+j|k} \leq \tilde{d}_{k+j|k+j_1}$, and given that C in the constraints $\tilde{X}_{k+j|k}$ and $\tilde{X}_{k+j|k+j_1}$ are the same, the relation $\tilde{X}_{k+j|k} \subseteq \tilde{X}_{k+j|k+j_1}$ holds.

□

Now, the following substitute optimization problem can be defined for step k:

Problem 3.

$$\min_{\phi_{u,k}} \mathcal{J}(x_k)$$

$$\text{s.t.: } u_{k+j|k} \in U, j \in \{0, \ldots, H-1\}; \tag{59a}$$

$$x_{k+j|k} \in \tilde{X}_{k+j|k}, j \in \{1, \ldots, H-1\}; \tag{59b}$$

$$x_{k+H|k} \in X_f. \tag{59c}$$

Lemma 3.7. *If the Assumptions 2, 3, 5, and 6 hold, then the solution to Problem 3 in step k will lead to a state x_{k+1} which also satisfies the constraint X_{k+1} at step $k+1$. Furthermore, this solution implies that Problem 3 again has a feasible solution in step $k+1$, and the controlled system is asymptotically stabilized into the origin.*

Proof. Following the reasoning in the proof of Lemma 3.4, as $x_{k+1|k} \in \tilde{X}_{k+1|k}$ in ensured by constraint (59b) in Problem 3, and since $\tilde{X}_{k+1|k} \subseteq X_{k+1}$ applies according to Lemma 3.5, the state x_{k+1} resulting from the solution of Problem 3 in step k is guaranteed to be contained in X_{k+1}. To obtain recursive feasibility, consider the state constraints of Problem 3 in step $k+1$:

$$x_{k+1+(j)|k+1} \in \tilde{X}_{k+1+(j)|k+1}, j \in \{1, \ldots, H-1\}; \tag{60a}$$

$$x_{k+1+(H)|k+1} \in X_f. \tag{60b}$$

If the state sequence $\phi^*_{x,k} = \{x^*_{k+1|k}, x^*_{k+2|k}, \ldots, x^*_{k+H|k}\}$ is the optimal solution to Problem 3, let a candidate state sequence $\phi^{cd}_{x,k+1} = \{x^*_{k+2|k}, \ldots, x^*_{k+H|k}, x_{k+H+1|k}\}$ be obtained from $\phi^*_{x,k}$ with $x_{k+H+1|k} = f(x^*_{k+H|k}, \kappa_f \cdot x^*_{k+H|k}) \in X_f$.

For the first $H-2$ states in $\phi^{cd}_{x,k+1}$, the inclusion $x^*_{k+j+1|k} \in \tilde{X}_{k+j+1|k}$ applies for all $j \in \{1, \cdots, H-2\}$ according to constraint (59b). Based on Lemma 3.6, $\tilde{X}_{k+j+1|k} \subseteq \tilde{X}_{k+1+(j)|k+1}$ holds for all $j \in \{1, \cdots, H-2\}$, and thus $x^*_{k+j+1|k} \in \tilde{X}_{k+j+1|k} \subseteq \tilde{X}_{k+1+(j)|k+1}$, for all $j \in \{1, \cdots, H-2\}$. Since the penultimate state $x^*_{k+H|k}$ in $\phi^{cd}_{x,k+1}$ is contained in X_f given (59c), and since $X_f \subseteq \tilde{X}_{k+H|k} \subseteq \tilde{X}_{k+1+(H-1)|k+1}$ applies according to Assumption 6 and Lemma 3.6, it also applies that $x^*_{k+H|k} \in \tilde{X}_{k+1+(H-1)|k+1}$. Furthermore, the last state $x_{k+H+1|k}$ in $\phi^{cd}_{x,k+1}$ satisfies $x_{k+H+1|k} \in X_f$ according to Assumption 2.

Hence, the sequence $\phi^{cd}_{x,k+1}$ satisfies all state constraints of Problem 3 in step $k+1$. In addition, the input sequence leading to $\phi^{cd}_{x,k+1}$ must satisfy the input constraints according to Assumption 2 and (59a), thus recursive feasibility of the MPC strategy according to Def. 1 is guaranteed.

The proof of asymptotic stability is the same as in Lemma 3.4, where the recursive feasibility implies the monotonic decrease of the costs (establishing a Lypunov function). The single steps are not repeated here in detail. □

3.2.1 Illustrating Example

Here, an application of proposed method to the use case of human-robot collaboration as considered is illustrated. As it is generally hard to obtain an exact model for human motion prediction (see the discussion in Deliverable 2.1), we employ experimental data obtained offline to over-approximate the space occupied by a human arm within predictive control of a robot manipulator: the human arm is assumed to move maximally $5cm$ in every $7ms$ (approximately $0.75m/s$) in all directions. The $7ms$ here is the sampling time of the robot manipulator, in which the position of its end-effector is updated. The task of the predictive controller is to move the end-effector from the current position to a desired goal position without collision with the human arm.

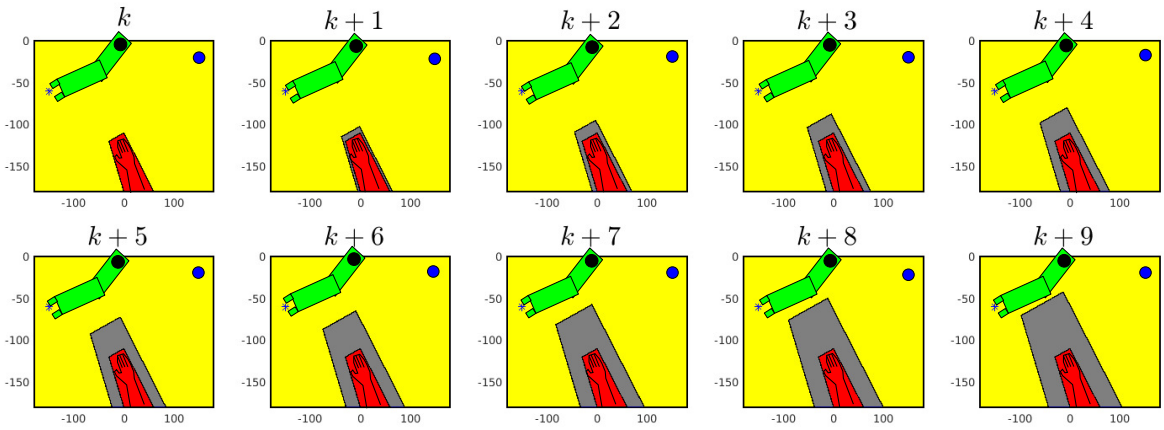


Figure 25: Conservative estimation of the human arm position over the future $H = 9$ steps, based on its current position in step k: the gray regions are the regions potentially occupied by the human arm in each step according to its maximal movement per step, and the yellow regions are the feasible regions for the robot manipulator; the blue dot marks the selected target position.

Note that the considered problem is related to the first case of applying MPC in a time-varying environment in the preceding sections, in which only the bounded change of the state constraints is known. Thus, based on the current, measurable position of the human arm as well as its maximal change per step, a conservative estimation of the position over the future steps is determined according to (52), see Fig. 25. Thus, the space (in yellow) complementary to the region potentially occupied by the human arm (in gray) is the feasible region for the robot manipulator.

Then, as the recursive feasibility and asymptotic stability is proven for this application

of the predictive controller corresponding to Problem 2, the desired control task is exemplarily proven to be solvable, see the plot of the position trajectory of the end-effector from a start to a goal position in Fig. 26.

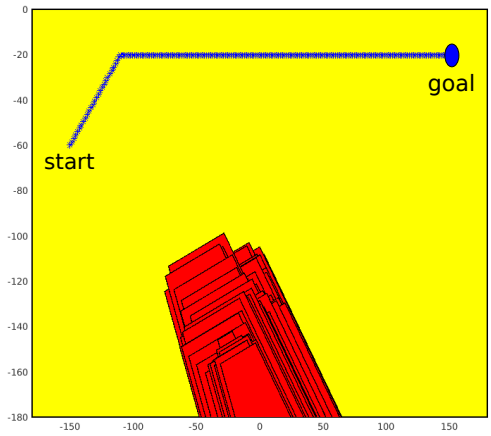


Figure 26: The target position is reached after 198 steps (1.386s) and the red regions are the positions of the human arm during the transition.

Concluding remarks

In this section, we have addressed the conditions on ensuring recursive feasibility and asymptotic stability of MPC strategies for nonlinear dynamic systems under time-varying state constraints. For the case that no suitable model for the prediction of the state constraints are available, the key point is to construct a conservative bound on the change of the constraints, based on knowledge of the current constraints. In the second case, when an uncertain model of the environment is available, tightened constraints can be used to conservatively bound the change of the constraints by taking the maximal prediction uncertainty into account. In both cases, recursive feasibility and asymptotic stability can be shown. In future work, the findings in this section will be extended to hybrid dynamics as well as to communication of constraints in settings of distributed MPC.

4 Conclusions

In this deliverable, we described new approaches developed within the UnCoVerCPS project that exploit the decomposition of a system into subsystems for improving the scalability of verification methods. This is particularly interesting when the problem includes discrete variables since in some cases the combinatorial complexity can hamper the problem solution.

We also considered the adoption of online verification methods for controller tuning and addressed important issues like stability and recursive feasibility in model predictive control when using time-varying constrained sets arising from reach set computation, and the impact of using coarse model abstractions to provide rapid although conservative verification results in safety critical systems. An algorithm that provides formal safety guarantees is presented in the latter case.

The described methods find application in the use cases of the projects and, in particular, those on autonomous driving (see Section 3.1), collaborative robotics (see Section 3.2.1), and on smart grids (a paper is in preparation at PoliMi on the usage of the parallel decomposition approach in Section 2.2.2 for implementing ancillary services offered to the main grid), besides other industrial case studies as that by Bosch in Section 2.1.4 validating the methods in Section 2.1.

References

[1] http://spaceex.imag.fr/.

[2] A. Alessio and A. Bemporad. A survey on explicit model predictive control. In *Nonlinear model predictive control*, pages 345–369. Springer, 2009.

[3] M. Althoff and J. M. Dolan. Online verification of automated road vehicles using reachability analysis. *IEEE Transactions on Robotics*, 30(4):903–918, 2014.

[4] M. Althoff, M. Koschi, and S. Manzinger. CommonRoad: Composable benchmarks for motion planning on roads. In *IEEE Intelligent Vehicles Symposium*, pages 719–726, 2017.

[5] M. Althoff and S. Magdici. Set-based prediction of traffic participants on arbitrary road networks. *IEEE Transactions on Intelligent Vehicles*, 1(2):187–202, 2016.

[6] J.-P. Aubin and I. Ekeland. Estimates of the duality gap in nonconvex optimization. *Mathematics of Operations Research*, 1(3):225–245, 1976.

[7] A. Bemporad, G. Ferrari-Trecate, and M. Morari. Observability and Controllability of Piecewise Affine and Hybrid Systems. *IEEE Transactions on Automatic Control*, 45(10):1864–1876, 2000.

[8] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[9] D. Bertsekas, G. Lauer, N. Sandell, and T. Posbergh. Optimal short-term scheduling of large-scale power systems. *IEEE Transactions on Automatic Control*, 28(1):1–11, 1983.

[10] D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[11] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[12] F. Blanchini. Set invariance in control. *Automatica*, 35(11):1747–1767, 1999.

[13] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling. Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*, pages 41–50. ACM, 2018.

[14] S. Bouraine, T. Fraichard, and H. Salhi. Provably safe navigation for mobile robots with limited field-of-views in dynamic environments. *Autonomous Robots*, 32(3):267–283, 2012.

[15] S. I. Dudov and E. A. Meshcheryakova. Method for finding an approximate solution of the asphericity problem for a convex body. *Computational Mathematics and Mathematical Physics*, 53(10):1483–1493, 2013.

[16] J. Eilbrecht and O. Stursberg. Cooperative driving using a hierarchy of mixed-integer programming and tracking control. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*, pages 673–678. IEEE, 2017.

[17] C. Ericson. *Real-time collision detection*. CRC Press, 2004.

[18] A. Falsone, K. Margellos, and M. Prandini. A decentralized approach to multi-agent milps: finite-time feasibility and performance guarantees. *Automatica*, 2018. Provisionally accepted as a regular paper.

[19] G. Frehse, C. L. Guernic, A. Donzé, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In G. Gopalakrishnan and S. Qadeer, editors, *CAV*, LNCS. Springer, 2011.

[20] A. M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study 2*, pages 82–114, 1974.

[21] A. Girard, C. Le Guernic, and O. Maler. Efficient computation of reachable sets of linear time-invariant systems with inputs. In *HSCC'06*. Springer, 2006.

[22] F. Gruber and M. Althoff. Anytime safety verification of autonomous vehicles. In *IEEE 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018. Accepted.

[23] W. Heemels, B. D. Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, July 2001.

[24] R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[25] M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-art*. Springer Science & Business Media, 2009.

[26] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.

[27] D. Kontny and O. Stursberg. Online adaption of motion paths to time-varying constraints using homotopies. *IFAC-PapersOnLine*, 50(1):3331–3337, 2017.

[28] M. Koschi and M. Althoff. SPOT: A tool for set-based prediction of traffic participants. In *IEEE Intelligent Vehicles Symposium*, pages 1686–1693, 2017.

[29] G. Lafferriere, G. J. Pappas, and S. Yovine. Symbolic reachability computation for families of linear vector fields. *Symbolic Computation*, 32:231–253, 2001.

[30] J. Löfberg. Oops! i cannot do it again: Testing for recursive feasibility in mpc. *Automatica*, 48(3):550–555, 2012.

[31] A. V. Lotov and A. I. Pospelov. The modified method of refined bounds for polyhedral approximation of convex polytopes. *Computational Mathematics and Mathematical Physics*, 48(6):933–941, 2008.

[32] S. Magdici and M. Althoff. Fail-safe motion planning of autonomous vehicles. In *Proc. of the 19th International IEEE Conference on Intelligent Transportation Systems*, pages 452–458, 2016.

[33] T. Manrique, T. Fiacchini, Mand Chambrion, and G. Millérioux. Mpc tracking under time-varying polytopic constraints for real-time applications. In *Control Conference (ECC), 2014 European*, pages 1480–1485. IEEE, 2014.

[34] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[35] N. J. Redondo and A. Conejo. Short-term hydro-thermal coordination by lagrangian relaxation: solution of the dual problem. *IEEE Transactions on Power Systems*, 14(1):89–95, 1999.

[36] A. Rizaldi, S. Söntges, and M. Althoff. On time-memory trade-off for collision detection. In *Proc. of the IEEE Intelligent Vehicles Symposium*, pages 1173 – 1180, 2015.

[37] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. Reachset conformance testing of hybrid automata. In *Proc. of Hybrid Systems: Computation and Control*, pages 277–286, 2016.

[38] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff. Ensuring drivability of planned motions using formal methods. In *Proc. of the 20th IEEE International Conference on Intelligent Transportation Systems*, pages 1–8, 2017.

[39] P. Scokaert, D. Mayne, and J. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44(3):648–654, 1999.

[40] N. Shor. *Minimization Methods for Non-Differentiable Functions.* Springer, 1985.

[41] G. Söderlind. The logarithmic norm. history and modern theory. *BIT Numerical Mathematics*, 46(3):631–652, 2006.

[42] R. Vignali and M. Prandini. Input design for a cascading system: An approach based on system decomposition and non-influential input detection. In *2014 IEEE Multi-Conference on Systems and Control*, Antibes, France, October 2014.

[43] R. Vignali and M. Prandini. Model reduction of discrete time hybrid systems: A structural approach based on observability. In *2016 International Workshop on Symbolic and Numerical Methods for Reachability Analysis (SNR)*, pages 1–6, April 2016.

[44] R. Vujanic, P. M. Esfahani, P. J. Goulart, S. Mariéthoz, and M. Morari. A decomposition method for large scale MILPs, with performance guarantees and a power system application. *Automatica*, 67:144–156, 2016.

[45] N. Wada, H. Tomosugi, and M. Saeki. Model predictive tracking control for a linear system under time-varying input constraints. *International Journal of Robust and Nonlinear Control*, 23(9):945–964, 2013.

[46] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. In *IEEE Conference on Robotics and Automation*, pages 987–993, 2010.

[47] S. Zilberstein. Using anytime algorithms in intelligent systems. *AI magazine*, 17(3):73–83, 1996.