Correct-by-Construction Control Synthesis for High-Dimensional Systems

by

Lars Petter Nilsson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in the University of Michigan
2017

Doctoral Committee:

Professor Jessy W. Grizzle, Co-Chair
Assistant Professor Necmiye Ozay, Co-Chair
Professor Stéphane Lafortune
Assistant Professor Ram Vasudevan

Lars Petter Nilsson
pettni@umich.edu
ORCID iD: 0000-0001-8748-6936

Table of Contents

List of Figures

List of Tables

List of Appendices

List of Acronyms

$GR(1)$ Generalized Reactivity (1).

AFTS Augmented Finite Transition System.

DFTS Deterministic Finite Transition System.

DSOS Diagonally dominant Sum of Squares.

FTS Finite Transition System.

HJB Hamilton Jacobi Bellman.

ILP Integer Linear Program.

LMI Linear Matrix Inequality.

LP Linear Program.

LTL Linear Temporal Logic.

SDP Semi-Definite Program.

SDSOS Scaled Diagonally dominant Sum of Squares.

SOS Sum of Squares.

List of Symbols

Discrete spaces		**Continuous spaces**	
Discrete state space	$\mathbb{X}$	Continuous state space	$\mathcal{X}$
Discrete input space	$\mathbb{U}$	Continuous input space	$\mathcal{U}$
Discrete disturbance space	$\mathbb{D}$	Continuous disturbance space	$\mathcal{D}$
Discrete output space	$\mathbb{Y}$	Continuous output space	$\mathcal{Y}$
Dynamical systems			
Transition system	$\mathcal{T} = (\mathbb{X}, \mathbb{U}, \longrightarrow, h)$		
Discrete-time system	$\Sigma_{DT} = (\mathcal{X}, \mathcal{U}, \mathcal{D}, f, h)$		
Continuous-time system	$\Sigma_{CT} = (\mathcal{X}, \mathcal{U}, \mathcal{D}, f, h)$		
Switched system	$\Sigma_{Sw} = (\mathcal{X}, \mathbb{U}, \mathcal{D}, \{f_\mu(x,d)\}_{\mu\in\mathbb{U}}), h$		
Time indices			
Discrete time	k	Continuous time	t
Discrete-valued states		**Continuous-valued states**	
State	ξ	State	x
State trajectory	$\zeta(k)$	State trajectory	$\mathbf{x}(k)$ / $\mathbf{x}(t)$
Input	μ	Input	u
Input sequence	$\sigma(k)$	Input sequence	$\mathbf{u}(k)$ / $\mathbf{u}(t)$
Output	γ	Output	y
Output sequence	$\Upsilon(k)$	Output sequence	$\mathbf{y}(k)$ / $\mathbf{y}(t)$
		Disturbance	d
		Disturbance sequence	$\mathbf{d}(k)$ / $\mathbf{d}(t)$
Sets and set operators			
Integers (non-negative)	$\mathbb{Z}$ ($\mathbb{Z}_+$)	Reals (non-negative)	$\mathbb{R}$ ($\mathbb{R}_+$)
Set $\{0, \ldots, N-1\}$	$[N]$	Indicator function of X	$\mathbb{1}_X$
Set closure	cl	Set interior	int
Least common multiplier	lcm	Convex hull	conv
Various math			
Flow of vector field f	$\phi_f(t, x; \mathbf{u}, \mathbf{d})$	Euclidean inner product	$\langle \cdot, \cdot \rangle$

(*continued on next page...*)

(*... continued from previous page*)			
Identity function	Id	r-Ball at x in p-norm	$\mathcal{B}_p(x, r)$
Linear temporal logic			
Logical and	$\wedge$	Logical or	$\vee$
Logical negation	$\neg$	Logical implication	$\Longrightarrow$
Temporal next	$\bigcirc$	Temporal until	$\mathbf{U}$
Temporal eventually	$\Diamond$	Temporal always	$\Box$

- Indexing in this thesis is **zero-based** [35]: for a vector $v \in \mathbb{R}^n$ the first element is v_0, and the last element is v_{n-1}.

Abstract

There is a need for controller design methodologies that enable early detection and elimination of unsafe designs. This thesis is about correct-by-construction synthesis methods—a collection of techniques that provide mathematical guarantees on correct behavior with respect to a formal specification. While these methods have attractive theoretical properties, there are fundamental scalability limitations that inhibit wide adoption; this work is concerned with ways of overcoming the scalability issue. In particular, three techniques for improving scalability are addressed.

Firstly, a specification-guided abstraction-refinement technique with two novel inventions is presented. As opposed to traditional uniform abstractions, this technique creates non-uniform abstractions that are iteratively refined in "promising" areas of the state space to create a parsimonious abstraction. The abstraction is also augmented with additional information in the form of *progress groups* that can encode transience properties of the underlying concrete system. Examples demonstrate that both these novelties can reduce the computational burden significantly.

Secondly, ways to decompose large problems into smaller ones while preserving correctness guarantees are considered. Since synthesis techniques typically scale exponentially with system dimension, decomposition can render otherwise intractable problems tractable. Two techniques for computation of separable invariant sets are presented; one for joint computation via LMIs, and one for iterative localized computation for systems with nonlinear interconnections.

The final part of the thesis presents a way to leverage problem structure in order to provide mathematical correctness guarantees on aggregate behavior for very large collections of systems. In particular, *counting problems* are introduced, which due to their symmetry admit a novel form of abstraction whose size is a fraction of that of a traditional naïve abstraction. The counting problem on the abstraction is solved via ILPs optimization.

Chapter 1.

Introduction

Control theory is the field of mathematics concerned with controlling a system so that it behaves in a desired way. Since its inception, the field has had a major impact in engineering applications ranging from efficient power distribution to the NASA mars rovers. While traditional control theory excels at determining equilibrium properties such as stability or reference tracking, techniques are lagging when it comes to performing more complex and dynamical tasks such as autonomous driving, air traffic control, or robotic walking, where out-of-equilibrium behavior is equally important. This constitutes a severe limitation for modern engineered systems, which are becoming increasingly complex while simultaneously being expected to perform more difficult tasks. Moreover, the systems often operate in the vicinity of humans which makes safe and correct behavior paramount.

Current control engineering practice largely relies on heuristic techniques such as PID controllers, which require extensive testing and tuning in order to "ensure" that the system meets its specifications. A design process such as the "design V" iterates over specifications and design, where an error encountered late in the process may cause significant delays. In essence, control design for complex systems is a time-consuming process that not only results in little to no intuition about the internal mechanisms of the system (thus rendering debugging or component replacement difficult), but also fails to provide actual guarantees on the final performance. The number of tests is necessarily finite and comprises only a fraction of the number of scenarios encountered when a product is deployed at a massive scale. Recent rare-event failures such as the Toyota "self-accelerating cars", or Boeing's power system issues with the Dreamliner jet, highlight the need for a design paradigm capable of providing mathematical performance guarantees, and of eliminating errors at an early stage in the process.

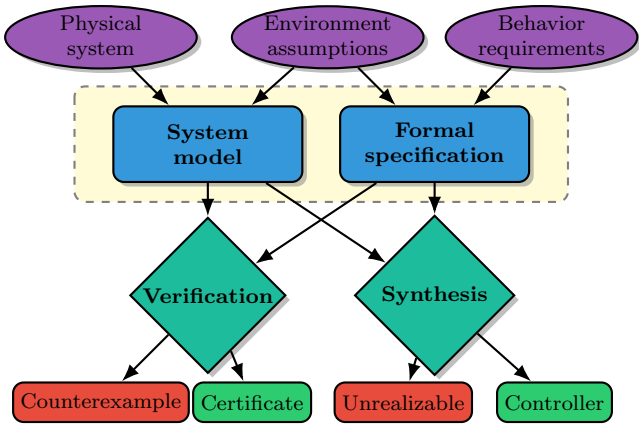

Figure 1.1: Illustration of steps involved in verification and synthesis.

An effort to address the shortcomings of traditional tools is the sub-field of *formal methods* in control. Formal methods have their roots in computer science, where they are used to provide guarantees against deadlocks and other undesired behavior in software. Also control software can be analyzed in a principled way by applying similar ideas to dynamical systems. A formal analysis can either lead to earlier detection of bugs and logical errors, which reduces costly testing and tuning, or to a mathematical guarantee on correct behavior, which increases the confidence that the module works as intended. More specifically, formal methods guarantee that a system fulfills a certain *formal specification* which specifies assumptions on the system environment and the desired system behavior. Having a component endowed with such explicit assumptions and guarantees may be beneficial also in other regards than error prevention. Firstly, the assumptions and guarantees serve as a clear demarcation of performance limits. One of the major barriers towards autonomous driving is the legal one: who is at fault if an autonomous car is involved in an accident? By introducing software that is mathematically guaranteed to function correctly under explicit conditions, a manufacturer can clearly specify the performance limits of its cars. Secondly, an assume-guarantee framework facilitates a modular controller design approach. While a holistic design may require a complete overhaul if a single component is replaced, a modular design with local assumptions and guarantees enables a component to be replaced as long as the new component can provide the same local guarantees under the same local assumptions.

The problems considered in research on formal methods in control can be classified into two categories: verification and synthesis. As illustrated in Figure 1.1 the inputs to both

types of problems are a system model and formal specification. The system model should capture the relevant dynamics of the physical system, and can also encode restrictions on environmental behavior, whereas the formal specification is formed from behavioral requirements and assumptions that are not captured in the system model[1]. In verification, the system model is closed-loop and the objective is to verify that its behavior meets the specification for *all possible executions*, or give a counterexample of violation that can guide debugging. The objective in synthesis is more ambitious: the system model is open-loop and the objective is to synthesize a controller such that the resulting closed-loop behavior meets the specification, i.e., is *correct-by-construction*.

This work is focused on the latter: correct-by-construction control synthesis for continuous systems. There is a vast literature on synthesis for finite-state systems, and many associated finite synthesis tools have been developed. One natural way to attack synthesis for continuous systems is therefore to construct a finite abstraction and leverage existing tools and techniques on the generated abstraction. However, there are also direct approaches that do not require system abstraction; typically such methods rely on computation and analysis of sets in the continuous state space. Synthesis methods have been successful at controlling low-dimensional continuous systems (up to around five states), but higher-dimensional systems pose a challenge: both abstraction and set manipulation scale badly with dimension. Several approaches exist to mitigate or overcome the curse of dimensionality; this thesis presents work in three directions.

Summary of contributions and thesis overview. Chapter 2 deals with non-uniform methods for constructing abstractions. The main contributions in the chapter are as follows:

- It is shown that the number of discrete states in an abstraction can be reduced by orders of magnitude by utilizing a *non-uniform* abstraction procedure that refines the abstraction only when motivated by the underlying finite synthesis procedure.
- Abstractions are typically transition systems that encode possible transitions of the concrete system. For a coarse abstraction there can be significant *spurious behavior*—trajectories that are possible in the partition but not in the underlying system. A

[1]Whether environmental assumptions are encoded in the system model or as part of the specification is a design choice.

method to reduce spurious behavior by augmenting the abstraction with *progress groups* that encode liveness properties is proposed.

The topic of Chapter 3 is decomposition of a large problem into smaller subproblems as a means of reducing the computational burden. Consider a synthesis method that is exponential in the problem "size". Loosely speaking, if a problem of size N can be decomposed into k sub-problems of equal size N/k, then the decomposed problem has complexity $\mathcal{O}(ke^{N/k})$ rather than $\mathcal{O}(e^N)$ for the full-sized problem. A principled way to decompose a formal synthesis problem is to find separable controlled invariant sets; two methods for computing such sets are presented:

- Synthesis of symmetric polyhedral separable controlled invariant sets for linear systems via a centralized collection of Linear Matrix Inequalities.
- Iterative computation of polyhedral separable controlled invariant sets for linear systems with nonlinear interconnections.

Both methods are customizable with regards to information availability; the effect of one subsystem on another can be treated as either measurable or non-measurable, where the latter is more conservative but does not require on-line subsystem communication.

Off-the-shelf formal synthesis tools and methods are typically broad in terms of what dynamics and specifications they consider, but do not exploit structural properties that are often present in synthesis problems. In Chapter 4 a special class of highly symmetrical problems is introduced. The main contributions of the chapter are as follows:

- The *counting problem* is defined for arbitrary transition systems and it is shown that the inherent symmetries can be exploited to enable synthesis for large-scale systems.
- An abstraction procedure that transforms a continuous-state counting problem into a finite-state one, together with results relating the solvability of the two problems.
- A solution method for finite-state counting problems via Integer Linear Programs (ILPs).

All the above methods are illustrated with relevant numerical examples. The thesis is concluded in Chapter 5 with an attempt to outline future research directions.

Acknowledgment of previous publications. Parts of the results in this thesis have previously appeared as published work. For Chapter 2, the relevant publications are [90] and [94]; for Chapter 3, [93] and [119]; and for Chapter 4, [89] and [88]. In addition, the publications [86, 87, 91, 92, 115, 116] contain related results developed in parallel.

Chapter overview. The goal of the rest of this chapter is to introduce the main ideas underpinning formal control synthesis methods both for finite- and continuous-state systems, and to provide an overview of earlier work. While the main focus is on synthesis for continuous systems, finite synthesis theory is a crucial ingredient and is therefore a cornerstone of the discussion. Firstly, finite synthesis algorithms can be generalized to continuous-state systems provided that the appropriate reachability computations can be performed relatively efficiently. Secondly, many methods rely on finite system abstractions that reduce a continuous synthesis problem into a finite one, allowing existing tools for finite synthesis to be leveraged. Finally, finite synthesis algorithms are more intuitive than their continuous counterparts while exhibiting the same fundamental principles.

A formal synthesis problem consists of a formal specification and a formal system model; the objective is to control the system so that its executions satisfy the specification. The outcome of a synthesis algorithm is only as good as the formal specification and the system model, it is therefore crucial that the specification is adequately stated and that the system model captures relevant aspects of the dynamics. Mathematical reasoning requires precise definitions of both formal specifications and systems, which is the purpose of sections 1.1 and 1.2 below.

In the subsequent Section 1.3 finite synthesis theory is summarized, and fixed-point characterizations of the synthesis problem for different types of specifications are provided. In principle, the same characterizations are valid for any discrete-time system, and can be computed to arbitrary precision if reachability computations are possible. Continuous-time PDE generalizations of the fixed points are also provided when possible.

Finally, theory relevant to abstractions is introduced; a useful abstraction must possess the property that satisfaction of the specification in the abstraction implies satisfaction also in the concrete system. The theory of simulation relations introduced in Section 1.4 is a principled way to express such properties.

1.1. Formal Specifications

As opposed to textual specifications that are often encountered in practice, a formal specification must have a precise mathematical interpretation. Due to the ambiguous nature of the English language, textual specifications are often imprecise. Therefore just the formalization step (c.f. Figure 1.1) may be valuable, since it removes ambiguities that may be interpreted differently by different persons. The automotive safety standard ISO 26262 indeed recommends formal treatment of specifications, even if verification against these specifications is only performed through simulations (so called *semi-formal* verification).

Formal specifications are expressed in a formal language with precisely defined grammar and semantics. The most widely known formal language is Linear Temporal Logic (LTL) which is introduced briefly in the following; see e.g. [12] for a more complete overview. LTL combines traditional logical operators with temporal operators that together are able to express a wide range of practical properties. The language is defined over a set of *atomic propositions* AP; in this work it is without loss of generality assumed that each atomic proposition is identified with a subset of an output space $\mathcal{Y}$. The set of states in $\mathcal{Y}$ where the atomic proposition a holds is denoted $[\![a]\!]_{\mathcal{Y}}$ below. A formula is interpreted over infinite strings $\Upsilon = \Upsilon(0)\Upsilon(1)\Upsilon(2)\ldots$ of outputs in $\mathcal{Y}$. The grammar of LTL is recursively defined in terms of a given set of atomic propositions AP as follows:

1. `True` is an LTL formula.
2. An atomic proposition $a \in AP$ is an LTL formula.
3. Given LTL formulas φ and ψ, $\neg\varphi$, $\varphi \wedge \psi$, $\bigcirc\varphi$ and $\varphi \,\mathbf{U}\, \psi$ are LTL formulas.

Next the semantics of LTL are defined, which specify how an LTL formula should be interpreted.

Definition 1.1. *Let $\Upsilon = \Upsilon(0)\Upsilon(1)\Upsilon(2)\ldots$ be an output trajectory and let φ be an LTL formula. The notation $(\Upsilon, k) \models \varphi$ denotes that Υ **satisfies** φ **at time** k, where satisfaction is recursively defined as follows:*

- $(\Upsilon, k) \models$ `True`.
- *If $\varphi = a$ for $a \in AP$, then $(\Upsilon, k) \models \varphi$ if and only if $\Upsilon(k) \in [\![a]\!]_{\mathcal{Y}}$.*
- *If $\varphi = \neg\psi$, then $(\Upsilon, k) \models \varphi$ if and only if $(\Upsilon, k) \not\models \psi$.*

time:	$k=0$		$k=1$		$k=2$		$k=3$		$k=4$		$\cdots$
$\bigcirc\varphi$:	$\cdot$	—	φ	—	$\cdot$	—	$\cdot$	—	$\cdot$	—	$\cdots$
$\varphi\,\mathbf{U}\,\psi$:	φ	—	φ	—	φ	—	ψ	—	$\cdot$	—	$\cdots$
$\Diamond\varphi$:	$\neg\varphi$	—	$\neg\varphi$	—	$\neg\varphi$	—	φ	—	$\cdot$	—	$\cdots$
$\Box\varphi$:	φ	—	φ	—	φ	—	φ	—	φ	—	$\cdots$

Figure 1.2: Temporal operators of Linear Temporal Logic. The diagram shows satisfying traces for the basic temporal operators in LTL. The *next* specification $\bigcirc\varphi$ holds at time $k=0$ if φ holds at time $k=1$. An *until* specification $\varphi\,\mathbf{U}\,\psi$ holds at time $k=0$ if φ holds at all time steps until ψ holds, and ψ must hold at some time step in the future. *Eventually* $\Diamond\varphi$ requires that φ holds at some time step in the future, while *always* $\Box\varphi$ requires φ to hold for all $k\in\mathbb{Z}_+$.

- *If* $\varphi=\psi_1\wedge\psi_2$, *then* $(\Upsilon,k)\models\varphi$ *if and only if* $(\Upsilon,k)\models\psi_1$ *and* $(\Upsilon,k)\models\psi_2$.
- *If* $\varphi=\bigcirc\psi$, *then* $(\Upsilon,k)\models\varphi$ *if and only if* $(\Upsilon,k+1)\models\psi$.
- *If* $\varphi=\psi_1\,\mathbf{U}\,\psi_2$, *then* $(\Upsilon,k)\models\varphi$ *if and only if there exists* $k\in\mathbb{N}$ *such that* $(\Upsilon,k)\models\psi_2$ *and* $(\Upsilon,k')\models\psi_1$ *for all* $0\leq k'<k$.

An output sequence Υ *is said to* ***satisfy*** φ, *written* $\Upsilon\models\varphi$, *if* $(\Upsilon,0)\models\varphi$.

Not ($\neg$) and *and* ($\wedge$) are standard logical negation and conjunction, whereas *next* ($\bigcirc$) and *until* ($\mathbf{U}$) are temporal operators named in accordance with the semantics. In addition to these fundamental operators, the derived operators $\texttt{False}=\neg\texttt{True}$, *or* $\varphi\vee\psi=\neg(\neg\varphi\wedge\neg\varphi)$, *eventually* $\Diamond\varphi=\texttt{True}\,\mathbf{U}\,\varphi$, and *always* $\Box\varphi=\neg\Diamond\neg\varphi$ can be defined[2]. Figure 1.2 illustrates the semantics of the basic temporal operators.

LTL satisfaction can also be defined for a continuous-time output trajectory—provided that the "next" operator ($\bigcirc$) that lacks interpretation in continuous time is not present in the formula. The fragment of LTL without the next operator is commonly denoted as $\text{LTL}_{\setminus\bigcirc}$.

Definition 1.2. *Let* φ *be a* $\text{LTL}_{\setminus\bigcirc}$ *specification and let* $\mathbf{y}(t):\mathbb{R}_+\to\mathcal{Y}$ *be a continuous-time output trajectory. Then* $\mathbf{y}\models\varphi$ *if and only if there exists a strictly increasing sequence* $\{t_k\}_{k=0}^{\infty}$ *in* $\mathbb{R}_+$ *with the following properties:*

- $\lim_{k\to\infty}t_k=\infty$.

[2]In the literature additional derived operators such as *weak until* and *release* sometimes appear. LTL can also be defined in terms of a different set of fundamental operators.

- *For all* $a \in AP$, *either* $\mathbf{y}([t_k, t_{k+1})) \subset [\![a]\!]_{\mathcal{Y}}$, *or* $\mathbf{y}([t_k, t_{k+1})) \subset ([\![a]\!]_{\mathcal{Y}})^C$.
- *The discrete output sequence* $\Upsilon = \mathbf{y}(t_0)\mathbf{y}(t_1)\mathbf{y}(t_2)\ldots$ *satisfies* φ, *i.e.* $\Upsilon \models \varphi$.

1.2. System Descriptions

Attention is now shifted to the other component of a formal synthesis problem: the system model. These models can be of different levels of sophistication; in this thesis the following types of systems are considered:

- Finite state, discrete time (FS/DT),
- continuous state, discrete time (CS/DT),
- continuous state, continuous time (CS/CT),
- switched systems.

Different representations are needed for these different types of systems. The canonical representation of a (FS/DT) system is a *transition system*, which is the first system introduced in Section 1.2.1. Although also (CS/DT) systems as well as sampled continuous-time systems [74] can be interpreted as transition systems—which is useful in analysis—the standard compact representations given below are often preferable.

1.2.1. Transition Systems

A transition system serves as a model for finite discrete-time systems.

Definition 1.3. *A* ***transition system*** *is a tuple* $\mathcal{T} = (\mathbb{X}, \mathbb{U}, \longrightarrow, h)$, *where*

- $\mathbb{X}$ *is a set of states.*
- $\mathbb{U}$ *is a set of control inputs, or* actions.
- $\longrightarrow \subset \mathbb{X} \times \mathbb{U} \times \mathbb{X}$ *is a set of transitions;*
- $h : \mathbb{X} \longrightarrow \mathcal{Y}$ *is an output map.*

*If both the set of states and the set of control inputs are finite, $\mathcal{T}$ is called a **finite** transition system (FTS). If in addition the transitions are deterministic, i.e. $(\xi, \mu, \xi') \in \longrightarrow$ and $(\xi, \mu, \xi'') \in \longrightarrow$ implies $\xi' = \xi''$, $\mathcal{T}$ is called a **Deterministic** Finite Transition System (DFTS).*

As an intuitive shorthand notation for transitions, $\xi \xrightarrow{\mu} \xi'$ will be used to indicate that $(\xi, \mu, \xi') \in \longrightarrow$, and ξ' will be called a μ-*successor* of ξ, and ξ a μ-*predecessor* of ξ'. In a non-deterministic system a state may have several μ-successors.

Definition 1.4. *A **controller** for a transition system $\mathcal{T}$ is a tuple $\pi = (\pi_{\mathbb{X}}, \pi_{\mathbb{M}}, \mathbb{M}, m_0)$, where*

- $\pi_{\mathbb{X}} : \mathbb{X} \times \mathbb{M} \longrightarrow 2^{\mathcal{U}}$ *maps the current system and memory state to a set of control inputs.*
- $\pi_{\mathbb{M}} : \mathbb{X} \times \mathbb{M} \longrightarrow \mathbb{M}$ *updates the memory state.*
- $\mathbb{M}$ *is a set of controller memory states.*
- $m_0 \in \mathbb{M}$ *is the initial memory state.*

When the set $\mathbb{M}$ is finite, the controller is called a **finite-memory controller**.

Definition 1.5. *A **trajectory** of a transition system $\mathcal{T}$ is a sequence of states $\zeta = \zeta(0)\zeta(1)\zeta(2)\ldots$ with the property that $\zeta(k) \xrightarrow{\sigma(k)} \zeta(k+1)$ for some $\sigma(k) \in \mathbb{U}$, for all $k \geq 0$. If $\sigma(k)$ is generated by a controller π, i.e. $\sigma(k) \in \pi(\zeta(k), m(k))$, then ζ is a π-**controlled trajectory** of $\mathcal{T}$. If ζ is of infinite length it is called **maximal**.*

With these definitions in place it is time to define what it means for a trajectory of the system to satisfy a specification.

Definition 1.6. *A trajectory ζ of $\mathcal{T}$ is said to **satisfy** the specification φ if*

$$\Upsilon = h(\zeta(0))h(\zeta(1))h(\zeta(2))\ldots \models \varphi. \tag{1.1}$$

The goal of control synthesis is to construct a controller that generate trajectories with this property; algorithms for doing so rely on *backwards reachability computations*. For a given set X, the backwards reachable set comprises all states from where a transition to X can be enforced regardless of non-determinism.

Definition 1.7. *Let $\mathcal{T}$ be a transition system and consider a set $X \subset \mathbb{X}$. The* ***backwards reachable set from*** *X is the set*

$$\mathrm{Pre}^{\mathcal{T}}(X) = \{\xi \in \mathbb{X} : \exists \mu \in \mathbb{U}\ \forall (\xi, \mu, \xi') \in \longrightarrow,\ \xi' \in X\}. \tag{1.2}$$

1.2.2. Discrete-Time Systems

In a transition system there is no spatial relationship between the different states and control inputs. In addition, the non-determinism lacks structure. This is a shortcoming in analysis of non-finite systems where geometrical structure is present. Instead the following standard model is appropriate for continuous-state, discrete-time systems:

$$\Sigma_{DT} : \begin{array}{l} \mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k)), \\ \mathbf{x}(k) \in \mathcal{X} \subset \mathbb{R}^{n_x},\ \mathbf{u}(k) \in \mathcal{U} \subset \mathbb{R}^{n_u}, \mathbf{d}(k) \in \mathcal{D} \subset \mathbb{R}^{n_d}. \end{array} \tag{1.3}$$

For an output function $h : \mathcal{X} \longrightarrow \mathcal{Y}$ the system description can be captured in a tuple $\Sigma_{DT} : (\mathcal{X}, \mathcal{U}, \mathcal{D}, f, h)$. It is easy to see that the same system can be described as a (non-finite) transition system $\mathcal{T}(\Sigma_{DT}) = (\mathcal{X}, \mathcal{U}, \longrightarrow, h)$, where for all $x, u, x' \in \mathcal{X} \times \mathcal{U} \times \mathcal{X}$,

$$(x, u, x') \in \longrightarrow \quad \text{if and only if } \exists d \in \mathcal{D} \text{ s.t. } x' = f(x, u, d).$$

Via this connection Definitions 1.4-1.7 can be adopted for controllers, trajectories, satisfaction and backwards reachable sets also for this class of systems.

Reachability computations for discrete-time systems

For a Finite Transition System (FTS) $\mathcal{T}$, $\mathrm{Pre}^{\mathcal{T}}(X)$ is conceptually easy to calculate since all subsets are finite objects: for all states ξ one simply needs to determine whether there exists a $\mu \in \mathbb{U}$ such that all μ-successors of ξ are in X. However, for a continuous-state system the task becomes more difficult. The backwards reachable set of a discrete-time system can be written

$$\mathrm{Pre}^{\Sigma_{DT}}(X) = \{x \in \mathcal{X} : \exists u \in \mathcal{U}\ \forall d \in \mathcal{D},\ f(x, u, d) \in X\}. \tag{1.4}$$

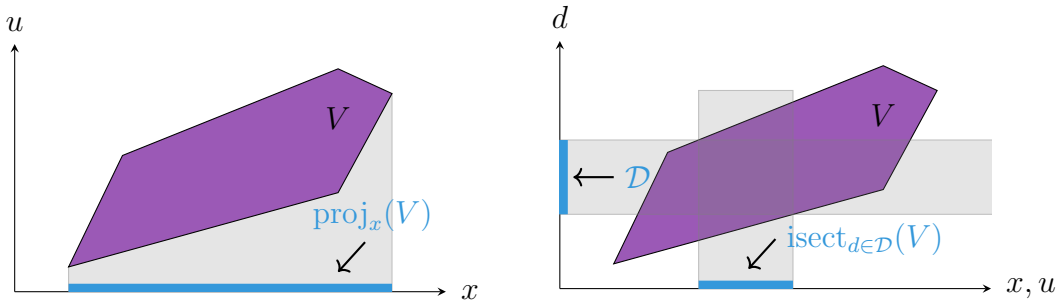

Figure 1.3: Projection and intersection operators comprising the backwards reachable set operation.

In essence, evaluation of $\text{Pre}^{\Sigma_{DT}}(X)$ requires eliminating the two quantifiers in front of u and d. The quantifier elimination can be written as a two-stage operation:

$$\text{proj}_x \circ \text{isect}_{d\in\mathcal{D}} \left(f^{-1}(X)\right), \tag{1.5}$$

where each operator eliminates one quantifier. Here $\text{isect}_{d\in\mathcal{D}} : 2^{\mathcal{X}\times\mathcal{U}\times\mathcal{D}} \longrightarrow 2^{\mathcal{X}\times\mathcal{U}}$ is defined as follows:

$$\text{isect}_{d\in\mathcal{D}}(V) = \{(x,u) \ : \ \forall d \in \mathcal{D}, (x,u,d) \in V\}.$$

As can be seen, this mapping reduces the "dimension" of a set by mapping it to a subset of $\mathcal{X}\times\mathcal{U}$. Similarly, $\text{proj}_x : 2^{\mathcal{X}\times\mathcal{U}} \longrightarrow 2^{\mathcal{X}}$ is the projection onto $\mathcal{X}$, i.e.

$$\text{proj}_x(V) = \{x \in \mathcal{X} \ : \ \exists u \in \mathcal{U}, \ (x,u) \in V\}.$$

The operations are illustrated in Figure 1.3 and are in general difficult to evaluate. In addition, it is advantageous for synthesis purposes to be *set-closed*, meaning that $\text{Pre}^{\Sigma_{DT}}(X)$ should be a set of the same class as X. This is often not true, i.e., an ellipse may not be mapped into another ellipse by $\text{Pre}^{\Sigma_{DT}}$.

Affine linear systems. One example where continuous-state reachable sets are both computable and set-closed are affine systems together with polyhedral sets, i.e., a system $\Sigma_{DT} : \mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) + E\mathbf{d}(k)$ with input and disturbance constraints $\mathcal{U} = \{u \ : \ H_u u \leq h_u\}$, $\mathcal{D} = \{d \ : \ H_d d \leq h_d\}$. If also the set $X = \{H_x x \leq h_x\}$ is a

polyhedron, then (1.5) can be written as

$$\mathrm{proj}_x \circ \mathrm{isect}_{d\in\mathcal{D}} \left(\left\{ (x,u,d) \;:\; \begin{bmatrix} H_xA & H_xB & H_xE \\ 0 & H_u & 0 \\ 0 & 0 & H_d \end{bmatrix} \begin{bmatrix} x \\ u \\ d \end{bmatrix} \leq \begin{bmatrix} h_x \\ h_u \\ h_d \end{bmatrix} \right\} \right). \tag{1.6}$$

Firstly, the inner operator $\mathrm{isect}_{d\in\mathcal{D}}$ can be evaluated by considering the *worst-case* disturbance effect $\max_{d\in\mathcal{D}} H_xEd$. This expression should be evaluated row-wise, i.e.,

$$\left[\max_{d\in\mathcal{D}} H_xEd \right]_i = \max_{d\in\mathcal{D}} \left[H_xE\right]_i d, \tag{1.7}$$

which amounts to solving a linear program for each row. Incorporating into (1.6) yields

$$\mathrm{Pre}^{\Sigma_{DT}}(X) = \mathrm{proj}_x \left(\left\{ (x,u) \;:\; \begin{bmatrix} H_xA & H_xB \\ 0 & H_u \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} \leq \begin{bmatrix} h_x - \max_{d\in\mathcal{D}} H_xEd \\ h_u \end{bmatrix} \right\} \right).$$

Secondly, the projection proj_x can be computed using e.g. Fourier-Motzkin elimination implemented in tools such as the Multi-Parametric Toolbox [57]. In Chapter 3 alternative ways to compute $\mathrm{Pre}^{\Sigma_{DT}}$ for linear systems are discussed.

Polynomial systems. Even if exact reachability computations are intractable, also conservative backwards reachable sets $\widehat{\mathrm{Pre}}^{\Sigma_{DT}}(X)$ such that $\widehat{\mathrm{Pre}}^{\Sigma_{DT}}(X) \subset \mathrm{Pre}^{\Sigma_{DT}}(X)$ can be used for sound (but not complete) synthesis. Such guaranteed inner approximations are for instance possible for polynomial autonomous systems with disturbance, when X is a semi-algebraic set [71].

1.2.3. Continuous-Time Systems

As a model of continuous state/continuous time (CS/CT) systems, controlled ordinary differential equations of the form

$$\Sigma_{CT}: \begin{array}{l} \dfrac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = f(\mathbf{x}(t),\mathbf{u}(t),\mathbf{d}(t)), \\ \mathbf{x}(t)\in\mathcal{X}\subset\mathbb{R}^{n_x},\ \mathbf{u}(t)\in\mathcal{U}\subset\mathbb{R}^{n_u},\ \mathbf{d}(t)\in\mathcal{D}\subset\mathbb{R}^{n_d}, \end{array} \tag{1.8}$$

are used. A **controller** π for Σ_{CT} is a rule that determines the input $\mathbf{u}(t)$ based on the current state $\mathbf{x}(t)$ and potential discrete or continuous internal controller states. A controller that only depends on $\mathbf{x}(k)$ is called a (state-)feedback controller. A **trajectory** of Σ_{CT} is a trajectory $\mathbf{x} : I \to \mathcal{X}$ for an interval $I \subset \mathbb{R}_+$ that is absolutely continuous and such that

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \mathbf{d}(t)) \quad \text{almost everywhere on } I$$

for some input and disturbance trajectories $\mathbf{u}$ and $\mathbf{d}$. If $\mathbf{u}$ is produced by a controller π, then the trajectory is π**-controlled** and if $I = \mathbb{R}_+$ the trajectory is called **maximal**. A related concept is the **flow operator** ϕ_f which given input and disturbance sequences maps an initial state x and time t to the (weak) solution of (1.8). It has the following properties:

$$\phi_f(0, x; \mathbf{u}, \mathbf{d}) = x, \quad \frac{\mathrm{d}}{\mathrm{d}t}\phi_f(t, x; \mathbf{u}, \mathbf{d}) = f(\phi_f(t, x; \mathbf{u}, \mathbf{d}), \mathbf{u}(t), \mathbf{d}(t)) \quad \text{for almost all } t.$$

Remark 1.1. *Absolutely continuous solutions that satisfy an ODE almost everywhere are known as a Carathéodory solutions. This type of weak solution enjoys existence and uniqueness properties in more general instances than traditional solutions do (e.g., discontinuous right-hand sides) [128, p. 121].*

For an $\mathrm{LTL}_{\setminus\bigcirc}$ specification φ and an output map $h : \mathcal{X} \to \mathcal{Y}$, a trajectory $\mathbf{x}(t)$ is said to **satisfy** φ if $h \circ \mathbf{x} : \mathbb{R}_+ \to \mathcal{Y}$ satisfies φ, i.e. $h \circ \mathbf{x} \models \varphi$.

Infinitesimal backwards reachability operator. In the following a concept analogous to the backwards reachable set operator for discrete-time systems is introduced in an attempt to highlight the similarities between discrete and continuous synthesis algorithms. It is convenient to represent a continuous set X as the zero super-level set of a function h_X, i.e. $X = \{x \in \mathcal{X} : h_X(x) \geq 0\}$. If h_X is differentiable, for a small time $\mathrm{d}t$ the infinitesimal version of the backwards reachable set from X is

$$\begin{aligned} &\{x \ : \ \exists u \in \mathcal{U} \ \forall d \in \mathcal{D} \ h_X\left(t, x + f(x, u, d)\mathrm{d}t\right) \geq 0\} \\ &\quad = \left\{x \ : \ \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} h_X(t, x) + \left\langle \frac{\partial}{\partial x} h_X(t, x), f(x, u, d) \right\rangle \mathrm{d}t \geq 0 \right\}. \end{aligned} \tag{1.9}$$

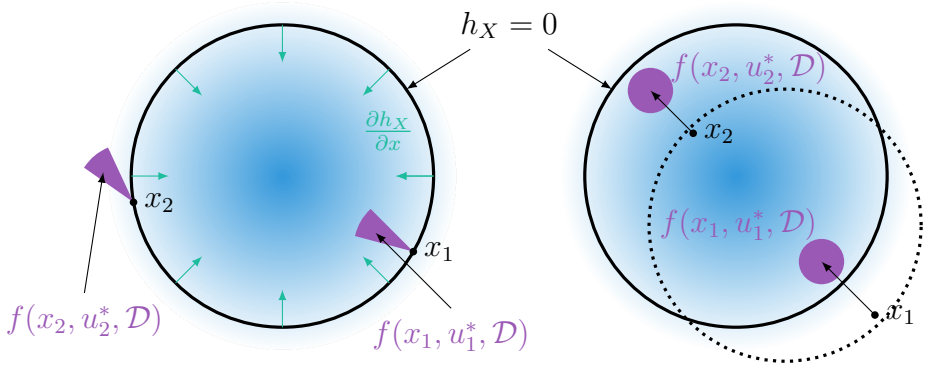

Figure 1.4: Backwards reachability in continuous and discrete time. The figures illustrate of how the continuous-time level set function (left) and discrete-time level set function (right) are propagated by the backwards reachability operators (1.9) **and** (1.11)**. The intensity of the blue color corresponds to the magnitude of a level set function h_X that is positive within the black circle.**
In continuous time (left), the level set function h_X increases at x_1 where it is possible to select a control u_1^* such that the cone of resulting flow vectors $f(x_1, u_1^*, \mathcal{D})$ is aligned with the gradient $\frac{\partial h_X}{\partial x}$. Conversely, h_X decreases at x_2 where the alignment is not possible.
In discrete time (right), the updated value function is positive (dotted circle) at points x where it is possible to select a u^* such that $h_X(f(x_1, u^*, \mathcal{D})) \geq 0$.

h_X can thus be propagated in time to obtain the following continuous counterpart of the backwards reachable set:

$$h_X\left(t + \mathrm{d}t, x\right) = h_X(t, x) + \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} \left\langle \frac{\partial h_X}{\partial x}(t, x), f(x, u, d) \right\rangle \mathrm{d}t.$$

Using the notation $F^{\Sigma_{CT}}(x, p) = \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} \langle p, f(x, u, d) \rangle$, the evolution of h_X under the system Σ_{CT} is in the limit $\mathrm{d}t \to 0$ described by the PDE

$$\frac{\partial h_X}{\partial t} = F^{\Sigma_{CT}}\left(x, \frac{\partial h_X}{\partial x}(t, x)\right). \tag{1.10}$$

In the autonomous case $\dot{x} = f(x)$ this PDE is known as the (backwards) *transport equation* and can be solved via the method of characteristics [38, p. 97]. In general the right-hand side is discontinuous and solutions may not exist in the traditional sense; an appropriate type of weak solution is *viscosity solutions* [29].

Remark 1.2. *To further highlight the connection between discrete-time and continuous-*

time backwards reachability, remark that the discrete-time version for a system Σ_{DT} : $\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k), \mathbf{d}(k))$ *can be written as mapping* $(\mathbb{X} \to \mathbb{R}) \longrightarrow (\mathbb{X} \to \mathbb{R})$ *that maps a function* h_X *that assigns a real number to each state in* $\mathbb{X}$ *to a new function* $h_{\mathrm{Pre}^{\Sigma_{DT}}(X)}$ *of the same kind.*

$$h_X \mapsto h_{\mathrm{Pre}^{\Sigma_{DT}}(X)}, \quad h_{\mathrm{Pre}^{\Sigma_{DT}}(X)}(x) = \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} h_X(f(x,u,d)). \tag{1.11}$$

It can be seen that if $X = \{x : h_X(x) \geq 0\}$*, then* $\mathrm{Pre}^{\Sigma_{DT}}(X) = \{x : h_{\mathrm{Pre}^{\Sigma_{DT}}(X)}(x) \geq 0\}$*. Figure 1.4 illustrates the two cases.*

1.2.4. Switched Systems

Switched systems can be defined in both discrete and continuous time. A switched system is a collection of uncontrolled systems that are either all of the form (1.3), or all of the form (1.8). Each such subsystem describes a *mode* of the switched system.

Definition 1.8. *A* ***switched system*** *is a tuple* $\Sigma_{Sw} = (\mathcal{X}, \mathbb{U}, \mathcal{D}, \{f_\mu\}_{\mu \in \mathbb{U}}, h)$*, where* $\mathbb{U}$ *is a finite set of* ***modes****.*

The evolution of Σ_{Sw} is governed by

$$D^+\mathbf{x}(t) = f_{\sigma(t)}(\mathbf{x}(t), \mathbf{d}(t)), \quad \mathbf{x}(t) \in \mathcal{X},\ \sigma(t) \in \mathbb{U},\ \mathbf{d}(t) \in \mathcal{D}, \tag{1.12}$$

where

$$D^+\mathbf{x}(t) = \begin{cases} \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t), & \text{for continuous time,} \\ \mathbf{x}(t+1), & \text{for discrete time.} \end{cases}$$

A **trajectory** of a switched system is a function $\mathbf{x}(t) : I \to \mathcal{X}$ for a (continuous or discrete) interval I that is piecewise assembled from trajectories of the underlying continuous-time or discrete-time systems. It is called **maximal** if the interval I is unbounded. A **switching protocol** for a switched system is a partial function $\pi : \mathcal{X} \times \mathbb{M} \longrightarrow \mathbb{U}$ together with an internal state update function $\pi_{\mathbb{M}}$. The function π maps an internal memory state $m(t) \in \mathbb{M}$ and system state $\mathbf{x}(t) \in \mathcal{X}$ to an input $\sigma(t) \in \mathbb{U}$, and the memory $m(t)$ follows some internal memory dynamics described by $\pi_{\mathbb{M}}$.

1.3. Synthesis Algorithms

Having introduced formal specifications and system models, the synthesis problem can be stated.

Problem 1.1. *Given a system Σ of one of the forms in Section 1.2, and an LTL specification φ, synthesize a controller π and a* ***winning set*** $\mathrm{Win}^{\Sigma}(\varphi)$ *with the property that all π-controlled trajectories of Σ originating in $\mathrm{Win}^{\Sigma}(\varphi)$ satisfy φ.*

The synthesis problem exhibits a duality between input and disturbance that arises from the following dichotomy:

1. There exists a way to select inputs such that for all disturbances, the specification φ is enforced.

2. For all ways to select inputs, there exists a disturbance such that $\neg\varphi$ is enforced.

This can be interpreted from a game-theoretic perspective, where player 1 controls the input and player 2 controls the disturbance, and player 1 acts first (and thus has the first-mover disadvantage). The objective of player 1 is to enforce satisfaction of φ, while player 2 tries to achieve $\neg\varphi$. Since the objectives are mutually exclusive, exactly one player has a winning strategy from any initial state[3]. As a consequence, interchanging quantifiers is equivalent to taking the complement of the winning set which is useful for reasoning about winning sets, but not for constructing a controller.

LTL synthesis for a FTS $\mathcal{T}$ is well understood; the general problem can be solved by representing the specification φ as a finite system $\mathcal{T}_\varphi$ and reason about reachability on the product system $\mathcal{T} \times \mathcal{T}_\varphi$. In particular, the winning set can be characterized as a fixed point in the state space of $\mathcal{T} \times \mathcal{T}_\varphi$ which can be found by iteratively applying $\mathrm{Pre}^{\mathcal{T}\times\mathcal{T}_\varphi}$, while simultaneously extracting a controller, see [12, 17] for details. While the theory of discrete synthesis is well understood, practical tools are seeing continual improvements [59].

Moving to systems with continuous state spaces the situation immediately worsens. The problem of determining whether a subset of the state space is reachable, which is a special case of synthesis, is known to be undecidable for relatively simple hybrid systems [56], although positive results also exist [69]. The *Skolem Pisot problem* is closely related, for which decidability is still open [15]. These negative results suggest that a sound and

[3]For the formal result about determinacy of two-player games with perfect information, see e.g. [85].

complete synthesis algorithm is impossible to construct for most classes of systems, and that the best case scenario is an algorithm that approximates the winning set with some convergence guarantee. Ideally, the synthesis problem should be solved for a controlled system that is subject to disturbance and the solution method should compute either the exact winning set or an inner approximation. Another aspect is the division between off-line and on-line computational burden. For practical usability the on-line computation must be light enough to be reliably executed at sample rate appropriate for the system at hand. In general there is a trade-off between off-line computational burden, memory requirements, and on-line computational burden. To summarize the following are desirable properties of a synthesis method:

1. It is correct-by-construction (as opposed to approximately correct).
2. It computes the exact winning set or provides convergence guarantee.
3. It treats general system classes (ideally, nonlinear hybrid systems with input and exogenous disturbance).
4. It has manageable on-line computational complexity.

Since it is very difficult to simultaneously incorporate all these characteristics, some methods study autonomous systems, do not consider disturbance, or provide approximations that are either outer approximations or just approximations.

An LTL specification can express a wide range of desired temporal properties and mandate very complex behaviors, as a result the algorithmic complexity of LTL synthesis for finite systems is double exponential in the size of the LTL formula [98, 99] which makes synthesis intractable for large specifications and/or systems. However, specifications encountered in practice often fall into one of a handful of categories; too complex specifications are impossible to interpret even for an expert and are therefore of limited value. One useful subset of LTL with favorable complexity is the Generalized Reactivity (1) ($GR(1)$) fragment. Synthesis in $GR(1)$ is known to have complexity that is quadratic both in the size of the state space and in the formula length [21]. Another advantage of studying fragments of LTL is that synthesis algorithms that operate directly on the state space rather than on a product system can be constructed.

Below a FTS $\mathcal{T} = (\mathbb{X}, \mathbb{U}, \longrightarrow, h)$ is considered in order to illustrate how winning sets of certain specifications can be characterized by defining fixed points in the finite state space

$\mathbb{X}$. Computing the winning set is one part of Problem 1.1, the other part is construction of the controller. It turns out that the controller can often be constructed as a by-product in the computation of the winning set, as that computation necessarily must take controls into account. This is the case for the fixed points below; a controller can be constructed by extracting the μ's selected in the evaluations of $\mathrm{Pre}^{\mathcal{T}}$ and mapping them into a hierarchy of reachability objectives derived from the fixed points. For this reason controller extraction is disregarded below; focus is on computation of the winning set.

Notation is borrowed from μ-calculus [68] for succinct expression of fixed points. Let $\kappa : 2^{\mathbb{X}} \longrightarrow 2^{\mathbb{X}}$ be a mapping that is monotone with respect to set inclusion, i.e., $V \subset W \implies \kappa(V) \subset \kappa(W)$. Due to monotonicity, repeated applications of κ necessarily converge due to finiteness of $\mathbb{X}$.

Definition 1.9. *The* ***greatest fixed point of*** κ*, written* $\nu V\ \kappa(V)$*, is the value after convergence of the set sequence*

$$V_0 = \mathbb{X}, \quad V_{k+1} = \kappa(V_k).$$

Correspondingly, the ***smallest fixed point of*** κ*, written* $\mu V\ \kappa(V)$*, is the value after convergence of*

$$V_0 = \emptyset, \quad V_{k+1} = \kappa(V_k).$$

Fixed points are themselves set-valued maps that map parameters of the mapping κ into the resulting fixed point. It is logical to define higher-order fixed points and talk about **level** k **fixed points**: a level 0 fixed point is defined in terms of a standard set-valued mapping, a level 1 fixed point is defined in terms of a level 0 fixed point, and so on[4].

A fixed point can be seen as a short-hand notation for writing an algorithm; each fixed point below can be converted into an algorithm with nested applications of the involved mapping, where the level of nesting is equal to the fixed-point level. Winning sets expressed as level 0 fixed points also have an interesting continuous analogue in the form of a PDE based on (1.10). In the following $[\![a]\!]$ denotes the subset of the state space where the atomic proposition $a \in AP$ is satisfied, i.e. $[\![a]\!] = h^{-1}\left([\![a]\!]_{\mathcal{Y}}\right)$.

[4]This notion is equivalent to the concept of alternation depth in μ-calculus.

1.3.1. Invariance Problems

Arguably the most fundamental type of specification—as well as one of the most important ones due to the intimate connection with system safety—is invariance. The LTL formula of an invariance specification $\Box a$, *always* a, mandates that the system state should remain in the set $[\![a]\!]$ at all times. It is well known (see e.g. [19]) that the winning set of an invariance specification is

$$\mathrm{Win}^{\mathcal{T}}(\Box a) = \nu V_0 \ \left([\![a]\!] \cap \mathrm{Pre}^{\mathcal{T}}(V_0) \right). \tag{1.13}$$

For the continuous-time analogue, let g_a be a function that characterizes the set $[\![a]\!]$, i.e. $[\![a]\!] = \{x : g_a(x) \geq 0\}$. The goal is to find a function $h(t,x)$ satisfying a property analogous to (1.13). In the context of interpreting sets as the zero superlevel set of functions, set intersection corresponds to the minimum operation: $x \in [\![a]\!] \cap [\![b]\!]$ if and only if $\min(g_a, g_b)(x) \geq 0$. By applying the same steps as in (1.9)-(1.10) the following continuous analogue of (1.13) is obtained:

$$\begin{aligned} h(0,x) &= g_a(x), \\ \frac{\partial h}{\partial t} &= \min\left(g_a - h, F\left(x, \frac{\partial h}{\partial x}\right)\right), \quad t > 0, x \in \mathcal{X}. \end{aligned} \tag{1.14}$$

As rigorously shown in [79], $\mathrm{Win}^{\Sigma_{CT}}(\Box a) = \{x : \lim_{t \longrightarrow \infty} h(t,x) \geq 0\}$ where $h(t,x)$ is the viscosity solution [29] of (1.14)[5]. The first term in the minimization ensures that $h(t,\cdot) \leq g_a$ for all $t \geq 0$; hence the set described by $h(t,\cdot)$ is a subset of $[\![a]\!]$. A controller that enforces invariance can also be extracted by considering the gradient of the resulting level set function.

Bibliographical remarks. Computation of $\mathrm{Win}(\Box a)$ via (1.13) is a well understood problem for linear discrete-time systems when $[\![a]\!]$ is polyhedral [19]—the Pre operator can be implemented as a polyhedral projection as described in Section 1.2.2 and preserves convexity. Stability conditions that guarantee that the algorithm completes in a finite number of iterations have also been discovered [44]; if such conditions are not fulfilled it is possible to approximate $\mathrm{Win}(\Box a)$ to arbitrary precision both from the inside and outside [34, 112]. A

[5] In [79] the expression $g_a - h$ is substituted for 0, but the two formulas yield equivalent results. The finite analogue is that (1.13) can equivalently be written as the recursion $V_0 = [\![a]\!]$, $V_{k+1} = V_k \cap \mathrm{Pre}^{\mathcal{T}}(V_k)$.

conservative but flexible LMI-based procedure that does not require iterations was studied in [125]. Explicit computation or approximation of $\mathrm{Win}(\Box a)$ requires manipulation of polyhedra in the combined dimension of state space and input space; in particular, polyhedral projection is required which is a computationally expensive operation. Under the restriction that the input is constant—which induces some conservatism—the projection step can be avoided which results in an implicit inner approximation of $\mathrm{Win}(\Box a)$. Such inner approximation are known as *reference governors* and have been subject to significant research efforts [42].

Beyond linear systems, the invariance problem has been considered for polynomial systems via optimization over positive polynomials [66]. More specifically, these methods rely on a reformulation of the problem as an infinite-dimensional moment problem [70]. The dual of such a formulation turns out to be an optimization problem with positivity constraints that can be formulated as Sum of Squares (SOS) constraints[6], thus resulting in an optimization problem over the set of SOS polynomials which in turn is a convex Semi-Definite Program (SDP)—but still infinite-dimensional. By truncating the maximal polynomial degree a finite-dimensional SDP is obtained, from which an outer approximation of $\mathrm{Win}(\Box a)$ can be extracted. As the truncation degree increases the result converges to $\mathrm{Win}(\Box a)$ from the outside. A great benefit of the method is that a convex optimization problem is obtained even when the involved sets are non-convex. In addition, no time or space discretization is required for the procedure. However, outer approximations are less desirable from a synthesis perspective as it is not possible to enforce the invariance specification $\Box a$ from everywhere in the obtained set, and it is difficult to quantify the approximation error. Inner approximations would be more useful but are harder to obtain. A two-stage method generating inner approximations has been proposed, but it comes without any convergence guarantees [65]. As an alternative to SOS, polynomial positivity constraints can instead be formulated as Diagonally dominant Sum of Squares (DSOS) or Scaled Diagonally dominant Sum of Squares (SDSOS) constraints, that result in linear or conic optimization problems, respectively [2]. Such problems are more efficient to solve than an SDP, so DSOS and SDSOS may enable more complex problems to be solved at the cost of some conservativeness: the set of DSOS polynomials is a strict subset of the set of

[6] A SOS constraint is in general more restrictive than a polynomial positivity constraint, but in certain cases they are equivalent. Such results are known under the collective name Positivstellensatz; the most applicable in this case is the Positivstellensatz due to Putinar concerning polynomials that are positive on compact sets [104].

SDSOS polynomials which in turn is a strict subset of SOS polynomials, and convergence is not guaranteed as the degree grows [61]. A different polynomial-algebraic approach for discrete-time systems was considered in [11] using a polynomial lifting technique.

For arbitrary nonlinear systems the problem has been addressed in the context of Hamilton Jacobi Bellman (HJB) theory [83, 126] by approximating the level set of the solution of (1.14) using level-set methods. Level-set methods do not scale favorably and can not guarantee an inner approximation, but can in theory approximate the winning set to arbitrary precision.

A less automated approach are *control barrier functions* [9]. Similar to Lyapunov analysis, this method largely relies on manual design of functions that satisfy a differential inequality equivalent to the steady-state version of (1.14). Once a control barrier function is obtained, an elegant quadratic program-based controller can be used to enforce the specification, but due to the manual design step this method presupposes good physical intuition of the system.

1.3.2. Reach-Avoid Problems

The basic reach-avoid problem is to find the set from where the set $[\![b]\!]$ can eventually be reached, while remaining in the set $[\![a]\!]$. Also this type of specification has a winning set that can be characterized as a level 0 fixed point:

$$\mathrm{Win}^{\mathcal{T}}(a \,\mathbf{U}\, b) = \mu V_0 \ \left([\![b]\!] \cup ([\![a]\!] \cap \mathrm{Pre}(V_0))\right). \tag{1.15}$$

Again there is an analogous continuous-time PDE obtained by replacing set union with max:

$$\begin{aligned} h(0,x) &= g_b(x), \\ \frac{\partial h}{\partial t} &= \max\left(g_b - h, \min\left(g_a - h, F\left(x, \frac{\partial h}{\partial x}\right)\right)\right), \quad t > 0, x \in \mathcal{X}. \end{aligned} \tag{1.16}$$

Bibliographical remarks. Computation of $\mathrm{Win}(a \,\mathbf{U}\, b)$ presents difficulties already for discrete-time linear systems due to the union operation in (1.15): the union of two polyhedra is in general not a polyhedron. To avoid having to keep track of sets represented by unions of polyhedra, an inner approximation can be obtained by replacing $[\![b]\!] \cup ([\![a]\!] \cap \mathrm{Pre}(V_0))$ by $[\![a]\!] \cap \mathrm{Pre}(V_0)$, which yields the same result in the case without

disturbance.

In the literature the set $\mathrm{Win}\left(\Diamond b\right)$ is commonly referred to as the *domain of attraction* (DOA) of $[\![b]\!]$. Also this synthesis problem has been addressed for polynomial systems by utilizing sums of squares methods [54, 84, 127], resulting in outer approximations of the winning sets, and for nonlinear systems through numerical level-set solutions of (1.16) [41].

1.3.3. Higher-Level Fixed Points for Complex Specifications

Winning sets of more complex specifications typically require higher-level fixed points; here a few examples are provided. In [94] level 2 fixed points were proposed for specifications of type $\varphi = \Diamond\Box b \wedge \left(\bigwedge_{i\in I} \Box\Diamond c^i\right)$ and the dual $\psi = \Diamond a \vee \left(\bigvee_{i\in I} \Diamond\Box b^i\right) \vee \Box\Diamond c$. The former type which was first proposed in [129] encompasses specifications of persistence and recurrence types that are useful in practice, and it is easy to add an invariance part $\wedge\Box a$ by restricting the state space to $\mathrm{Win}^{\mathcal{T}}(\Box a)$.

$$\mathrm{Win}^{\mathcal{T}}\left(\varphi\right) = \mu V_2\ \nu V_1\ \bigcap_{i\in I} \mu V_0\ \ \mathrm{Pre}^{\mathcal{T}}(V_2) \cup \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}}(V_1)\right) \cup \left([\![b]\!] \cap \mathrm{Pre}^{\mathcal{T}}(V_0)\right),$$

$$\mathrm{Win}^{\mathcal{T}}\left(\psi\right) = \nu V_2\ \mu V_1\ \bigcup_{J\subset I} \nu \begin{bmatrix} \vdots \\ V_0^J \\ \vdots \end{bmatrix} \begin{bmatrix} \vdots \\ \kappa^J(V_2, V_1, \{V_0^K\}) \\ \vdots \end{bmatrix},$$

$$\kappa^J\left(V_2, V_1, \{V_0^K\}\right)) = \begin{pmatrix} [\![a]\!] \cup \left([\![c]\!] \cap \mathrm{Pre}^{\mathcal{T}}(V_2)\right) \cup \mathrm{Pre}^{\mathcal{T}}(V_1) \\ \cup \left(\left(\bigcap_{i\in J} [\![b^i]\!]\right) \cap \mathrm{Pre}^{\mathcal{T}}\left(\bigcup_{L\subset 2^J} V_0^L\right)\right) \end{pmatrix}.$$

Because of the nested character of these algorithms it is important to optimize them to the largest possible extent; fixed-point computations can sometimes be "warm started" with the result of a previous computation [97] which allows a trade-off between memory and time.

Unfortunately, there is no straightforward way of transferring these higher-level fixed-point algorithms into succinct continuous-time analogues corresponding to the PDEs above. The issue is the nestedness: infinitesimal calculus would require a concept of fixed-point derivative with respect to parameters of the corresponding mapping. This presents a challenge for HJB methods when it comes to dealing with complex specifications.

Another practical class of specification that allows computation directly on the system

state space $\mathbb{X}$ is *mode-target* specifications. In essence, the specification is a conjunction of mode-target pairs (m^i, t^i) and specifies that if the system settles in $[\![m^i]\!]$, it must eventually settle in $[\![t^i]\!]$. A mode-target specification can be converted to a GR(1)-formula, but can be computed more efficiently when modes are mutually exclusive with the more parsimonious level 2 fixed point [13]

$$\mathrm{Win}^{\mathcal{T}}\left(\bigwedge_{i\in I} \Diamond\Box m^i \implies \Diamond\Box t^i\right) = \nu V_2 \bigcap_{i\in I} \mu V_1\, \nu V_0 \begin{pmatrix} \left(\mathrm{Pre}^{\mathcal{T}}(V_0) \cap [\![m^i]\!] \cap [\![t^i]\!]\right) \\ \cup \left([\![\neg m^i]\!] \cap \mathrm{Pre}^{\mathcal{T}}(V_2)\right) \\ \cup \mathrm{Pre}^{\mathcal{T}}(V_1) \end{pmatrix}.$$

The fixed points for general GR(1) specifications can be found in [21].

Bibliographical remarks. Research on synthesis for continuous-state systems for larger subsets of LTL has also been conducted. Since the synthesis problem for continuous-state systems suffers both from negative decidability results as well as the complexity stemming from LTL synthesis and reachability computations, such approaches invariably scale badly with system dimension.

A significant body of work is devoted to abstraction procedures that map a continuous-state synthesis problem into a finite one. A finite abstraction of a continuous system is a FTS where each discrete state corresponds to a set of states in the concrete continuous system. As a consequence of the finite representation, abstractions typically introduce conservativeness in synthesis; that is, the maximal winning set for the concrete system is not recovered. Abstractions can for example be homogeneous grids [102, 123], or polyhedral partitions for discrete-time [40, 64, 124] or continuous-time [73] systems. Tools such as Pessoa [81], SCOTS [113] and TuLiP [40] automate both abstraction and subsequent finite synthesis.

Correct-by-construction control synthesis via abstractions has also been considered for different logics than LTL. Computational Tree Logic (CTL) has similar grammar to LTL, but differs in semantics: while LTL is interpreted over individual output trajectories (single future), CTL is interpreted over sets of output trajectories (multiple futures). There are behaviors that can be expressed as a CTL specification but not as an LTL specification, and vice versa. An example of a specification that is in CTL but not in LTL is "there should always exist a path to a ξ_0" (although that state is not necessarily reached). Supervisory control of Discrete-Event Systems (DESs) against a CTL specification is reminiscent

of reactive LTL synthesis; algorithms are based on fixed points and specific problem instances can be shown to be equivalent [36]. Abstraction methods that construct a DES for subsequent analysis via supervisory control include [31] and [107].

In addition to abstraction procedures, also native continuous-domain methods have been explored. Instead of treating infinite-horizon LTL properties, the scope can be limited to *metric* temporal logic (MTL)—a bounded version of LTL where temporal operators are augmented with an interval I limiting the scope of satisfaction [67]. That is, $(\zeta, k) \models \Box_I a$ if and only if $\zeta_k = \texttt{True}$ for all $k \in I$, and similarly for other temporal operators. The same specifications can be expressed less succinctly in standard LTL as a conjunction or disjunction of stacked "next" operations, implying that MTL is a strict subset of LTL.

By introducing binary auxiliary variables, MTL specifications can be reformulated as integer programming constraints, which combined with constraints representing the dynamics can be used within an MPC-like framework to generate trajectories starting from a given initial state. This method was for instance studied in [106, 130], providing a complete approach for systems whose dynamics can be stated as mixed integer linear constraints. Since such methods do not explicitly characterize the set of initial states from where the specification can be enforced, an optimization problem must be solved on-line when the initial state is known.

The reason that integer constraints are needed is to treat "or" conditions in the specification that induce non-convexity. A different approach was explored in [114], where integer constraints were substituted for unions of polyhedra. The number of unions grows exponentially with the length of the formula which again illustrates the complexity of the general synthesis problem.

1.4. Simulation Relations and System Abstractions

As discussed above, synthesis for continuous-time systems is difficult. A natural approach is therefore to convert the concrete continuous synthesis problem into a finite one via abstraction. For the procedure to be meaningful it must be possible to transfer a controller designed for the abstraction to the concrete system in a way that preserves specification satisfaction.

There are several notions of *simulation relations* that formalize such relationship between the concrete system and its abstraction.

Definition 1.10. *Let $\mathcal{T}_1 = (\mathbb{X}_1, \mathbb{U}, \longrightarrow_1, h_1)$ and $\mathcal{T}_2 = (\mathbb{X}_2, \mathbb{U}, \longrightarrow_2, h_2)$ be two (finite or infinite) transition systems with identical input and output spaces. $\mathcal{T}_2$ is said to be an* ***over-approximation*** *of $\mathcal{T}_1$, written $\mathcal{T}_1 \preceq \mathcal{T}_2$, if there exists an* ***abstraction function*** *$\alpha : \mathbb{X}_1 \to \mathbb{X}_2$ such that for all $\xi_1 \in \mathbb{X}_1$,*

- $h_1(\xi_1) = h_2(\alpha(\xi_1))$.
- *For all $\mu \in \mathbb{U}$, for all ξ_1' such that $\xi_1 \xrightarrow{\mu} \xi_1'$, there exists a transition $\alpha(\xi_1) \xrightarrow{\mu} \alpha(\xi_1')$.*

This one-way relation is enough to guarantee that a controller $\pi_2 = (\pi_{\mathbb{X}_2}, \pi_{\mathbb{M}_2}, \mathbb{M}_2, m_{0,2})$ designed for $\mathcal{T}_2$ can be implemented on $\mathcal{T}_1$ with preserved correctness: if all π_2-controlled trajectories of $\mathcal{T}_2$ satisfy a specification, then all π_1-controlled trajectories of $\mathcal{T}_1$ satisfies the same specification, where $\pi_1 = (\pi_{\mathbb{X}_1}, \pi_{\mathbb{M}_1}, \mathbb{M}_1, m_{0,1})$ for

$$\pi_{\mathbb{X}_1}(\xi_1, m) = \pi_{\mathbb{X}_2}(\alpha(\xi_1), m),\ \pi_{\mathbb{M}_1}(\xi_1, m) = \pi_{\mathbb{M}_2}(\alpha(\xi_1), m),\ \mathbb{M}_1 = \mathbb{M}_2,\ m_{0,1} = m_{0,2}. \quad (1.17)$$

Theorem 1.1. *Suppose that π_2 enforces φ on $\mathcal{T}_2$ for initial conditions in X, then π_1 enforces φ on $\mathcal{T}_1$ for initial conditions in $\alpha^{-1}(X)$.*

Proof. All π_2-controlled trajectories ζ_2 of $\mathcal{T}_2$ satisfy φ, i.e.

$$h_2(\zeta_2(0))h_2(\zeta_2(1))h_2(\zeta_2(2)) \ldots \models \varphi.$$

It is therefore enough to show that all output trajectories of $\mathcal{T}_1$ generated by π_1 are possible output trajectories of $\mathcal{T}_2$ generated by π_2. Take a π_1-controlled trajectory $\zeta_1(0)\zeta_1(1)\zeta_1(2)\ldots$ of $\mathcal{T}_1$ such that $\zeta_1(0) \in \alpha^{-1}(X)$. corresponding π_2-controlled trajectory ζ_2 of $\mathcal{T}_2$ can be iteratively constructed in a way such that

$$\zeta_2(k) = \alpha(\zeta_1(k)) \quad \forall k \in \mathbb{Z}_+. \quad (1.18)$$

Output equivalence $h_1(\zeta_1(k)) = h_2(\zeta_2(k))$ then immediately follows from the first condition in Definition 1.10.

The validity of (1.18) is shown by induction. By selecting $\zeta_2(0) = \alpha(\zeta_1(0))$, $\zeta_2(0) \in X$ is obtained. Suppose that (1.18) holds at time k and that the internal controller states are equal: $m_1(k) = m_2(k)$. By (1.17) then $\pi_{\mathbb{X}_1}(\zeta_1(k), m_1(k)) = \pi_{\mathbb{X}_2}(\zeta_2(k), m_2(k))$, i.e. the same control action(s) are selected by both controllers. It now follows by the second condition in Definition 1.10 that there exists a π_2-controlled successor $\zeta_2(k+1)$ of $\zeta_2(k)$ such that

$\alpha(\zeta_1(k+1)) = \zeta_2(k+1)$. Furthermore the internal controller states update identically by (1.17) which completes the proof. □

A one-way simulation relation is however insufficient to make conclusions about *necessity*; even if no controller exists for $\mathcal{T}_2$ the synthesis problem for $\mathcal{T}_1$ could potentially have a solution. To infer such negative results from the abstraction a two-way relationship is required. An obvious generalization of (1.10) is that of a *bisimulation* [12], but a bisimulation imposes a sort of equivalence between the two systems that is usually too restrictive. As a remedy the notion of *approximate* bisimulation relations has been proposed [100, 102].

Definition 1.11. *Let $\mathcal{T}_1 = (\mathbb{X}_1, \mathbb{U}, \longrightarrow_1, h_1)$ and $\mathcal{T}_2 = (\mathbb{X}_2, \mathbb{U}, \longrightarrow_2, h_2)$ be two transition systems with identical input and output spaces. They are said to be ϵ-**approximately bisimilar** in the norm $\|\cdot\|$, written $\mathcal{T}_1 \cong_\epsilon \mathcal{T}_2$, if there exists a relation $\sim$ between $\mathbb{X}_1$ and $\mathbb{X}_2$ such that for all $\xi_1 \sim \xi_2$,*

- $\|h_1(\xi_1) - h_2(\xi_2)\| \leq \epsilon$.
- *For all ξ_1' such that $\xi_1 \xrightarrow{\mu}_1 \xi_1'$, there exists a transition $\xi_2 \xrightarrow{\mu}_2 \xi_2'$ with $\xi_1' \sim \xi_2'$.*
- *For all ξ_2' such that $\xi_2 \xrightarrow{\mu}_2 \xi_2'$ there exists a transition $\xi_1 \xrightarrow{\mu}_1 \xi_1'$ with $\xi_1' \sim \xi_2'$.*

Compared to Definition 1.10 the output equivalence requirement has been weakened, but a converse requirement has been added. This implies that a controller implemented according to (1.17) generates trajectories that are ϵ-close in the output space; thus, if a specification is enforced for $\mathcal{T}_2$ it is approximately enforced also on $\mathcal{T}_1$. Conversely, if the specification can not be enforced on $\mathcal{T}_2$ it can be shown that any trajectory of $\mathcal{T}_1$ must be ϵ-close to a violation of the specification. Therefore the notion of ϵ-approximate bisimulation also implies an approximate equivalence of enforceable specifications.

Bibliographical remarks. The concept of over-approximation in Definition 1.10 was first introduced in [76] and is closely related to the recently proposed idea of *feedback refinement relations* [108].

(Exact) bisimulation is a well-known concept from computer science, and there is an algorithm for computing the coarsest bisimilar system [12]. The notion of an approximate bisimulation was proposed in [100] and it is known that an approximately bisimilar deterministic abstraction can always be constructed for incrementally stable systems [46]. An extension called *alternating* approximate bisimilarity, which formalizes control selection

under nondeterminism, was later proposed [102]; enabling non-deterministic abstractions of systems that are not incrementally stable [133].

Chapter 2.

Non-Uniform Abstractions

In this chapter the control synthesis problem for switched systems is considered. In particular, an abstraction-based procedure capable of dealing with nonlinear dynamics is proposed, together with a specification-guided refinement algorithm that refines the partition only in relevant areas. The refinement procedure results in a non-uniform abstraction which can be orders of magnitude smaller than a comparable uniform abstraction.

The starting point is a Finite Transition System (FTS)-based abstraction that satisfies the condition in Definition 1.10, meaning that it captures all transitions of the concrete continuous system. This is achieved by partitioning the state space into cells, and analyzing cell boundaries in order to determine whether transitions between cells are possible. This type of abstraction procedure—previously studied in [73, 121]—is sound in that it captures all possible behaviors of the concrete system, but often results in significant nondeterminism that may cause specifications to become unrealizable on the abstraction. To counteract this negative effect, the abstraction is augmented with transience properties of the concrete system that eliminate spurious behaviors. This results in a new type of transition system called Augmented Finite Transition System (AFTS).

Chapter overview. The following Section 2.1 introduces general theory of AFTSs and is followed by Section 2.2 that contains a detailed explanation of AFTS-based abstraction of switched systems. Next, a specification-guided abstraction-synthesis-refinement loop is proposed in Section 2.3, followed in Section 2.4 by an adaptation of the fixed point-based winning set characterizations from Section 1.3 to AFTSs. Numerical examples are shown in Section 2.5 and the chapter is concluded in Section 2.6.

Related work. Ideas similar to AFTSs have previously been considered for verification [14, 62] and counter example-guided abstraction refinement has been proposed in the model checking community [55]; the refinement based on candidate losing sets in this chapter can be seen as a generalization of the refinement strategy in [55]. In the context of abstraction-based control synthesis, eliminating potentially spurious self-loops—which are special instances of spurious cycles—has been proposed as a post-processing step during synthesis [28, 132]. AFTSs and associated synthesis algorithms provide a systematic methodology to describe all potential spurious cycles in a unified way, thus eliminating the need for post-processing. AFTS-based abstractions are also related to the notion of fair simulation relations studied for purely discrete systems [12]. Incremental synthesis for partition refinement has been proposed for stochastic linear systems [122] and for deterministic abstractions [80]. Compared to this work, both of these papers consider a different fragment of LTL, namely $GR(1)$, and the class of continuous dynamics is limited. Work in the same direction has also appeared previously in [95] and [121].

2.1. Augmented Finite Transition Systems

Below AFTSs are introduced, along with relations between AFTSs and switched systems.

Definition 2.1. *An **Augmented Finite Transition System (AFTS)** is a pair $\mathcal{T}^+ = (\mathcal{T}, \mathcal{G})$, where $\mathcal{T} = (\mathbb{X}, \mathbb{U}, \longrightarrow, h)$ is a FTS and $\mathcal{G} : 2^{\mathbb{U}} \longrightarrow 2^{2^{\mathbb{X}}}$ is a **progress group map**.*

As can be seen, the only difference between an AFTS and a standard FTS (c.f. Definition 1.3) is the progress group map, which maps a subset of actions $U \in 2^{\mathbb{U}}$ to a set of subsets of states. A set $G \in \mathcal{G}(U)$ is called a *progress group under the action set U* and restricts the behavior of $\mathcal{T}^+$ in the following way: the system cannot remain indefinitely within G by exclusively choosing actions from U. In previous work [90, 121], only single-action progress group maps of the form $\mathcal{G} : \mathbb{U} \longrightarrow 2^{2^{\mathbb{X}}}$ were utilized. Figure 2.1 shows an example of how multi-action progress groups can encode synthesis-critical information that is not possible to encode with single-action progress groups.

Since progress groups restrict the behavior of a FTS in a way that may be incompatible with the usual semantics, an additional assumption needs to be made regarding the structure of the AFTS.

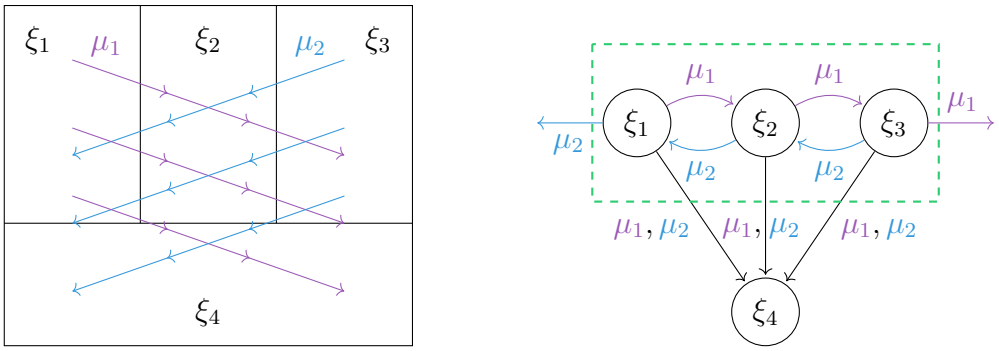

Figure 2.1: A multi-action progress group can encode information that restricts the behavior of an AFTS in a way that is useful for control purposes. The nondeterministic FTS to the right can be thought of as an abstraction (see Definition 2.5) of the switched system with two modes to the left. Without progress groups and starting in ξ_1, there is no way to choose between actions μ_1 and μ_2 to guarantee that the system ends up in ξ_4 since $\xi_1\xi_2\xi_3\xi_2\xi_1\xi_2\ldots$ is a valid trajectory. However, if $G = \{\xi_1, \xi_2, \xi_3\}$ is a progress group under actions $\{\mu_1, \mu_2\}$, i.e. $G \in \mathcal{G}(\{\mu_1, \mu_2\})$, the state can not remain indefinitely in G while these actions are used. This is information that can be inferred from the switched system. Therefore, by always selecting μ_1 in ξ_1 and μ_2 in ξ_3, the state can be controlled to ξ_4 in finite time. This type of progress group information is exploited in the synthesis algorithms presented later in the chapter.

Definition 2.2. *An AFTS $\mathcal{T}^+ = (\mathcal{T}, \mathcal{G})$ is said to be* ***well-formed*** *if the following holds for all $U \in 2^{\mathbb{U}}$ and for all subsets $V \in 2^G$ of all $G \in \mathcal{G}(U)$: there is a state $\xi_1 \in V$ such that for all actions $\mu \in U$ there exists a transition $(\xi_1, \mu, \xi_2) \in \longrightarrow$ such that $\xi_2 \notin V$.*

This condition is required to comply with the notion of a progress group; if an AFTS is not well-formed it is possible to select actions so that the state remains indefinitely in some $V \subset G$. Another way to interpret well-formedness is that a progress group $G \in \mathcal{G}(U)$ can not contain a controlled invariant set for actions restricted to U.

Below it is without loss of generality assumed that all actions are enabled at every state[7]. That is, for all $\xi \in \mathbb{X}$ and for all $\mu \in \mathbb{U}$, there exists at least one $\xi' \in \mathbb{X}$ such that $(\xi, \mu, \xi') \in \longrightarrow$. A **trajectory** of an AFTS $\mathcal{T}^+ = (\mathcal{T}, \mathcal{G})$ is a trajectory $\zeta = \zeta(0)\zeta(1)\zeta(2)\ldots$ of $\mathcal{T}$ generated from inputs $\sigma = \sigma(0)\sigma(1)\sigma(2)\ldots$ with the additional requirement that it satisfies the progress conditions encoded by $\mathcal{G}$. That is,

$$\forall U \in 2^{\mathbb{U}},\ \forall G \in \mathcal{G}(U),\ \forall T \geq 0,\ \exists k > T \text{ s.t. } (\zeta(k), \sigma(k)) \notin G \times U. \tag{2.1}$$

[7]A dummy state ξ_d can be added, together with transitions from all forbidden state-action pairs, to obtain an AFTS that is equivalent for synthesis purposes.

The following result implies that the synthesis problem for AFTSs against LTL specifications can be reduced to the standard synthesis problem for FTSs.

Proposition 2.1. *Given a well-formed AFTS $\mathcal{T}^+ = (\mathcal{T}, \mathcal{G})$ and an LTL formula φ, let*

$$\varphi_{G,U} = \neg\Diamond\Box(\xi \in G \wedge \mu \in U) = \Box\Diamond\left((\xi \notin G) \vee (\mu \notin U)\right), \tag{2.2}$$

for a progress group G under an action set U. Then, there exists a controller for $\mathcal{T}^+$ that guarantees the satisfaction of φ, if and only if there exists a controller for $\mathcal{T}$ that guarantees the satisfaction of the following modified specification φ' over an extended set of atomic propositions, defined as

$$\varphi' = \left(\bigwedge_{U \in 2^{\mathbb{U}}} \bigwedge_{G \in \mathcal{G}(U)} \varphi_{G,U}\right) \longrightarrow \varphi.$$

Proof. The progress group condition (2.1) is captured by the LTL formula (2.2). Therefore, the progress information can equivalently be stated as part of the formula φ'. □

Remark that if $\varphi \in GR(1)$, then also $\varphi' \in GR(1)$, so the algorithmic complexity of the synthesis problem is not affected by the addition of progress group encodings. Next, the notion of abstraction in Definition 1.10 is generalized to AFTSs.

Definition 2.3. *Given two AFTSs $\mathcal{T}_1^+ = (\mathcal{T}_1, \mathcal{G}_1)$ and $\mathcal{T}_2^+ = (\mathcal{T}_2, \mathcal{G}_2)$ with the same input sets, $\mathcal{T}_2^+$ is said to be an **over-approximation** of $\mathcal{T}_1^+$, written $\mathcal{T}_1^+ \preceq \mathcal{T}_2^+$, if*

- *$\mathcal{T}_1 \preceq \mathcal{T}_2$ in the sense of Definition 1.10 with abstraction function $\alpha : \mathbb{X}_1 \to \mathbb{X}_2$.*
- *For all $U \in \mathbb{U}$ and all $G_2 \in \mathcal{G}_2(U)$, there exists $G_1 \in \mathcal{G}_1(U)$ such that $\alpha^{-1}(G_2) \subset G_1$.*

Next, a similar relation is established between switched systems and AFTSs. This requires the concept of *transience*—the continuous equivalent of progress groups.

Definition 2.4. *Given a switched system $\Sigma_{Sw} = (\mathcal{X}, \mathbb{U}, \mathcal{D}, \{f_\mu\}_{\mu \in \mathbb{U}}, h)$, a set $X \subset \mathcal{X}$ is said to be **transient** on a set of modes $U \subset \mathbb{U}$ if and only if for any state $x_0 \in X$, for any disturbance $\mathbf{d}(t)$ taking values in $\mathcal{D}$, and input sequence $\sigma(t)$ taking values in $\mathbb{U}$, there exists a bounded interval I such that the trajectory $\mathbf{x} : I \longrightarrow X$ of $D^+\mathbf{x}(t) = f_{\sigma(t)}(\mathbf{x}(t), \mathbf{d}(t))$ with $\mathbf{x}(0) = x_0$ leaves X, i.e., $\mathbf{x}(I) \setminus X \neq \emptyset$.*

As with progress groups, there is an equivalent interpretation of transience in terms of controlled invariance: a set X is transient on a set of modes $U \subset \mathbb{U}$ if and only if it does not contain a controlled invariant set when the mode signal is restricted to U and d is regarded as a control input.

Definition 2.5. *An AFTS $\mathcal{T}^+ = (\mathbb{X}, \mathbb{U}, \longrightarrow, h_{\mathbb{X}}, \mathcal{G})$, is an **abstraction** of the switched system $\Sigma_{Sw} = (\mathcal{X}, \mathbb{U}, \mathcal{D}, \{f_\mu\}_{\mu\in\mathbb{U}}, h_{\mathcal{X}})$, denoted $\Sigma_{Sw} \preceq \mathcal{T}^+$, if there exists an **abstraction function** $\alpha : \mathcal{X} \longrightarrow \mathbb{X}$, such that the following statements hold.*

1. *For all $x \in \mathcal{X}$, $h_{\mathcal{X}}(x) = h_{\mathbb{X}}(\alpha(x))$.*
2. *Given states $\xi, \xi' \in \mathbb{X}$, if there exist $x_0 \in \alpha^{-1}(\xi)$, a compact time interval I (not a singleton), and a disturbance $\mathbf{d} : I \longrightarrow \mathcal{D}$ such that the corresponding trajectory $\mathbf{x} : I \longrightarrow X$ of $D^+\mathbf{x}(t) = f_\sigma(\mathbf{x}(t), \mathbf{d}(t))$, with $\mathbf{x}(0) = x_0$, satisfies*

$$\mathbf{x}\left(\max_{t\in I} t\right) \in \alpha^{-1}(\xi'), \quad \mathbf{x}(I) \subset \alpha^{-1}(\xi) \cup \alpha^{-1}(\xi'),$$

   then $(\xi, \mu, \xi') \in \longrightarrow$.
3. *The progress group map $\mathcal{G}$ is such that for all action sets $U \in 2^{\mathbb{U}}$, for all $G \in \mathcal{G}(U)$, the set $\alpha^{-1}(G)$ is transient on the mode set U of Σ_{Sw}.*

As shown next, the over-approximation as defined above preserves LTL realizability, which implies that a correct controller for an over-approximation $\mathcal{T} \succeq \Sigma_{Sw}$ can be implemented as a switching protocol on Σ_{Sw}.

Theorem 2.1. *Let Σ_{Sw} be a switched system that is over-approximated by an AFTS $\mathcal{T}^+$, i.e., $\mathcal{T}^+ \succeq \Sigma_{Sw}$. Furthermore, let φ be an LTL ($LTL_{\setminus\bigcirc}$ for continuous time, c.f. Definition 1.2) formula over the atomic propositions of Σ_{Sw} and $\mathcal{T}^+$. If there exists a controller for $\mathcal{T}^+$ that guarantees the satisfaction of φ, then[8] there exists a switching protocol for Σ_{Sw} that guarantees the satisfaction of φ.*

Proof. The proof shows that for any single-valued[9] controller $\pi_{\mathcal{T}^+} = (\pi_{\mathbb{X}}, \pi_{\mathbb{M}}, \mathbb{M}, m_0)$ for $\mathcal{T}^+$, there is a corresponding switching protocol $\pi_{\mathcal{X}}$ for Σ_{Sw} such that each trajectory of Σ_{Sw} resulting from $\pi_{\mathcal{X}}$ corresponds to a $\pi_{\mathcal{T}^+}$-controlled trajectory of $\mathcal{T}^+$.

[8] In the continuous time case, some mild conditions on continuous implementations of the strategy being non-Zeno are necessary. See Section 2.4.2 for more details.

[9] Taking values in $\mathbb{U}$, as opposed to in $2^{\mathbb{U}}$. Any set-valued controller can be restricted to a single-valued strategy by fixing a selection rule.

The switching protocol $\pi_{\mathcal{X}} : \mathcal{X} \times \mathbb{M} \to \mathbb{U}$ has an internal memory state $m_c(t)$. For continuous time, it is constructed inductively for a trajectory $\mathbf{x}$ with $\mathbf{x}(0) = x_0$. First, given an internal memory state $m_c(t)$ at time t, the switching protocol outputs $\pi_{\mathcal{X}}(\mathbf{x}(t), m_c(t)) = \pi_{\mathbb{X}}(\alpha(\mathbf{x}(t)), m_c(t))$. Secondly, let $\zeta(0) = \alpha(x_0)$, $m(0) = m_0$, and $t_1 = \inf\{t \geq 0 : \alpha(\mathbf{x}(t)) \neq \zeta(0)\}$. Then the internal memory state of $\pi_{\mathcal{X}}$ on the interval $[0, t_1)$ is $m_c(t) = m(0)$. Lastly, the update rule of the internal memory state is defined. For a time $t \geq t_k$, let the internal memory state of π be $m_c(t) = m(k)$. Then define $m_c(t)$ to be constant on the interval $[t_k, t_{k+1})$, for $t_{k+1} = \inf\{t \geq t_k : \alpha(\mathbf{x}(t)) \neq \alpha(\mathbf{x}(t_k))\}$. Define $\zeta(k+1) = \alpha(\lim_{\tau \searrow t_{k+1}} \mathbf{x}(\tau))$, where $\searrow$ indicates the limit from above, and let $m_c(t) = m(k+1) = \pi_{\mathbb{M}}(\zeta(k+1), m_k)$ for times $t \geq t_{k+1}$.

This construction results in a strictly increasing sequence $\{t_k\}_{k=0}^{\infty}$ in $[0, \infty)$ with $t_0 = 0$, such that the output maps are equivalent,

$$\begin{aligned} \pi_{\mathcal{X}}(\mathbf{x}(t), m_c(t)) &= \pi_{\mathbb{X}}(\zeta(k), m(k)), \\ h_{\mathcal{X}}(\mathbf{x}(t)) &= h_{\mathbb{X}}(\alpha(\mathbf{x}(t))) = h_{\mathbb{X}}(\zeta(k)), \end{aligned} \qquad \forall t \in [t_k, t_{k+1}). \tag{2.3}$$

For discrete time, the construction simplifies to initializing $\zeta(0) = \alpha(x_0)$ and $m_c(0) = \pi_{\mathbb{M}}(\zeta(0), m_0)$. The switching protocol outputs $\pi_{\mathcal{X}}(\mathbf{x}(k), m_c(k)) = \pi_{\mathbb{X}}(\alpha(\mathbf{x}(k)), m_c(k))$ and the internal memory state updates as $m_c(k+1) = \pi_{\mathbb{M}}(\alpha(\mathbf{x}(k+1)), m_c(k))$.

Also this construction leads to output equivalence between $\mathcal{T}^+$ and Σ_{Sw}, given as

$$h_{\mathcal{X}}(\mathbf{x}(k)) = h_{\mathbb{X}}(\alpha(\mathbf{x}(k))) = h_{\mathbb{X}}(\zeta(k)), \quad \forall k \in \mathbb{N}. \tag{2.4}$$

By construction, in both cases $\zeta = \zeta(0)\zeta(1)\zeta(2)\ldots$ is a $\pi_{\mathcal{T}^+}$-controlled trajectory of $\mathcal{T}^+$. If $\zeta \models \varphi$, it follows from (2.3) (resp. (2.4)) that also $\mathbf{x} \models \varphi$. □

2.2. Abstractions of Switched Systems

Below an abstraction of a switched system

$$\Sigma_{Sw} = (\mathcal{X}, \mathbb{U}, \mathcal{D}, \{f_\mu\}_{\mu \in \mathbb{U}}, h_{\mathcal{X}})$$

is constructed in the form of an AFTS. The resulting abstraction satisfies the properties of an over-approximation in Definition 2.5, which enables reasoning about trajectories of Σ_{Sw} via algorithmic analysis of the abstraction.

Consider an abstraction based on a partition of the domain $\mathcal{X}$, it can be defined in terms of the equivalence classes $\mathcal{X}/\sim_\alpha$ of an abstraction function α and its associated equivalence relation $\sim_\alpha$:

$$x \sim_\alpha x' \quad \text{iff} \quad \alpha(x) = \alpha(x').$$

In the following it is assumed that α takes a finite number of values, which also makes the corresponding partition $\mathcal{X}/\sim_\alpha$ finite, and that each partition cell $\alpha^{-1}(\xi) = \{x \in \mathcal{X} : \alpha(x) = \xi\}$ satisfies the regularity property $\mathrm{cl}\,(\mathrm{int}\,(\alpha^{-1}(\xi))) = \mathrm{cl}\,(\alpha^{-1}(\xi)) \neq \emptyset$. From a computational point of view, the type of partitions that are used later (e.g., consisting of hyper boxes or convex polyhedra with non-empty interior) naturally satisfy this assumption. Attention is furthermore restricted to abstraction functions α that are *proposition preserving* with respect to the set of atomic propositions AP. Being proposition preserving means that continuous states that are in the same equivalence class have identical truth evaluations with respect to AP, i.e., for $x, x' \in \mathcal{X}$ and $a \in AP$, $x \sim_\alpha x'$ implies that $h_\mathcal{X}(x) \in [\![a]\!]_\mathcal{Y}$ if and only if $h_\mathcal{X}(x') \in [\![a]\!]_\mathcal{Y}$.

Consider now an AFTS $\mathcal{T}^+ = (\mathbb{X}, \mathbb{U}, \longrightarrow, h_\mathbb{X}, \mathcal{G})$, where $\mathbb{X} = \alpha(\mathcal{X})$, $h_\mathbb{X} = h_\mathcal{X} \circ \alpha^{-1}$, and the constructions of $\longrightarrow$ and $\mathcal{G}$ are described below.

Determine transitions $\longrightarrow$. To make the abstraction an over-approximation, a transition (ξ, μ, ξ') must be added to the AFTS whenever there exists a corresponding trajectory between $\alpha^{-1}(\xi)$ and $\alpha^{-1}(\xi')$ in Σ_{Sw}, but an over-approximation can also contain other (spurious) transitions. To limit spurious behaviors in the abstraction the number of extra transitions should be minimized, this is done by in Algorithm 1 which systematically searches for *blocking certificates* that prove the absence of certain trajectories in Σ_{Sw}. The algorithm typically produces a nondeterministic AFTS, i.e., for a given state-action pair (ξ, μ) there may be several successors ξ' such that $(\xi, \mu, \xi') \in \longrightarrow$. Algorithm 1 depends on the following subroutine:

R.1 `isBlocked`$(C_1, C_2, f, \mathcal{D})$: Given two sets $C_1, C_2 \subset \mathbb{R}^n$, single-mode dynamics described by f, and a disturbance set $\mathcal{D}$, return `False` if there exists a non-singleton compact interval I and a trajectory $\mathbf{x} : I \longrightarrow X$ of $D^+\mathbf{x}(t) = f(\mathbf{x}(t), \mathbf{d}(t))$ that satisfies

$$\mathbf{x}(0) \in C_1, \quad \mathbf{x}\left(\max_{t \in I} t\right) \in C_2, \quad \mathbf{x}(I) \subset C_1 \cup C_2,$$

and `True` otherwise.

Algorithm 1: Compute transitions $\longrightarrow$

```
Function ComputeTrans({f_μ}_{μ∈𝕌}, 𝒟, 𝕏, α)
    Initialize ⟶ = ∅;
    for μ ∈ 𝕌 do
        for (ξ, ξ') ∈ 𝕏 × 𝕏 do
            if not isBlocked(α^{-1}(ξ), α^{-1}(ξ'), f_μ, 𝒟) then
                add (ξ, μ, ξ') to ⟶;
            end
        end
    end
    return ⟶;
```

Remark 2.1. *In the interest of keeping notation simple, treatment of potential out-of-domain trajectories is omitted in Algorithm 1. If there is a trajectory* $\mathbf{x}$ *on an interval* I *of* $D^+\mathbf{x}(t) = f_\mu(\mathbf{x}(t), \mathbf{d}(t))$ *such that*

$$\mathbf{x}(I) \subset \alpha^{-1}(\xi) \cup \mathcal{X}^C, \quad \mathcal{X}^C \cap \mathbf{x}(I) \neq \emptyset,$$

then the action μ *must be avoided at state* ξ *to avoid an out-of-domain transition. This can be encoded in the AFTS by adding a dummy state* ξ_{out} *to* $\mathbb{X}$*, and adding transitions* (ξ, μ, ξ_{out}) *to* $\longrightarrow$ *for all state-action pairs* (ξ, μ) *with potential out-of-domain trajectories, and then avoid* ξ_{out} *in synthesis by amending* $\square\neg\xi_{out}$ *to the specification. If* `isBlocked`$(C, \mathcal{X}^C, f_\mu, \mathcal{D})$ = `True`*, then there are no out-of-domain trajectories from* C *under the flow of* f_μ*.*

Determine progress groups $\mathcal{G}$. Algorithm 2 is used to extract a progress group map from Σ_{Sw}. In essence, it considers collections of discrete states and adds them to the progress group map if their pre-image under α is transient, as determined by the following subroutine:

R.2 `isTransient`$(C_1, \{f_\mu\}_{\mu\in U}, \mathcal{D})$: Given a set C_1, switched dynamics $\{f_\mu\}_{\mu\in U}$, and a disturbance set $\mathcal{D}$, return `False` if there exists a maximal trajectory $\mathbf{x}$ of $D^+\mathbf{x}(t) = f_{\sigma(t)}(\mathbf{x}(t), \mathbf{d}(t))$, with mode switching restricted to $\sigma(\cdot)$ taking values in U, such that $\mathbf{x}(I) \subset C_1$, and `True` otherwise.

Algorithm 1 by default adds many self transitions of the form (ξ, μ, ξ) to $\longrightarrow$. For instance, in continuous time `isBlocked`(C, C, f_μ, D) = `True` by definition. To eliminate spu-

rious self-loops, single-action progress groups can be added for states ξ for which $\alpha^{-1}(\{\xi\})$ is transient. The same progress properties can however also be encoded in larger progress groups, as discussed later in the section.

Algorithm 2: Compute progress group $\mathcal{G}$

```
1  Function ComputeProgGroup({f_μ}_{μ∈𝕌}, 𝒟, 𝕏, α)
2      initialize 𝒢 : 2^𝕌 ⟶ 2^(2^𝕏);
3      for U ∈ 2^𝕌 do
4          initialize 𝒢(U) = ∅;
5          for G ∈ 2^𝕏 do
6              if isTransient(α^{-1}(G), {f_μ}_{μ∈U}, 𝒟) then
7                  add G to 𝒢(U);
8              end
9          end
10     end
11     return 𝒢;
```

Whether the subroutines `isBlocked` and `isTransient` can be implemented, and how efficiently that can be done, depends on the form of the dynamics (1.12) (e.g., linear, polynomial) and the type of sets induced by α (e.g., hyper boxes, polyhedra, semi-algebraic sets). In the end of this section convex optimization formulations that can be used to implement `isBlocked` and `isTransient` for various combinations of dynamics and set descriptions are provided.

Algorithms 1 and 2 can typically be implemented more efficiently than the pseudo-code given here. For instance, at line 4 in Algorithm 1 it is sufficient to consider neighboring pairs of cells in continuous time, as transitions between all other pairs are blocked by definition. The search can also be heavily restricted in discrete time by calculating a global bound on transition distance and automatically infer `isBlocked` to be `True` for pairs of cells separated by a longer distance. Such localized search procedures improve the complexity of the algorithm.

Similarly, the exponential complexity of Algorithm 2 is impractical in most applications but can often be improved. There is no additional benefit of either 1) finding progress groups that are subsets of another progress group under the same action set, or 2) finding a single-action progress group that consists of states that are already part of a multi-action progress group containing the same action. In these cases, the transience properties are already captured by the "larger" progress groups. A reasonable choice is therefore a heuris-

tic search strategy that primarily focuses on finding large multi-action progress groups. In the abstraction-refinement loop described in Section 2.3, the progress group computed for the initial coarse abstraction where $|\mathbb{X}|$ is small is mapped through refinement steps, while search for new progress groups is conducted in localized candidate regions only. It is also worth noting that for systems with globally asymptotically stable equilibrium points per different subsets of inputs, the set of all cells that do not contain the equilibrium forms a large progress group and computation reduces to finding the cells containing equilibrium points.

Intuitively, `isBlocked` and `isTransient` search for certificates of blocked transitions, and transient subsets, so that spurious trajectories in the abstraction can be eliminated. However, it is not necessary that `isBlocked` and `isTransient` are exact in order for the conditions of Definition 2.5 to hold. It is enough that they do not return false positives, as captured in the following theorem. For the same reason, it is not necessary to find all possible progress groups, although more progress groups might increase the quality of the approximation.

Theorem 2.2. *Assume that an abstracting AFTS* $\mathcal{T}^+ = (\mathbb{X}, \mathbb{U}, \longrightarrow, h_{\mathbb{X}}, \mathcal{G})$ *is constructed from a switched system* $\Sigma_{Sw} = (\mathcal{X}, \mathbb{U}, \mathcal{D}, \{f_\mu\}_{\mu\in\mathbb{U}}, h_{\mathcal{X}})$ *as described above, using a proposition preserving abstraction function and implementations of* `isBlocked` *and* `isTransient` *that do not return false positives (i.e., if* `True` *is returned for a given query, the conditions for* `True` *in R.1 resp. R.2 are indeed satisfied). Then* $\mathcal{T}^+$ *is well-formed and* $\mathcal{T}^+ \succeq \Sigma_{Sw}$.

Proof. Condition (i) of Definition 2.5 is satisfied by the definition of $h_{\mathbb{X}}$. If `isBlocked` does not return false positives all transitions present in Σ_{Sw} are added to $\mathcal{T}^+$ by Algorithm 1, which assures the satisfaction of (ii). Similarly, if `isTransient` does not return false positives no progress groups are added by Algorithm 2 unless their pre-image is transient as required by (iii). Thus $\mathcal{T}^+ \succeq \Sigma_{Sw}$.

Next it is shown that $\mathcal{T}^+$ is well-formed. Assume for contradiction that (2.2) is not satisfied. Then there exists $G \in \mathcal{G}(U)$ such that for some $V \subset G$,

$$\forall \xi \in V, \exists \mu \in U \text{ s.t. } (\xi, \mu, \xi') \in \longrightarrow \implies \xi' \in V. \tag{2.5}$$

By the above, $\alpha^{-1}(V)$ is transient on U, meaning that all trajectories of Σ_{Sw} under modes in U eventually exit V. In particular, this holds for trajectories generated by selecting actions according to (2.5). But by the same reasoning as above, the discrete analogues of

these trajectories must be present also in $\mathcal{T}^+$ which is a contradiction of (2.5). □

Now concrete implementations of `isBlocked` and `isTransient` are provided for a variety of situations, starting with general formulations and then specializing to convex, efficient implementations for the special cases of linear and polynomial systems, for both discrete and continuous time. Some variants of `isBlocked` for linear and multi-affine dynamics on rectangular sets or simplices have previously appeared in the literature [16, 45, 52].

2.2.1. General Formulations

For general sets $C_1, C_2 \subset \mathcal{X}$ and dynamics described by $f : \mathcal{X} \times \mathcal{D} \longrightarrow \mathbb{R}^n$, the following optimization formulations can be used to determine whether there is a transition from C_1 to C_2.

Continuous time. Blocking is inferred by a Nagumo-type condition [128, Chapter 10, XVI]. Intuitively, if the vector field along a surface is always pointing "inwards", then no trajectories of that vector field can cross the surface to the "outside". Let f be a continuous function, and $\hat{N}_{C_1}(x)$ be the normal cone[10] of $\mathrm{cl}(C_1)$ at x. Then, if the optimal value of

$$\begin{cases} \max\limits_{x,d} & \hat{n}(x) \cdot f(x,d), \\ \text{s.t.} & x \in \mathrm{cl}(C_1) \cap \mathrm{cl}(C_2), \\ & d \in \mathcal{D}, \\ & \hat{n}(x) \in \hat{N}_{C_1}(x), \end{cases} \tag{BLK-CT}$$

is smaller than or equal to 0, the vector field f points "inwards" towards C_1 everywhere on $\mathrm{cl}(C_1) \cap \mathrm{cl}(C_2)$, which implies that $\texttt{isBlocked}(C_1, C_2, f, \mathcal{D}) = \texttt{True}$.

Discrete time. If the following program is infeasible,

$$\begin{cases} \text{find} & x \in C_1,\ d \in \mathcal{D}, \\ \text{s.t.} & f(x,d) \in C_2, \end{cases} \tag{BLK-DT}$$

[10] The normal cone of a general set can be defined as follows [25]. The tangent cone of a set C at x is the cone $\hat{T}_C(x) = \{u : \forall\{x_i\} \longrightarrow x,\ \forall\{t_i\} \longrightarrow 0 \text{ from above},\ \exists\{u_i\} \longrightarrow u \text{ s.t. } x_i + t_i u_i \in C \text{ for all } i\}$. Informally, $\hat{T}_C(x)$ consists of directions y in which infinitesimal moves from x remain in C. The normal cone of C at x is then the dual of $\hat{T}_C(x)$: $\hat{N}_C(x) = \{v : y^T v \leq 0\ \forall y \in \hat{T}_C(x)\}$.

then $\texttt{isBlocked}(C_1, C_2, f, \mathcal{D}) = \texttt{True}$.

As remarked earlier in the chapter, the concept of transience is equivalent to the non-existence of a controlled invariant set. Therefore, existence of a function satisfying certain Lyapunov-like conditions can be used to infer that a given set is transient. For a decay rate $\epsilon > 0$, the following search problem determines transience of a bounded set C_1 under dynamics governed by $\{f_\mu\}_{\mu \in U}$:

$$\begin{cases} \text{find} & B : \mathrm{cl}(C_1) \longrightarrow \mathbb{R}, \\ \text{s.t.} & \Delta_\mu B(x, d) \leq -\epsilon, \quad \forall x \in \mathrm{cl}(C_1), \quad \forall d \in \mathcal{D}, \quad \forall \mu \in U, \end{cases} \tag{TRS}$$

where

$$\Delta_\mu B(x, d) = \begin{cases} \langle \nabla B(x), f_\mu(x, d) \rangle, & \text{for continuous time,} \\ B(f_\mu(x, d)) - B(x), & \text{for discrete time.} \end{cases}$$

Proposition 2.2. *If a differentiable function B satisfying* (TRS) *can be found, then*

$$\texttt{isTransient}(C_1, \{f_\mu\}_{\mu \in U}, \mathcal{D}) = \texttt{True}.$$

Proof. Assume that a differentiable function B satisfying (TRS) exists. By compactness of $\mathrm{cl}(C_1)$ and continuity of B, B attains a minimal value on $\mathrm{cl}(C_1)$. Assume for contradiction that there is an unbounded interval I and a maximal trajectory $\mathbf{x} : I \longrightarrow \mathrm{cl}(C_1)$ generated by a disturbance $\mathbf{d} : I \longrightarrow \mathcal{D}$ and a switching signal $\sigma : I \longrightarrow U$.
For continuous time,

$$-\epsilon \geq \left\langle \nabla B(\mathbf{x}(t)), f_{\sigma(t)}(\mathbf{x}(t), \mathbf{d}(t)) \right\rangle = \frac{\mathrm{d}}{\mathrm{d}t} B(\mathbf{x}(t)), \quad \text{for almost all } t \in I,$$

Integrating up to time T yields $-\epsilon T \geq B(\mathbf{x}(T)) - B(\mathbf{x}(0))$, which contradicts the boundedness of B on $\mathrm{cl}(C_1)$.
For discrete time,

$$-\epsilon \geq B(f_{\sigma(t)}(\mathbf{x}(t), \mathbf{d}(t)) - B(\mathbf{x}(t)) = B(\mathbf{x}(t+1)) - B(\mathbf{x}(t)).$$

Summing from $t = 0$ to T gives

$$-\epsilon(T+1) \geq \sum_{t=0}^{T} \left(B(\mathbf{x}(t+1)) - B(\mathbf{x}(t)) \right) = B(\mathbf{x}(T+1)) - B(\mathbf{x}(0)),$$

due to a telescopic sum. This again contradicts the boundedness of B on $\mathrm{cl}(C_1)$. $\square$

Existence of such a function B can also be shown to be a necessary condition for transience of compact sets in both continuous [72] and discrete [60] time.

2.2.2. Linear System, Polyhedra

When f is affine and the sets $\mathrm{cl}(C_1)$ and $\mathrm{cl}(C_2)$ are polyhedra, the two subroutines can be implemented as Linear Programs (LPs).

Assume that f is an affine mapping, i.e. $f(x,d) = Ax + Ed + K$, and that the sets $\mathrm{cl}(C_i) = \{x : H_i x \leq h_i\}$ for $i = 1, 2$, and $\mathcal{D} = \{d : H_{\mathcal{D}} d \leq h_{\mathcal{D}}\}$ are (convex) polyhedra. The intersection $\mathrm{cl}(C_1) \cap \mathrm{cl}(C_2)$ is then the common facet of $\mathrm{cl}(C_1)$ and $\mathrm{cl}(C_2)$ which is itself a polyhedron with a normal $\hat{n}$ (assuming it has co-dimension 1).

Then, (BLK-CT) simplifies to the LP

$$\begin{cases} \max\limits_{x,d} & \hat{n}^T(Ax + Ed + K), \\ \text{s.t} & H_1 x \leq h_1,\ H_2 x \leq h_2,\ H_{\mathcal{D}} d \leq h_{\mathcal{D}}. \end{cases} \tag{BLK-CT-LIN}$$

For intersections with co-dimension higher than 1, the normal of any separating hyperplane between C_1 and C_2 can be used to obtain a blocking certificate.

Linear Programs attain their optimal values at vertices. Thus, in order to find the optimal value of (BLK-CT-LIN) it is enough to evaluate the objective function at the combinations of all vertices of the intersection $\mathrm{cl}(C_1) \cap \mathrm{cl}(C_2)$ with all vertices of $\mathcal{D}$. If the vertex enumeration is computationally cheap, as it is for instance for hyper boxes, the enumeration method can be beneficial compared to solving (BLK-CT-LIN) with a standard LP solver.

Similarly, (BLK-DT) becomes the linear feasibility problem

$$\begin{cases} \text{find} & x, d, \\ \text{s.t} & H_1 x \leq h_1,\ H_{\mathcal{D}} d \leq h_{\mathcal{D}}, \\ & H_2(Ax + Ed + K) \leq h_2, \end{cases} \tag{BLK-DT-LIN}$$

which is feasible if and only if $\texttt{isBlocked}(C_1, C_2, f, \mathcal{D}) = \texttt{False}$.

For linear systems, convex sets are controlled invariant if and only if they contain a controlled fixed point (in discrete time this is a special case of Kakutani's fixed point

theorem, see [39] for continuous time), so transience can be determined by proving the absence of such fixed points. Therefore the following LPs are proposed for the *single-mode* case, since a single-mode switched linear system is just a linear system. For the multi-mode case it is preferable to use the polynomial approach described in Section 2.2.3 below. For continuous time,

$$\begin{cases} \text{find} & x,\ d, \\ \text{s.t.} & H_1 x \leq h_1,\ H_{\mathcal{D}} d \leq h_{\mathcal{D}}, \\ & A_\mu x + E_\mu d + K_\mu = 0, \end{cases} \tag{TRS-CT-LIN}$$

and for discrete time,

$$\begin{cases} \text{find} & x,\ d, \\ \text{s.t.} & H_1 x \leq h_1,\ H_{\mathcal{D}} d \leq h_{\mathcal{D}}, \\ & A_\mu x + E_\mu d + K_\mu = x. \end{cases} \tag{TRS-DT-LIN}$$

Here, $\texttt{isTransient}(C_1, \{f_\mu\}, \mathcal{D}) = \texttt{True}$ if and only if the corresponding linear search problem is infeasible.

2.2.3. Polynomial System, Semi-Algebraic Sets

Assume now instead that f is polynomial and that $\mathrm{cl}(C_1), \mathrm{cl}(C_2)$, and $\mathcal{D}$ are simple semi-algebraic sets described by $\mathrm{cl}(C_j) = \{x : g^i_{C_j}(x) \geq 0,\ \forall i \in [k_{C_j}]\}$ with $\deg(g^i_{C_j}) \leq p_{C_j}$ for $j = 1, 2$, and $\mathcal{D} = \{d : g^i_{\mathcal{D}}(d) \geq 0,\ \forall i \in [k_{\mathcal{D}}]\}$ with $\deg(g^i_{\mathcal{D}}) \leq p_{\mathcal{D}}$.

First some notation required to state polynomial optimization problems is introduced. Let $\mathbb{R}_p[x_0, \ldots, x_{n-1}]$ be the set of real polynomials over the variables $\{x_i\}_{i\in[n]}$ and of degree at most p. Similarly, the notation $\Sigma_p[x_0, \ldots, x_{n-1}] \subset \mathbb{R}_p[x_0, \ldots, x_{n-1}]$ is used to denote the set of non-negative polynomials of degree at most p. To enable tractable optimization over the space of non-negative polynomials, the optimization must be further restricted to convex[11] subsets of $\Sigma_p[x_0, \ldots, x_{n-1}]$. For instance, it can be restricted to the set of SOS polynomials $\Sigma^{SOS}_p[x_0, \ldots, x_{n-1}]$, or to the set of SDSOS polynomials $\Sigma^{SDSOS}_p[x_0, \ldots, x_{n-1}]$. Positivity constraints translate to SDPs in the SOS case [96] and to second-order cone

[11] Optimization is performed over polynomial coefficients, so the convexity is with respect to coefficient vectors.

programs in the SDSOS case [1]; software such as YALMIP [77] and SPOTless[12] automate the translation. For a more thorough introduction to optimization in the space of positive polynomials, see e.g. the tutorial paper [2] and references therein.

The optimization formulations of `isBlocked` and `isTransient` below are conservative due to truncation of the maximal polynomial degree. Crucially, this conservatism does not result in false positives (cf. Proposition 4).

Continuous time. An upper bound of the optimal value of (BLK-CT) can be found by fixing a bound $2p$ on maximal polynomial degree, and solving

$$\begin{cases} \min\limits_{\alpha,\sigma^i_{C_j},\sigma^i_{\mathcal{D}}} & \alpha, \\ \text{s.t} & \alpha - \hat{n}(x)\cdot f(x,d) - \sum_{i\in[k_{C_1}]} \sigma^i_{C_1}(x,d) g^i_{C_1}(x) \\ & \quad - \sum_{i\in[k_{C_2}]} \sigma^i_{C_2}(x,d) g^i_{C_2}(x) \\ & \quad - \sum_{i\in[k_{\mathcal{D}}]} \sigma^i_{\mathcal{D}}(x,d) g^i_{\mathcal{D}}(d) \in \Sigma_{2p}[x,d], \\ & \sigma^i_{C_j}(x,d) \in \Sigma_{2p-p_{C_j}}[x,d],\ \forall i \in [k_{C_j}],\ j=1,2, \\ & \sigma^i_{\mathcal{D}}(x,d) \in \Sigma_{2p-p_{\mathcal{D}}}[x,d],\ \forall i \in [k_{\mathcal{D}}]. \end{cases} \tag{BLK-CT-POL}$$

The normal vector $\hat{n}(x)$ is chosen as the gradient of the level set function $g^{i^*}_{C_1}$ that induces the boundary between C_1 and C_2, $\hat{n}(x) = \pm\nabla G^{i^*}_{C_1}(x)$ with a sign that implies outward direction from C_1. In the degenerate cases where i^* is not unique any $g^i_{C_1}(x)$ that separates C_1 from C_2 can be used to obtain a blocking certificate—just as in the linear case. The optimization is over the scalar α and the positive polynomial multipliers $\sigma^i_{C_j}$ and $\sigma^i_{\mathcal{D}}$. If the optimal value of (BLK-CT-POL) is less than 0 it can be inferred that $\texttt{isBlocked}(C_1, C_2, f, \mathcal{D}) = \texttt{True}$, since $\alpha \leq 0$ implies that $0 \geq \hat{n}(x)\cdot f(x,d)$ for all $(x,d) \in (\mathrm{cl}(C_1) \cap \mathrm{cl}(C_2)) \times \mathcal{D}$.

[12] SPOTless handles SDSOS optimization through an add-on by Anirudha Majumdar available at `https://github.com/spot-toolbox/spotless`.

Discrete time. k_{C_2} optimization problems can be stated for $i \in [k_{C_2}]$,

$$\begin{cases} \min\limits_{\gamma, \sigma^j_{C_1}, \sigma^j_{\mathcal{D}}} & \gamma \\ \text{s.t} & \gamma - g^i_{C_2}(f(x,d)) - \sum_{j\in[k_{C_1}]} \sigma^j_{C_1}(x,d) g^j_{C_1}(x) \\ & \quad - \sum_{j\in[k_{\mathcal{D}}]} \sigma^j_{\mathcal{D}}(x,d) g^j_{\mathcal{D}}(d) \in \Sigma_{2p}[x,d], \\ & \sigma^j_{C_1}(x,d) \in \Sigma_{2p-p_{C_1}}[x,d],\ \forall j \in [k_{C_1}], \\ & \sigma^j_{\mathcal{D}}(x,d) \in \Sigma_{2p-p_{\mathcal{D}}}[x,d]\ \forall j \in [k_{\mathcal{D}}]. \end{cases} \tag{BLK-DT-POL-i}$$

If the optimal value of (BLK-DT-POL-i) for any i is less than 0, then $g^i_{C_2}(f(x,d)) < 0$ for all $x \in C_1$ and all $d \in \mathcal{D}$. Thus (BLK-DT) is infeasible so $\texttt{isBlocked}(C_1, C_2, f, \mathcal{D}) = \texttt{True}$. This procedure essentially amounts to over-approximating the one step reachable set of C_1 and intersecting it with C_2. If the intersection is empty no transitions are possible.

Finally, `isTransient` can be implemented using the following polynomial equivalent of (TRS):

$$\begin{cases} \text{find} & B \in \mathbb{R}_{2p}[x],\ \sigma^{i,\mu}_{C_1},\ \sigma^{i,\mu}_{\mathcal{D}} \\ \text{s.t.} & -\Delta_\mu B(x,d) - \epsilon - \sum_{i\in[k_{C_1}]} \sigma^{i,\mu}_{C_1}(x,d) g^i_{C_1}(x) \\ & \quad - \sum_{i\in[k_{\mathcal{D}}]} \sigma^{i,\mu}_{\mathcal{D}}(x,d) g^i_{\mathcal{D}}(d) \in \Sigma_{2p}[x,d],\ \ \forall \mu \in U, \\ & \sigma^{i,\mu}_{C_1}(x,d) \in \Sigma_{2p-p_{C_1}}[x,d],\ \ \forall i \in [k_{C_1}],\ \ \forall \mu \in U, \\ & \sigma^{i,\mu}_{\mathcal{D}}(x,d) \in \Sigma_{2p-p_{\mathcal{D}}}[x,d],\ \ \forall i \in [k_{\mathcal{D}}],\ \ \forall \mu \in U. \end{cases} \tag{TRS-POL}$$

If a B satisfying these conditions can be found, then $-\Delta_\mu B(x,d) - \epsilon \geq 0$ on $C_1 \times \mathcal{D}$ uniformly over $\mu \in U$, thus $\texttt{isTransient}(C_1, \{f_\mu\}_{\mu\in U}, \mathcal{D}) = \texttt{True}$.

2.3. Abstraction-Synthesis-Refinement

In order to reduce the computational effort associated with computing a fine-grained uniform partition, an algorithm that starts with a coarse partition and iteratively refines it in "promising" areas of the state space is proposed. Once an initial coarse abstraction $\mathcal{T}^+ = (\mathbb{X}, \mathbb{U}, \longrightarrow, h_{\mathbb{X}}, \mathcal{G})$ of a switched system Σ_{Sw} has been constructed, the synthesis algorithm attempts to compute a winning set $W \subset \mathbb{X}$ in which the specification can be enforced. In addition, it also computes a losing set $L \subset \mathbb{X}$ comprising initial conditions from

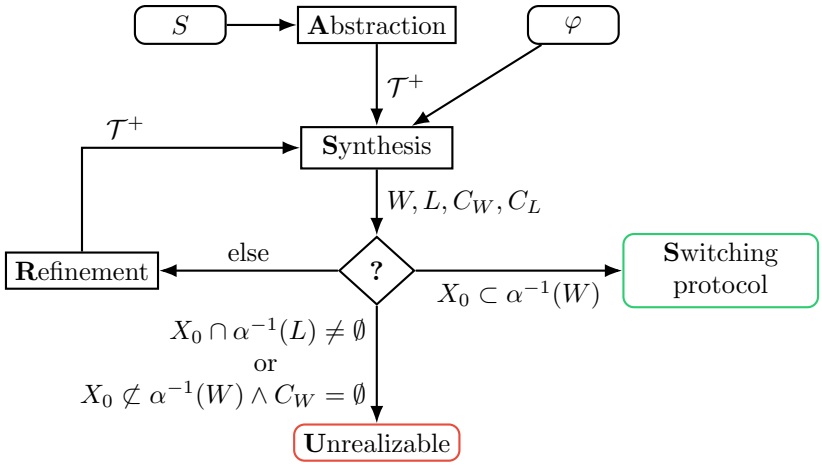

Figure 2.2: Schematic view of the abstraction-synthesis-refinement algorithm.

where there is no hope that φ can be satisfied even after further refinement. The refinement procedure is based on an initial set $X_0 \subset \mathcal{X}$; the problem is considered solved if the specification can be enforced from all initial conditions in X_0. The refinement, which produces increasingly tight over-approximations of Σ_{Sw}, is iterated until the synthesis algorithm can produce a controller (i.e., the initial set is covered by the winning set: $X_0 \subset \alpha^{-1}(W)$), until a counterexample is obtained that proves the unrealizability of the specification (i.e., the losing set intersects the initial set: $\alpha^{-1}(L) \cap X_0 \neq \emptyset$), or until the computational resources are exhausted. The algorithm, illustrated in Figure 2.2, contains three main components **Abstraction**, **Synthesis**, and **Refinement**.

The objective is to make the winning and losing sets as large as possible in order to extract as much information as possible about the realizability of φ. To this end also *candidate* winning and losing sets C_W and C_L that serve as feedback to the refinement step are computed. These sets represent areas where cell refinement may allow the actual winning and losing sets to be expanded in a later synthesis stage. In the following subsections the **Abstraction**, **Synthesis**, and **Refinement** procedures are described in detail.

2.3.1. Abstraction

To construct an initial abstraction of $\Sigma_{Sw} = (\mathcal{X}, \mathbb{U}, \mathcal{D}, \{f_\mu\}_{\mu \in \mathbb{U}}, h_\mathcal{X})$, any abstraction function α on $\mathcal{X}$ that is proposition preserving with respect to the set of atomic propositions AP suffices. Given such an abstraction function α—a natural choice is the coarsest α

that is proposition preserving—an abstracting AFTS $\mathcal{T}^+$ can be constructed following the procedure in Section 2.2.

2.3.2. Synthesis

In the synthesis step, winning, losing, and candidate winning and losing sets are computed. To define these sets, four operators $\mathrm{Win}^{\mathcal{T}^+}_{\sharp_1,\sharp_2}(\varphi)$ are introduced that take an AFTS $\mathcal{T}^+$ and an LTL formula φ and return the set of states in $\mathbb{X}$ where φ can be enforced. In $\mathrm{Win}^{\mathcal{T}^+}_{\sharp_1,\sharp_2}$, the symbols $\sharp_1$ and $\sharp_2$ are both quantifiers in the set $\{\exists, \forall\}$—thus making the number of possible combinations four—that denote whether the actions ($\sharp_1$), and the nondeterminism ($\sharp_2$), are controllable ($\exists$), or not ($\forall$). For instance, $\mathrm{Win}^{\mathcal{T}^+}_{\exists,\exists}(\varphi)$ returns the set of initial conditions from where φ can be enforced, provided that both the actions and the nondeterminism in $\mathcal{T}^+$ are controllable. Remark that such finite synthesis problems over an AFTS can be solved algorithmically for general LTL specifications (cf. Proposition 2.1).

Given the Win operators, the concepts of winning, losing, and candidate winning and losing sets are introduced as follows:

$$W = \mathrm{Win}^{\mathcal{T}^+}_{\exists,\forall}(\varphi), \qquad \text{(winning)}, \tag{2.6a}$$

$$C_W = \mathrm{Win}^{\mathcal{T}^+}_{\exists,\exists}(\varphi) \setminus W, \qquad \text{(candidate winning)}, \tag{2.6b}$$

$$L = \mathrm{Win}^{\mathcal{T}^+}_{\forall,\forall}(\neg\varphi), \qquad \text{(losing)}, \tag{2.6c}$$

$$C_L = \mathrm{Win}^{\mathcal{T}^+}_{\forall,\exists}(\neg\varphi) \setminus L, \qquad \text{(candidate losing)}. \tag{2.6d}$$

As can be seen, W and C_W are defined using $(\exists, \sharp)$ quantifiers, which corresponds to actions being controllable. The difference between the two is that nondeterminism is assumed to be uncontrollable for W, but controllable for C_W. The sets L and C_L are defined in an analogous manner for uncontrollable actions.

Given an AFTS $\mathcal{T}^+$ together with a specification φ, the synthesis step amounts to computing (2.6a)–(2.6d). These computations can be performed for general formulas using LTL synthesis algorithms, however, for structured specifications more efficient algorithms may be devised. In addition to efficiency, a second benefit of using formula-tailored algorithms is that the computational effort for consecutive synthesis steps can be reduced by using previous synthesis results; i.e., *incremental synthesis*. Thirdly, in the definitions above it can be shown that $(W \cup L)^C = C_W = C_L$[13], which implies that every discrete state that is

[13]Follows from $W^C = \mathrm{Win}^{\mathcal{T}^+}_{\forall,\exists}(\neg\varphi)$, $L^C = \mathrm{Win}^{\mathcal{T}^+}_{\exists,\exists}(\varphi)$, which implies $(W \cup L)^C = \mathrm{Win}^{\mathcal{T}^+}_{\exists,\exists}(\varphi) \cap \mathrm{Win}^{\mathcal{T}^+}_{\forall,\exists}(\neg\varphi)$.

not winning or losing is both a candidate winning and a candidate losing state. Restricting attention to specifications with specific structure makes it possible to compute localized candidate sets—a modification that potentially enhances the efficiency of the refinement procedure. In Section 2.4 tailored algorithms are provided for a meaningful fragment of LTL that enable localized refinement, as well as efficient incremental synthesis.

2.3.3. Refinement

In the event that an acceptable winning set was not found, i.e., $X_0 \not\subset \alpha^{-1}(W)$, and that infeasibility was not proven, the abstracting AFTS should be refined in the hope that a tighter over-approximation will reveal more information about the realizability of φ on Σ_{Sw}. The refinement is focused on regions of the state space where it may be beneficial, i.e., the candidate winning and losing sets.

Specifically, if at step k of the abstraction-refinement loop the current AFTS is $\mathcal{T}_k^+ = (\mathbb{X}^k, \mathbb{U}, \longrightarrow^k, h_{\mathbb{X}}^k, \mathcal{G}^k)$ with abstraction function α_k; select for $\xi \in C_W \cup C_L$ a cell $\alpha_k^{-1}(\xi)$ and split it into two parts. This results in two new discrete states ξ' and ξ''; define $\mathbb{X}^{k+1} = \big(\mathbb{X}^k \setminus \{\xi\}\big) \cup \{\xi', \xi''\}$ and a new abstraction function $\alpha_{k+1} : \mathcal{X} \longrightarrow \mathbb{X}^{k+1}$ as $\alpha_{k+1}(x) = \alpha_k(x)$ for $x \notin \alpha_k^{-1}(\xi)$, and such that the pair $\big(\alpha_{k+1}^{-1}(\xi'), \alpha_{k+1}^{-1}(\xi'')\big)$ forms a 2-cell partition of $\alpha_k^{-1}(\xi)$. The abstraction refinement function $\alpha_{k,k+1} : \mathbb{X}^{k+1} \to \mathbb{X}^k$ from $\mathcal{T}_{k+1}^+$ to $\mathcal{T}_k^+$ is implicitly defined by $\alpha_k = \alpha_{k,k+1} \circ \alpha_{k+1}$.

In order to update the transitions and progress groups for the new transition system, Algorithms 1 and 2 could be used to determine $\longrightarrow^{k+1}$ and $\mathcal{G}^{k+1}$. However, since the refinement only locally modifies the partition structure, the vast majority of all transitions and progress groups remain intact. Thus, only transitions involving the cells $\alpha_{k+1}^{-1}(\xi_{i_1})$ and $\alpha_{k+1}^{-1}(\xi_{i_2})$ need to re-computed, which reduces the required computational effort drastically. For progress groups, if $G \in \mathcal{G}^k(U)$, then $\alpha_{k+1,k}^{-1}(G)$ can be added to $\mathcal{G}^{k+1}(U)$ since the pre-image remains unchanged: $(\alpha^k)^{-1}(G) = \alpha_{k+1}^{-1}\big(\alpha_{k+1,k}^{-1}(G)\big)$. However, additional progress groups that satisfy (iii) of Definition 2.5 may have been revealed by the refinement, so a search for new progress groups may be appropriate. By construction, the following holds.

Proposition 2.3. *The refined AFTS $\mathcal{T}_{k+1}^+$ satisfies*

$$\Sigma_{Sw} \preceq \mathcal{T}_{k+1}^+ \preceq \mathcal{T}_k^+.$$

There are many ways of selecting a cell for splitting, and the splitting itself can be done

in several ways. Multiple cells can also be selected for splitting in the same refinement step. The next result shows that there are no "dead ends" in the space of all refinements, regardless of the refinement heuristics.

Proposition 2.4. *For any pair of abstracting AFTSs $\mathcal{T}_1^+ \succeq \Sigma_{Sw}$ and $\mathcal{T}_2^+ \succeq \Sigma_{Sw}$, there exists an AFTS $\mathcal{T}_3^+ \succeq \Sigma_{Sw}$ such that $\mathcal{T}_1^+ \succeq \mathcal{T}_3^+$ and $\mathcal{T}_2^+ \succeq \mathcal{T}_3^+$. In other words, there exists a refinement $\mathcal{T}_3^+$ of $\mathcal{T}_2^+$ such that $\mathcal{T}_3^+$ approximates Σ_{Sw} at least as well as $\mathcal{T}_1^+$ does.*

Proof. It is sufficient to "overlay" the partitions of $\mathcal{T}_1^+ = (\mathbb{X}_1, \mathbb{U}, \longrightarrow_1, h_{\mathbb{X}_1}, \mathcal{G}_1)$ and $\mathcal{T}_2^+ = (\mathbb{X}_2, \mathbb{U}, \longrightarrow_2, h_{\mathbb{X}_2}, \mathcal{G}_2)$ to create a new AFTS with the required property. Specifically, if α_1 and α_2 are abstraction functions for $\mathcal{T}_1^+$ and $\mathcal{T}_2^+$, let $\alpha_3 : \mathcal{X} \longrightarrow \mathbb{X}_1 \times \mathbb{X}_2$ be defined by $\alpha_3(x) = (\alpha_1(x), \alpha_2(x))$. $\longrightarrow_3$ is constructed as follows:

- For each $(\xi_1, \mu, \xi_1') \in \longrightarrow_1$, add $((\xi_1, \xi_2), \mu, (\xi_1', \xi_2'))$ to $\longrightarrow_3$ for all ξ_2 such that $(\xi_1, \xi_2) \in \alpha_3(\mathcal{X})$ and all ξ_2' such that $(\xi_1', \xi_2') \in \alpha_3(\mathcal{X})$.
- For each $(\xi_2, \mu, \xi_2') \in \longrightarrow_2$, add $((\xi_1, \xi_2), \mu, (\xi_1', \xi_2'))$ to $\longrightarrow_3$ for all ξ_1 such that $(\xi_1, \xi_2) \in \alpha_3(\mathcal{X})$ and all ξ_1' such that $(\xi_1', \xi_2') \in \alpha_3(\mathcal{X})$.

Finally, adding $\alpha_3 \circ \alpha_1^{-1}(G_1)$ and $\alpha_3 \circ \alpha_2^{-1}(G_2)$ to $\mathcal{G}_3(U)$ for all $G_1 \in \mathcal{G}_1(U)$ and all $G_2 \in \mathcal{G}_2(U)$ produces the required object as $\mathcal{T}_3^+ = (\alpha_3(\mathcal{X}), \mathbb{U}, \longrightarrow_3, h_3, \mathcal{G}_3)$, for $h_3(\xi_1, \xi_2) = h_{\mathbb{X}_1}(\xi_1) = h_{\mathbb{X}_2}(\xi_2)$ (well defined due to proposition preservation). $\mathcal{T}_3^+$ is a refinement of both $\mathcal{T}_1^+$ and $\mathcal{T}_2^+$ with refinement functions $\alpha_i(\xi_1, \xi_2) = \xi_i$ for $i = 1, 2$. □

2.4. Synthesis Algorithms for AFTSs

Now attention is restricted to the LTL fragment consisting of specifications of the form

$$\varphi = \Box a \wedge \Diamond\Box b \wedge \left(\bigwedge_{i \in I} \Box\Diamond c^i \right). \tag{2.7}$$

The objective is efficient computation of the winning, losing, and candidate sets for this type of specification. The candidate winning and losing sets require different controllability assumptions captured via a generalization of $\mathrm{Pre}^{\mathcal{T}}$ from Definition 1.7.

Consider an (augmented) transition system $\mathcal{T}^+ = (\mathcal{T}, \mathcal{G})$ and let $\sharp_1, \sharp_2$ be two quantifiers in the set $\{\exists, \forall\}$. Then the generalized backwards reachability operator for a mode set

$U \in 2^{\mathbb{U}}$ is

$$\mathrm{Pre}^{\mathcal{T}^+,U}_{\sharp_1,\sharp_2}(X) = \{\xi_1 \in \mathbb{X} : \sharp_1(\mu \in U)\ \sharp_2\ (\xi_2 \ \text{ s.t } \ (\xi_1, u, \xi_2) \in \longrightarrow),\ \ \xi_2 \in X\}. \tag{2.8}$$

In addition, both the quantified sets should be non-empty which is a subtle abuse of notation since $\forall(x \in \emptyset)$ is true by definition[14]. To exemplify, $\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\exists,\forall}(X)$ is the normal backwards reachability operator from Definition 1.7; it computes the set of all states that can be controlled to be in X at the next time step regardless of nondeterminism. Conversely, $\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\forall,\forall}(X)$ is the set of states from where every transition ends in X—regardless of action and how the nondeterminism is resolved. Using these operators as building blocks, algorithms capable of computing the winning set of a specification of type (2.7) can be constructed. In the following $\sharp$ denotes either $\exists$ or $\forall$.

In Section 1.3 fixed-point characterizations of the winning set for both φ and $\neg\varphi$ were presented. These algorithms did however not take into account the progress information encoded in an AFTS. In order to take advantage of this additional information, a backwards reachability operator that exploits progress properties to enforce transitions is required.

$$\mathrm{PGPre}^{\mathcal{T}^+}_{\exists,\forall}(Z,B) = \bigcup_{U\in 2^{\mathbb{U}}} \bigcup_{G\in\mathcal{G}(U)} \mathrm{Inv}^{U,G}_{\exists}(Z,B)\,, \tag{2.9a}$$

$$\mathrm{PGPre}^{\mathcal{T}^+}_{\forall,\forall}(Z,B) = \bigcup_{G\in\mathcal{G}(\mathbb{U})} \mathrm{Inv}^{\mathbb{U},G}_{\forall}(Z,B)\,, \tag{2.9b}$$

$$\mathrm{Inv}^{U,G}_{\sharp}(Z,B) = \begin{cases} Y_0 = (G \cap B) \setminus Z, \\ Y_{k+1} = Y_k \cap \mathrm{Pre}^{\mathcal{T}^+,U}_{\sharp,\forall}(Y_k \cup Z)\,. \end{cases} \tag{2.9c}$$

The invariance-like operator $\mathrm{Inv}^{U,G}_{\sharp,\forall}(Z,B)$ calculates a set Y_∞ contained in $G \cap B$ from where the state can either remain inside Y_∞ or enter Z. Since it can not remain inside $G \supset Y_\infty$ indefinitely due to the progress group condition, Z is eventually reached. Defined as the union over all progress groups of such sets, $\mathrm{PGPre}^{\mathcal{T}}_{\sharp,\forall}(Z,B)$ comprises all points from where progress properties can be leveraged to enforce an eventual transition to Z, while remaining in B. In the $(\forall,\forall)$ case, only progress groups over the whole mode set are taken into account; to leverage the progress property of G for $G \in \mathcal{G}(U)$ the actions must be constrained to U which is not possible if $U \neq \mathbb{U}$ and actions are uncontrollable.

[14]The precise statement is $\exists(\mu \in U)\ \sharp_1(\mu \in U)\ \exists(\xi_2 \ \text{ s.t } \ (\xi_1, u, \xi_2) \in \longrightarrow)\ \sharp_2\ (\xi_2 \ \text{ s.t } \ (\xi_1, u, \xi_2) \in \longrightarrow)$.

The fixed points given in Section 1.3 can now be generalized. Here the different levels are expressed separately in a more verbose manner, where the lower levels are parametrized by sets $Z_0, Z_1 \subset \mathbb{X}$.

$$\begin{aligned}
&\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\,(b\,\mathbf{U}\,Z_0) = \mu X\ Z_0 \cup \Big(\llbracket b \rrbracket \cap \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}(X)\Big) \cup \mathrm{PGPre}^{\mathcal{T}^+}_{\sharp,\forall}\,(X, \llbracket b \rrbracket) \\
&\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left((b\,\mathbf{U}\,Z_1) \vee \Box\left(b \wedge \left(\bigwedge_{i\in I} \Diamond c^i\right)\right)\right) \\
&= \nu W\ \bigcap_{i\in I} \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(b\,\mathbf{U}\ \left(Z_1 \cup \left(\llbracket b \rrbracket \cap \llbracket c^i \rrbracket \cap \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}\,(W)\right)\right)\right) \qquad (2.10)\\
&\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\Diamond\Box b \wedge \left(\bigwedge_{i\in I} \Box\Diamond c^i\right)\right) \\
&= \mu V\ \ \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\left(b\,\mathbf{U}\ \left(\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}(V) \cup \mathrm{PGPre}^{\mathcal{T}^+}_{\sharp,\forall}\,(V, \mathbb{X})\right)\right) \vee \Box\left(b \wedge \left(\bigwedge_{i\in I} \Diamond c^i\right)\right)\right)
\end{aligned}$$

The soundness and completeness of these algorithms are established in Appendix A (page 128-130), and, as remarked earlier, it is easy to solve for φ by restricting the state space to $\mathrm{Win}^{\mathcal{T}^+}(\Box a)$ before applying (2.10). The case $I = \emptyset$—i.e. when the property has no recurrence part—is not well defined in (2.10) but can be handled by setting $I = \{1\}$ and $C^1 =$ `True` which gives an equivalent specification.

Next, winning sets for the dual specification are considered.

$$
\begin{aligned}
&\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\bigvee_{i\in I}\left[\left(b^i \,\mathbf{U}\, Z_0\right)\vee\Box b^i\right]\right)\\
&\quad=\bigcup_{J\in 2^I}\nu\begin{bmatrix}\vdots\\ X^J\\ \vdots\end{bmatrix}\begin{bmatrix}\vdots\\ Z_0\cup\left(\left(\bigcap_{i\in J}[\![b^i]\!]\right)\cap\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}\left(\bigcup_{K\in 2^J}X^K\right)\right)\\ \vdots\end{bmatrix}\\
&\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\Diamond Z_1\vee\left(\bigvee_{i\in I}\Diamond\Box b^i\right)\right)\\
&\quad=\mu W\,\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\bigvee_{i\in I}\left[\left(b^i\,\mathbf{U}\,\left(\begin{pmatrix}Z_1\cup\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}(W)\\ \cup\mathrm{PGPre}^{\mathcal{T}^+}_{\sharp,\forall}(W,\mathbb{X})\end{pmatrix}\right)\right)\vee\Box b^i\right]\right)\\
&\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\Diamond a\vee\left(\bigvee_{i\in I}\Diamond\Box b^i\right)\vee\Box\Diamond c\right)\\
&\quad=\nu V\;\mathrm{Win}_{\sharp,\forall}\left(\Diamond\left([\![a]\!]\cup\left([\![c]\!]\cap\mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}(V)\right)\right)\vee\left(\bigvee_{i\in I}\Diamond\Box b^i\right)\right).
\end{aligned}
\tag{2.11}
$$

The proofs of correctness for these fixed points are in Appendix B on pages 130–132. By using duality, winning sets for both φ and $\neg\varphi$ can be computed also in the $(\sharp,\exists)$ cases. Define $\bar{\sharp}$ as the quantifier dual to $\sharp$, that is, $\bar{\exists}=\forall$ and $\bar{\forall}=\exists$. Then,

$$
\begin{aligned}
\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\exists}\left(\Box a\wedge\Diamond\Box b\wedge\bigwedge_{i\in I}\left(\Box\Diamond c^i\right)\right)&=\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\Diamond\neg a\vee\left(\bigvee_{i\in I}\Diamond\Box\neg c^i\right)\vee\Box\Diamond\neg b\right)^C,\\
\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\exists}\left(\Diamond a\vee\left(\bigvee_{i\in I}\Diamond\Box b^i\right)\vee\Box\Diamond c\right)&=\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\Box\neg a\wedge\Diamond\Box\neg c\wedge\left(\bigwedge_{i\in I}\Box\Diamond\neg b^i\right)\right)^C.
\end{aligned}
$$

Remark 2.2. *There is an interesting separation between winning sets that depend on progress group information, and those that do not. In particular, only smallest fixed-point algorithms (expressed with μ) leverage progress groups but not greatest fixed points (expressed with ν). As illustrated in Figure 2.3;* $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\Diamond b)\neq\mathrm{Win}^{\mathcal{T}}_{\sharp,\forall}(\Diamond b)$ *in general, but* $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\Box a)=\mathrm{Win}^{\mathcal{T}}_{\sharp,\forall}(\Box a)$[15]. *The underlying intuition is that progress groups enforce live-*

[15]The winning set of $\Box a$ can be computed for instance using the following special case: $W_0=Q$, $W_{k+1}=[\![a]\!]\,\mathbf{U}\,\left([\![a]\!]\cap\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}(W_k)\right)$. By induction it can be shown that W_{k+1} converges to $[\![a]\!]\cap\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}(W_k)$

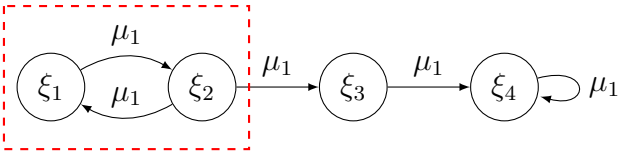

Figure 2.3: For the AFTS $\mathcal{T}^+ = (\mathcal{T}, \mathcal{G})$ illustrated above, if $\{\xi_1, \xi_2\} \in \mathcal{G}(\mu_1)$, then $\mathrm{Win}^{\mathcal{T}^+}_{\exists,\forall}(\Diamond\{\xi_4\}) = \{\xi_1, \xi_2, \xi_3, \xi_4\}$. However, the winning set of the same specification without the progress group information is just $\mathrm{Win}^{\mathcal{T}}_{\exists,\forall}(\Diamond\{\xi_4\}) = \{\xi_3, \xi_4\}$, since $\xi_1\xi_2\xi_1\xi_2\ldots$ is a valid trajectory of $\mathcal{T}$ that never reaches ξ_4.

ness conditions; they therefore affect the winning set of a liveness specification $\Diamond b$, but not that of a safety specification $\Box a$. By duality, the converse holds in the $(\sharp, \exists)$-cases.

2.4.1. Incremental Synthesis

Equipped with the fixed-point characterizations from the previous subsections, algorithms to compute a winning set W and a losing set L for a specification of the form (2.7) immediately follow. These algorithms provide efficient means of computing (2.6a) and (2.6c), as opposed to solving a general synthesis problem with the progress groups encoded in LTL. By unwinding the fixed points on which the winning and losing sets are defined, also *localized* versions of the candidate winning and losing sets are obtained. These are more specific than their general counterparts (2.6b) and (2.6d) and can serve as feedback to guide the refinement step, as illustrated in Figure 2.2.

Winning set $(\exists, \forall)$. The winning set is $W = \mathrm{Win}^{\mathcal{T}^+}_{\exists,\forall}(\varphi)$, so it can be computed using (2.10).

Losing set $(\forall, \forall)$. The losing set L is equal to $\mathrm{Win}^{\mathcal{T}^+}_{\forall,\forall}(\neg\varphi)$, which can be computed using (2.11).

Candidate winning set $(\exists, \exists)$. The winning set computation (2.10) is performed as a nested fixed-point operation; any enlargement of the inner fixed points may potentially enlarge the final winning set as well. By systematically unwrapping the fixed-point algorithms involved in (2.10), as presented in detail in Appendix A.2, the following candidate

after one iteration which implies that no progress group information is used.

winning set C_W is obtained that is typically much smaller than its general counterpart (2.6b):

$$\begin{aligned} C_W = &\left(\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\exists,\exists}(V_\infty) \setminus V_\infty\right) \cup (W_1 \setminus V_1) \\ &\cup \left(\llbracket b \rrbracket \bigcap \left(\bigcup_{i\in I} \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\exists,\exists}(W_{1,i}) \setminus W_{1,i}\right)\right) \cup (\llbracket a \rrbracket \setminus V_{inv}). \end{aligned} \tag{2.12}$$

Here, $V_{inv} = \mathrm{Win}^{\mathcal{T}^+}_{\exists,\forall}(\Box a)$ and V_1, V_∞, are intermediate results in (2.10). In addition,

$$W_1 = \bigcap_{i\in I} W_{1,i}, \quad W_{1,i} = \mathrm{Win}^{\mathcal{T}^+}_{\exists,\forall}\left(b \,\mathbf{U}\, \left(b \wedge c^i\right)\right).$$

Candidate losing set $(\forall,\exists)$**.** Finally, a localized heuristic version of (2.6d) is given as follows:

$$C_L = (V_1 \setminus V_\infty) \cup \left(\mathrm{Pre}^{\mathcal{T},\mathbb{U}}_{\forall,\exists}(V_1) \setminus V_1\right) \cup \left(\bigcup_i \llbracket b^i \rrbracket \setminus W_1\right). \tag{2.13}$$

This set consists of states that are potential members of the losing set in a refined AFTS. Again, V_1 and V_∞ are intermediate results from (2.11), and

$$W_1 = \mathrm{Win}^{\mathcal{T}^+}_{\exists,\exists}\left(\bigvee_{i\in I}\left[\Box b^i \vee \left(b^i \,\mathbf{U}\, (a \vee c)\right)\right]\right).$$

A motivation behind this choice of C_L is provided in Appendix A.2.

Apart from being able to extract localized candidate sets, an advantage of the algorithms presented above is that certain winning set computations can be performed incrementally over refinement cycles (cf. Figure 2.2). Smallest fixed points of an expanding algorithm always grow after refinement. Therefore, fixed points from previous iterations can be mapped to the refined partition and the algorithm can be initialized with these sets to speed up convergence to the smallest fixed point. To exemplify, let $V^t_\infty = \mathrm{Win}^{\mathcal{T}^+_t}_{\exists,\forall}\left(\Diamond\Box b \wedge \left(\bigwedge_{i\in I}\Box\Diamond c^i\right)\right)$. Then, if $\mathcal{T}^+_t \succeq \mathcal{T}^+_{t+1}$ with a refinement function β, $V^{t+1}_\infty = \mathrm{Win}^{\mathcal{T}^+_{t+1}}_{\exists,\forall}\left(\Diamond\Box b \wedge \left(\bigwedge_{i\in I}\Box\Diamond c^i\right)\right)$ can be computed using a modified version of (2.10):

$$V^{t+1}_\infty = \begin{cases} V_0 = \beta^{-1}(V^t_\infty), \\ V_{k+1} = \mathrm{Win}^{\mathcal{T}^+_{t+1}}_{\exists,\forall}\left(\left(b \,\mathbf{U}\, \mathrm{Pre}^{\mathcal{T}^+_{t+1},\mathbb{U}}_{\exists,\forall}(V_k)\right) \vee \Box\left(b \wedge \left(\bigwedge_{i\in I}\Diamond c^i\right)\right)\right). \end{cases}$$

In the same way, results for $\mathcal{T}^+_t$ of other expanding fixed-point computations can be saved

to get a "warm start" when computing the same smallest fixed points for $\mathcal{T}^+_{t+1}$. This does however not apply for contracting algorithms since the greatest fixed point for a contracting algorithm is smaller in $\mathcal{T}^+_t$ than in $\mathcal{T}^+_{t+1}$.

2.4.2. Controller Extraction

For controller implementation, a controller π that enforces the specification in the winning set W must be extracted. All fixed points encountered when computing W are ultimately defined in terms of $\text{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\exists,\forall}$. It is therefore enough to extract control actions during calls to this operator and map the results into higher-level fixed points together with memory variables that control a hierarchy of partial reachability objectives. In particular, this is done by extracting the actions that satisfy the existence condition $\exists \mu \in \mathbb{U}$ in (2.8). If the synthesis problem is solved with a general LTL solver, a controller is typically obtained together with the winning set.

Remark 2.3. *Certain algorithms in Section 2.4 are based on taking set complements. In these cases an enforcing strategy can not be extracted; only the winning set is obtained. However, primal algorithms are available for all necessary specifications in the* $(\exists, \forall)$*-case that is of interest when computing* W*.*

As shown in Theorem 2.1, a controller π for the AFTS $\mathcal{T}^+ = (\mathbb{X}, \mathbb{U}, \longrightarrow, h_{\mathbb{X}}, \mathcal{G}) \succeq \Sigma_{Sw}$ can be implemented as a switching protocol on Σ_{Sw}. There are in general several winning actions for any given state of π. This control freedom can used to pursue performance objectives, or to eliminate potential continuous-time Zeno behavior, as described next.

Zeno behavior. For continuous-time switched systems, it may happen that the solution found for $\mathcal{T}^+$ induces Zeno behavior when implemented on Σ_{Sw}, as illustrated in Figure 2.4. In addition to being undesirable due to its non-physical properties, Zeno behavior prohibits maximality of continuous-time trajectories, and thus prevents liveness-type specifications from being fulfilled. On the contrary, if Zeno behavior is not induced, the continuous switching signal is piecewise constant which ensures maximality of trajectories. Below several strategies for eliminating Zeno behavior are discussed. For simplicity, consider a memoryless[16] controller $\pi_0 : \mathbb{X} \longrightarrow 2^{\mathbb{U}}$. It induces a closed loop graph $T_{\pi_0} = (\mathbb{X}, E_{\pi_0})$ with

[16]For controllers with (finite) memory, the same analysis can be done using lifts and projections with respect to a space where states are augmented with the internal controller state.

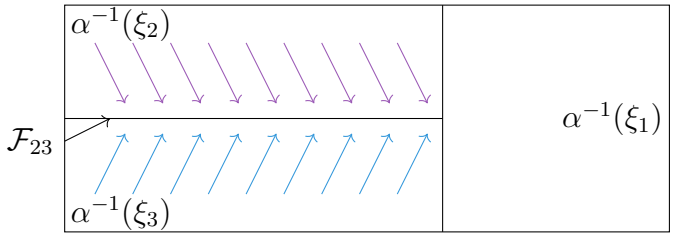

Figure 2.4: Illustration of an abstraction that induces Zeno behavior. The arrows correspond to vector fields for two different actions. Let a discrete controller pick the purple action in ξ_2 and the blue action in ξ_3. Implementing an abstraction-based switching protocol to reach $\alpha^{-1}(\xi_1)$ from $\alpha^{-1}(\xi_2) \cup \alpha^{-1}(\xi_3)$ might induce Zeno behavior at the facet $\mathcal{F}_{23}$.

edge set E_{π_0} given by

$$E_{\pi_0} = \bigcup_{\xi \in \mathbb{X}} \{(\xi, \xi') : (\xi, \mu, \xi') \in \longrightarrow \text{ for some } \mu \in \pi_0(\xi)\}.$$

This graph describes (an over-approximation of) all discrete transitions that might occur at runtime. A simple cycle $(\xi_0, \xi_1, \ldots, \xi_{k-1})$ in T_{π_0} is said to be *Zeno-prone* if there exists a point that is a limit point of all corresponding partition cells, i.e. $\bigcap_{i \in [k]} \mathrm{cl}\,(\alpha^{-1}(\xi_i)) \neq \emptyset$. All cycles of length 2 are automatically Zeno-prone. Since Zeno-prone cycles are by definition a local concept in the graph (the maximal length of a Zeno-prone cycle is the maximum number of jointly adjacent cells in the abstraction partition) it is significantly faster to enumerate all Zeno-prone cycles than to enumerate all simple cycles.

If there are no Zeno-prone cycles, Zeno behavior is not present in Σ_{Sw}. Specifically, if a restricted controller $\bar{\pi}_0 : \mathbb{X} \longrightarrow 2^{\mathbb{U}}$ can be found such that $\bar{\pi}_0(\xi) \subset \pi_0(\xi)$ for all ξ, and such that its induced graph $T_{\bar{\pi}_0}$ is free from Zeno-prone cycles, implementing $\bar{\pi}_0 \circ \alpha$ as a feedback controller on the continuous time switched system results in both correct and Zeno-free behavior.

Next a heuristic for eliminating Zeno behavior is proposed that, in addition, reduces the number of switches which may be beneficial in non-ideal systems where switches don't occur instantaneously. Given a controller $\pi : \mathbb{X} \times \mathbb{M} \longrightarrow 2^{\mathbb{U}}$ with internal state m, define for trajectory histories $\zeta = \zeta(0)\zeta(1)\ldots\zeta(k-1)$ and $\sigma = \sigma(0)\sigma(1)\ldots\sigma(k-1)$ a controller

$\tilde{\pi}$ as

$$\tilde{\pi}(\zeta, \sigma, \zeta(k)) = \begin{cases} \sigma(k-1), & \text{if } \sigma(k-1) \in \pi(\zeta(k), m(k)), \\ \text{any } \mu \in \pi(\zeta(k), m(k)) & \text{otherwise.} \end{cases}$$

In essence, this controller uses the same action until it is necessary to select another one in order to ensure future correctness. If Zeno behavior is still present when using $\tilde{\pi}$ on Σ_{Sw}, further refinement of Zeno-prone areas may be necessary in order to eliminate Zeno behavior.

Finally, a third option to ensure that trajectories are free from Zeno behavior is to time-discretize the continuous-time system and instead solve a discrete-time synthesis problem. When implementing the resulting controller as a continuous-time switching protocol, the sample time used in the time-discretization can be used as the controller sampling rate to ensure a minimum dwell time for switching, and thus render Zeno behavior impossible. Under Lipschitz assumptions on the vector fields, margins can be added to (for safety)—or removed from (for liveness)—the atomic propositions to ensure that the final closed-loop system satisfies the specification in continuous time.

2.5. Examples

In this section two examples are presented that demonstrate the effectiveness of the proposed approach. The first example compares the performance of FTSs to that of AFTSs, while the second example compares the proposed abstraction-refinement procedure to the approach in [121] that uses non-incremental synthesis on a uniform state-space partition partition. The examples are solved with a prototype implementation that partitions the state space into hyper boxes. It is available at `https://github.com/pettni/abstr-refinement`.

2.5.1. Numerical Example

Consider a continuous-time switched system, taken from [95], with four modes $u \in \mathbb{U} = \{0, 1, 2, 3\}$ and with the following dynamics:

$$f_0 = \begin{bmatrix} -x_1 - 1.5x_0 - 0.5x_0^3 \\ x_0 - x_1^2 + 2 + d \end{bmatrix}, \qquad f_1 = \begin{bmatrix} -x_1 - 1.5x_0 - 0.5x_0^3 \\ x_0 - x_1 + d \end{bmatrix},$$
$$f_2 = \begin{bmatrix} -x_1 - 1.5x_0 - 0.5x_0^3 + 2 \\ x_0 + 10 + d \end{bmatrix}, \quad f_3 = \begin{bmatrix} -x_1 - 1.5x_0 - 0.5x_0^3 - 1.5 \\ x_0 + 10 + d \end{bmatrix},$$

where d is the disturbance. The domain is taken to be $X = [-2, 2] \times [-1.5, 3]$. Define the sets $X_a = [-2, -0.5] \times [-1.5, -1]$ and $X_b = [-1, -0.2] \times [0.5, 1.8]$ with associated atomic propositions $a = x \notin X_a$ and $b = x \in X_b$. The goal is to synthesize a switching protocol to guarantee the satisfaction of the specification $\varphi = \Box a \wedge \Diamond\Box b$.

The proposed incremental abstraction-refinement algorithm using AFTSs was executed to find a switching protocol together with a winning set from where this protocol is guaranteed to satisfy the specification. Two cases were considered: one with a bounded disturbance level $|d| \leq 0.5$, as well as the case without disturbance $|d| = 0$. As a comparison, the same algorithm was also executed using FTSs without progress groups, where transience was used only to eliminate self-loops. Figure 2.5 shows the resulting winning sets after 80 iterations. The results clearly indicate that using AFTSs as abstractions reduces the conservatism. It is also worth mentioning that when using AFTSs, the algorithm terminated after 19 iterations in the case without disturbance, and after 70 iterations in the case with disturbance, since no candidate sets were left and the exact maximal winning set of the continuous problem was recovered.

For this example, the `isBlocked` and `isTransient` functions were computed based on an SDSOS-based implementation of the programs (BLK-CT-POL) and (TRS-POL).

2.5.2. Radiant Systems in Buildings

The second example is a hydronic radiant system for buildings in which chilled supply water is pumped through a concrete slab which in turn acts as a heat reservoir that moderates the building temperature. By controlling the pump it is possible to regulate the temperature

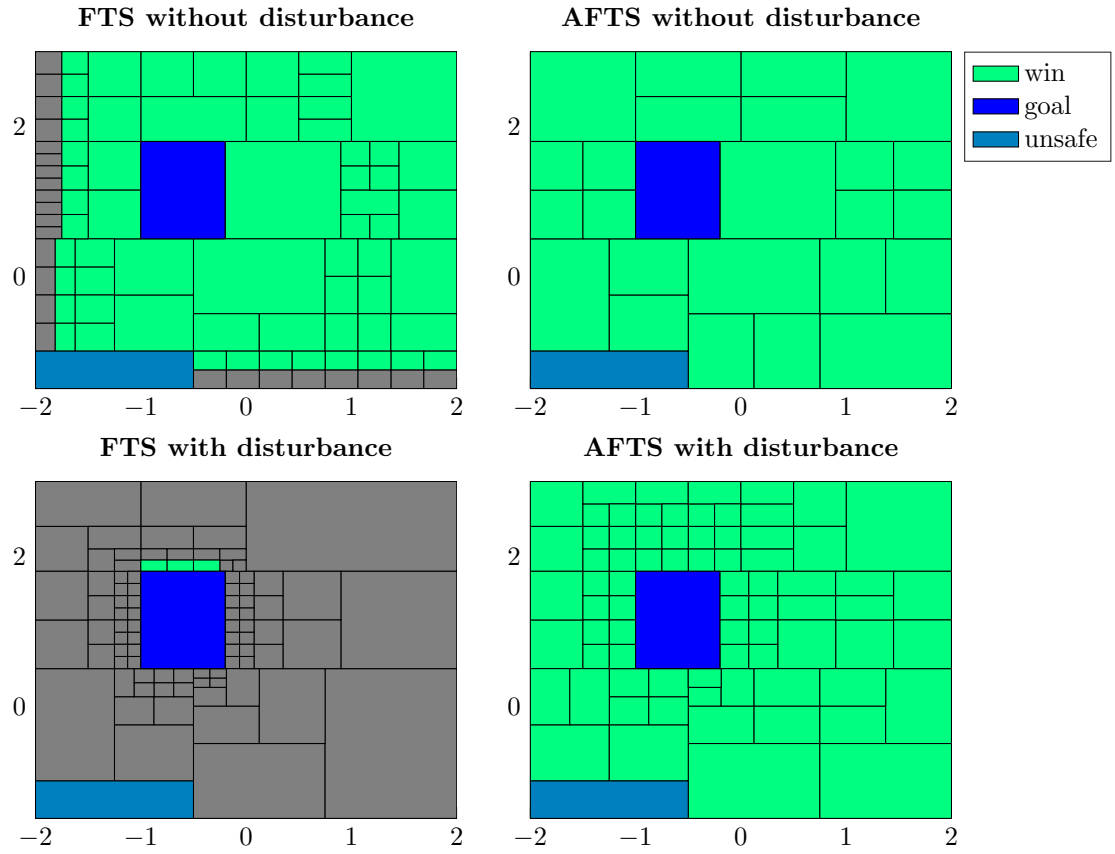

Figure 2.5: Four different abstractions that illustrate the advantage of progress groups. Winning sets computed using the proposed incremental abstraction refinement framework are shown in green. For this example abstractions based on AFTSs outperform FTS-based abstractions by allowing a larger winning set to be computed using fewer discrete states. The incremental refinement process creates a finer partition in relevant areas of the state space.

in the building. Such a system can be modeled as an RC circuit [110]

$$C_i \frac{\mathrm{d}}{\mathrm{d}t} T_i = \sum_{j \neq i} \frac{1}{R_{ij}} (T_j - T_i) + q_i,$$

where C_i is the thermal capacitance $[J/K]$ of zone i that is assumed to have homogeneous temperature T_i, $R_{ij} = R_{ji}$ is the thermal resistance $[K/W]$ of the barrier between zone i and zone j, and q_i is the heat added $[W]$ to zone i. The added heat may for example come from persons utilizing a room, or sunshine, which both vary over the course of a day. To describe a system with two rooms connected to the same hydronic system, consider

Table 2.1: Parameter values for hydronic heating.

Room 1 floor dimensions	$A_{1c} = 8m \times 6m$	$R_{1c} = \frac{r_{1c}}{A_{1c}} = 0.0026\frac{K}{W}$
Room 1 area to outside	$A_{1o} = 24m^2$	$R_{1o} = \frac{r_{1o}}{A_{1o}} = 0.0491\frac{K}{W}$
Room 2 floor dimensions	$A_{2c} = 6m \times 6m$	$R_{2c} = \frac{r_{2c}}{A_{2c}} = 0.0035\frac{K}{W}$
Room 2 area to outside	$A_{2o} = 18m^2$	$R_{2o} = \frac{r_{2o}}{A_{2o}} = 0.0654\frac{K}{W}$
Interconnecting wall area	$A_{12} = 18m^2$	$R_{12} = \frac{r_{12}}{A_{12}} = 0.0439\frac{K}{W}$
Total slab area	$A_c = 84m^2$	$R_{cw} = \frac{r_{wc}}{A_c} = 0.0012\frac{K}{W}$
Room 1 volume	$V_1 = 144m^3$	$C_1 = 5c_a\rho_a V_1 = 8.774 \times 10^5\frac{J}{K}$
Room 2 volume	$V_2 = 108m^3$	$C_2 = 5c_a\rho_a V_2 = 6.580 \times 10^5\frac{J}{K}$
Slab volume	$V_c = 10.5m^3$	$C_c = c_c\rho_c V_c = 3.623 \times 10^7\frac{J}{K}$
Nominal heat gains	$q_1 = \left(6\frac{W}{m^2}\right) \times A_{1c}$	$q_2 = \left(8\frac{W}{m^2}\right) \times A_{2c}$

a model with five temperature nodes; temperatures T_1 and T_2 for the two rooms, T_c for the concrete slab, T_o for the building exterior, and T_w for the supply water. The latter two temperatures are assumed to be constant. If an uncertain amount of heat $q_i + \Delta\mathbf{q}_i(t)$ is added to room i, the dynamics of mode 1 representing a pump that is turned on are written as follows:

$$\begin{aligned}
C_c\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}_c &= \sum_{i=1}^{2}\frac{1}{R_{ic}}\left(\mathbf{T}_i - \mathbf{T}_c\right) + \frac{1}{R_{cw}}\left(T_w - \mathbf{T}_c\right), \\
C_1\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}_1 &= \frac{1}{R_{1c}}\left(\mathbf{T}_c - \mathbf{T}_1\right) + \frac{1}{R_{1o}}\left(T_o - \mathbf{T}_1\right) + \frac{1}{R_{12}}\left(\mathbf{T}_2 - \mathbf{T}_1\right) + q_1 + \Delta\mathbf{q}_1, \\
C_2\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{T}_2 &= \frac{1}{R_{2c}}\left(\mathbf{T}_c - \mathbf{T}_2\right) + \frac{1}{R_{2o}}\left(T_o - \mathbf{T}_2\right) + \frac{1}{R_{12}}\left(\mathbf{T}_1 - \mathbf{T}_2\right) + q_2 + \Delta\mathbf{q}_2.
\end{aligned} \tag{2.14}$$

The dynamics when the water pump is turned off (mode 2) are identical to those of mode 1 but with $R_{cw} = +\infty$. Thermal capacitance is easy to compute for a solid material like concrete[17]; for the rooms the thermal capacitance is computed from air volume, and furniture and other objects are compensated for by multiplying with a factor 5 [120]. Formulas and standard parameters from [51] are used to calculate thermal heat insulation coefficients[18] r_{ij} which give the thermal resistances as $R_{ij} = r_{ij}/A_{ij}$ for the corresponding wall or floor area. The inner wall heat insulation coefficient is taken to be $r_{12} = 0.79m^2K/W$ [120]. In Table 2.1 the dimensions of the zones are listed along with the associated resistances and capacitances.

[17]The formula is $c\rho V$, where c is specific heat $[J/kgK]$, ρ is density $[kg/m^3]$, and V is volume $[m^3]$.
[18]$r_{1o} = r_{2o} = 1.178m^2K/W$, $r_{1c} = r_{2c} = 0.125m^2K/W$, $r_{cw} = 0.102m^2K/W$.

Assume that the outdoor temperature is $T_o = 30°C$, that the water temperature is $T_w = 18°C$, and that the uncertainty is bounded as $|\Delta q_i| \leq \left(3\frac{W}{m^2}\right) \times A_i$ for $i = 1, 2$. Set the domain to $20 \leq T_c, T_1, T_2 \leq 28$ and consider as objective to control the room temperatures to a desired range, denoted by a proposition SET ($21 \leq T_c \leq 27$ and $22 \leq T_1, T_2 \leq 25$), and guarantee invariance in this range, captured by the LTL formula $\varphi_2 = \Diamond\Box SET$. The problem is under-actuated and the equilibrium points of (2.14) are outside of SET, so it does not have a trivial solution.

Applying the abstraction-synthesis-refinement loop yields a winning set that covers 76% of the domain after 1200 iterations, which takes about 45 minutes on a modern desktop computer. After the winning set has been found the part represented by $\mathrm{Win}_{\exists,\forall}(\Box SET)$ is further refined to eliminate Zenoness of the controller. Figure 2.6 shows a simulation trace of the corresponding controller under disturbance, and a 3D illustration of trajectories converging to the target set.

The example above requires using multi-mode progress groups to compute a reasonably sized winning set. Indeed, the approach in [121] fails to find a winning set for this problem for relatively large uniform abstractions. For comparison, the same problem without disturbance and with the parameter set from [121] is also considered. That paper proposes efficient algorithms for synthesis; the bottle neck is creating the abstraction which for this problem takes 4.5 hours for an abstraction with 4000 states. Solving the synthesis problem on the resulting abstraction gives a winning set that covers 63% of the domain. In contrast, with the abstraction-synthesis-refinement loop it is on the same computer possible to incrementally construct a non-uniform abstraction with a winning set that covers 64% of the domain in 1 minute. The abstraction has 270 states at this stage which shows the benefit of the incremental and non-uniform approach. With additional iterations it is possible to further enlarge the winning set to cover 83% of the domain.

2.6. Conclusions

This chapter treated AFTSs and described how they can be used to construct abstractions of continuous-state switched systems, both in discrete and continuous time. The advantage of AFTSs over standard FTSs is the possibility to encode liveness properties of the continuous dynamics in progress groups, which eliminates spurious trajectories in the abstraction and thus provides a tighter over-approximation. The procedure for constructing

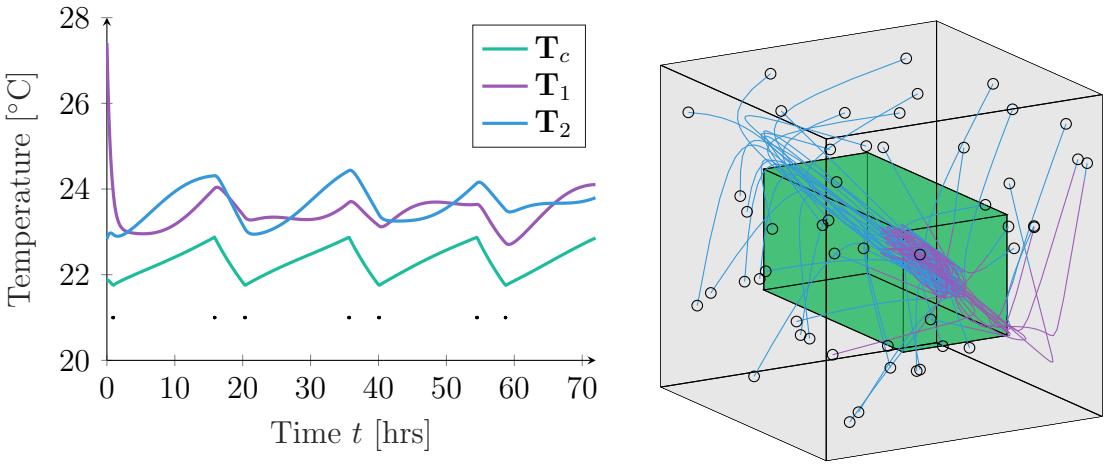

Figure 2.6: Trajectories of the radiant system. Left: Plot of temperatures versus time when each room is subject to an independent sinusoidal disturbance that models varying utilization of the room throughout the day. As can be seen, the temperatures converge to the desired ranges and remain there. The black dots at the bottom indicate switches. Right: Plot showing how 50 sample trajectories converge to the target *SET* (green) starting from different parts of the winning set. Different trajectory colors correspond to different control actions.

an abstracting AFTS hinges upon the implementation of two subroutines `isBlocked` and `isTransient`, for which convex optimization-based implementations were provided in two important cases: linear and polynomial dynamics.

The control synthesis problems for general LTL specifications on an AFTS was discussed, and an abstraction-synthesis-refinement loop was proposed which refines an over-approximating AFTS until a satisfactory solution—or a counter-example—is found. Efficient AFTS synthesis algorithms for an important fragment of LTL were also provided. The benefits of using these algorithms include incremental synthesis and localized partition refinement, both reducing the required computational effort compared to using general-purpose synthesis tools. Methods to eliminate Zeno behavior in continuous-time systems when implementing a switching protocol obtained from an abstraction were also discussed, and the method was illustrated on two examples.

Design choices in an implementation can have significant effects on the practical scalability of the algorithm; in particular how the AFTS is stored in memory. The implementation used to solve the examples in this chapter was based on an explicit sparse representation; an alternative is to use Binary Decision Diagrams (BDDs). While BDDs reduce the mem-

ory footprint significantly, the time required for synthesis is sometimes increased. For this reason tools such as SCOTS [113] provide both options. Current work is underway to explore BDDs as an alternative data structure and investigate the best way to encode information via binary variables. Other future directions include identifying classes of nonlinear dynamical systems for which the `isBlocked` and `isTransient` subroutines can be implemented even more efficiently than the positive polynomial-based approach considered in this work (e.g., [131]), and combining the theory with abstraction decomposition ideas [33, 82].

Chapter 3.

Decomposition for Formal Synthesis

A natural way to approach a large-scale problem is to decompose it into smaller subproblems, such that each subproblem is of lower dimensionality and thus easier to solve. Crucially, the decomposition must be done in a principled way in order for the sub-solutions to constitute a valid solution to the original problem, as illustrated in Figure 3.1. A popular framework for compositional analysis is *assume-guarantee reasoning*, where each subcomponent is endowed with a "contract" specifying its allowed behavior [18, 24]. In particular, an assume-guarantee contract for a subsystem contains assumptions on the behavior of other subsystems, and guarantees on the subsystem's own behavior as it relates to other subsystems. Also the system environment can incorporated as a component with behavior restricted by a guarantee.

There are more aspects to decomposition than scalability. First of all, depending on how the assume-guarantee contracts are synthesized, the compositional approach may be conservative compared to a monolithic solution as shown in the following example:

Example 3.1. *Consider the following two-dimensional system:*

$$\begin{bmatrix}\mathbf{x}_1(k+1)\\ \mathbf{x}_2(k+1)\end{bmatrix} = \begin{bmatrix}1 & 0.2\\ 0.2 & 1\end{bmatrix}\begin{bmatrix}\mathbf{x}_1(k)\\ \mathbf{x}_2(k)\end{bmatrix} + \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\mathbf{u}_1(k)\\ \mathbf{u}_2(k)\end{bmatrix}, \quad -0.1 \leq \mathbf{u}_1(k), \mathbf{u}_2(k) \leq 0.1. \quad (3.1)$$

Below, separable and centralized computation of an invariant set contained in $[-1, 1]^2$ *are compared.*

The separable case is symmetric; consider subsystem 1 which is as follows:

$$\mathbf{x}_1(k+1) = \mathbf{x}_1(k) + \mathbf{u}_1(k) + \underbrace{0.2\mathbf{x}_2(k)}_{disturbance}, \quad -0.1 \leq \mathbf{u}_1(k) \leq 0.1. \quad (3.2)$$

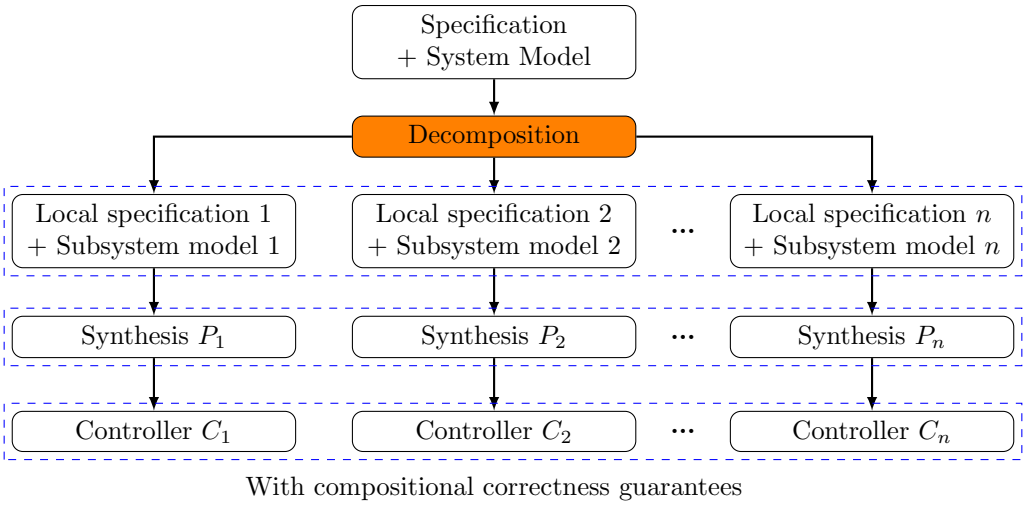

Figure 3.1: Decomposition of a formal synthesis problem. Scalability can be improved by decomposing a large problem into smaller pieces.

Restricting attention to symmetric contracts, the objective is to find the largest number $\bar{x}$ *such that* $[-\bar{x}, \bar{x}]$ *is controlled invariant for subsystem* (3.2) *under the assumption that* $-\bar{x} \leq \mathbf{x}_2 \leq \bar{x}$*. Elementary calculations reveal that the maximal such number is* $\bar{x} = 0.5$*, so the maximal symmetric separable controlled invariant set is* $\mathcal{C}_1 = [-0.5, 0.5]^2$ *with volume 1.*

In the centralized case the machinery from Section 1.2.2 and 1.3.1 can be employed which results in a (non-separable) controlled invariant set $\mathcal{C}_2$ *with volume 3. The two invariant sets are displayed in Figure 3.2.*

Although conservatism is undesirable in itself, system modularity may in many cases be desirable. For instance, if the plant of one subsystem is modified, the local controller can be updated to satisfy the same contract without need to re-design other components. Assume-guarantee contracts may also facilitate debugging: by monitoring satisfaction of contracts it is easy to determine where in the system a fault occurs.

In addition to assume-guarantee contracts, a second crucial aspect of a decomposition framework is assumptions on communication. If the current state of one subsystem is made available, other subsystem can use that information in their pursuit to satisfy their own contracts. Again there is a trade-off between conservativeness and modularity: transmission of state information has the potential to make the approach less conservative (compared to a monolithic controller), but comes with increased subsystem interdependence.

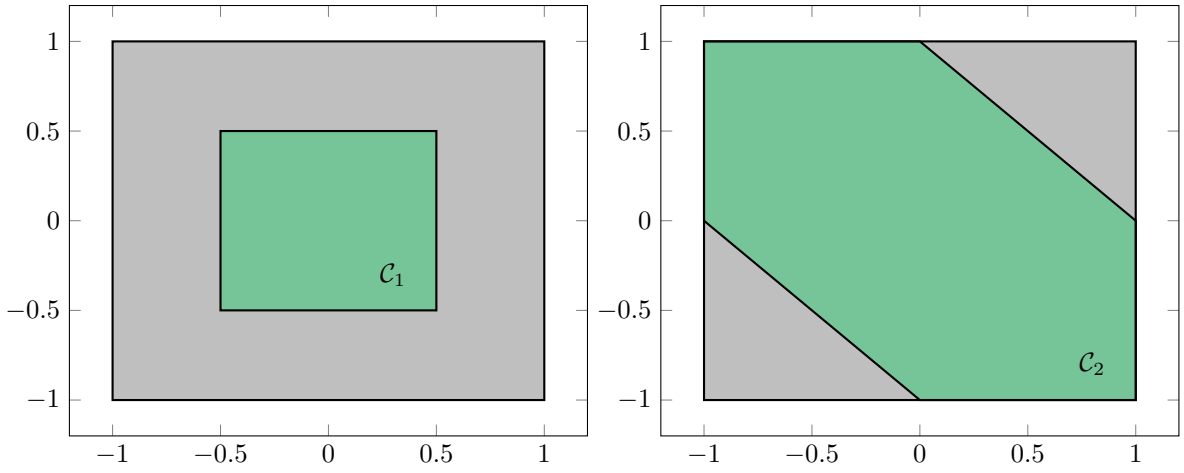

Figure 3.2: Illustration of separable and centralized invariant sets. Invariant sets are plotted in green and are contained in the box $[-1,1]^2$ (gray). The separable set (left) is $\mathcal{C}_1 = [-0.5, 0.5] \times [-0.5, 0.5]$; the interval $[-0.5, 0.5]$ is controlled invariant for (3.2). The larger centralized invariant set $\mathcal{C}_2$ (right) is controlled invariant for (3.1).

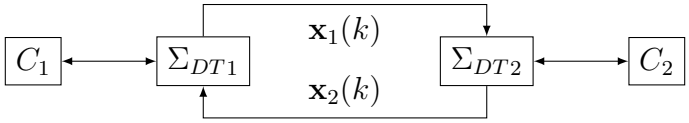

Figure 3.3: Example of an assume-guarantee interconnection. If the controller C_1 for Σ_{DT1} enforces $\square(\mathbf{x}_2 \in X_2) \implies \square(\mathbf{x}_1 \in X_1)$ and the controller C_2 for Σ_{DT2} enforces $\square(\mathbf{x}_1 \in X_1) \implies \square(\mathbf{x}_2 \in X_2)$ the composition satisfies $\square(\mathbf{x}_1 \in X_1 \wedge \mathbf{x}_2 \in X_2)$ for appropriate initial conditions.

Example 3.2. *Let Σ_{DT1} and Σ_{DT2} be two discrete-time systems with states $\mathbf{x}_1$ and $\mathbf{x}_2$ and state update functions $f_1(x_1, x_2, u_1)$ and $f_2(x_1, x_2, u_2)$ as illustrated in Figure 3.3. Suppose that they are endowed with two assume-guarantee contracts*

$$\begin{aligned} &\textit{Contract for } \Sigma_{DT1}: \ \square(\mathbf{x}_2 \in X_2) \Longrightarrow \square(\mathbf{x}_1 \in X_1), \\ &\textit{Contract for } \Sigma_{DT2}: \ \square(\mathbf{x}_1 \in X_1) \Longrightarrow \square(\mathbf{x}_2 \in X_2). \end{aligned}$$

Consider the task of subsystem Σ_{DT1}: it needs to keep its state $\mathbf{x}_1$ inside of X_1 under the assumption that $\mathbf{x}_2$ is kept in X_2. If the current state $\mathbf{x}_2(k)$ is available, a controller for Σ_{DT1} must enforce invariance of a set $Y_1 \subset X_1$ with the property that for all $x_1 \in Y_1$

$$\forall x_2 \in X_2 \ \exists u_1 \in \mathcal{U}_1 \ f_1(x_1, x_2, u_1) \in Y_1. \tag{3.3}$$

On the other hand, if the current state $\mathbf{x}_2(k)$ *is unknown, the controller for* Σ_{DT1} *must find an input that works for all possible* $x_2 \in X_2$*;*

$$\exists u_1 \in \mathcal{U}_1 \; \forall x_2 \in X_2 \; f_1(x_1, x_2, u_1) \in Y_1. \tag{3.4}$$

The switch of quantifiers illustrates that (3.4) *is a more difficult control problem: a* u_1 *satisfying* (3.4) *also satisfies* (3.3)*, but the opposite is in general not true.*

A system decomposition is essentially a partition of the system states into subsystems; where each subsystem is responsible for controlling its own states. Given a system decomposition and an overall specification, the distributed synthesis problem is to find a set of feasible assume-guarantee contracts along with local controllers that enforce the contracts under appropriate information transmission assumptions.

In this chapter, for a collection of N interconnected and constrained discrete-time systems Σ_{DTn} for $n \in [N]$, the objective is to calculate contracts $\mathcal{C}_n$ in the form of polyhedral sets such that the n'th subsystem can enforce the contract

$$\bigwedge_{\substack{m \in [N] \\ m \neq n}} \square \left(\mathbf{x}_m \in \mathcal{C}_m \right) \Longrightarrow \square \left(\mathbf{x}_n \in \mathcal{C}_n \right). \tag{3.5}$$

Provided that initial conditions are within these sets, i.e. $\mathbf{x}_n(0) \in \mathcal{C}_n$ for all $n \in [N]$, and that each subsystem enforces its contract, it follows that the overall invariance condition $\wedge_{n \in [N]} \square \left(\mathbf{x}_n \in \mathcal{C}_n \right)$ is enforced.

A contract of the form (3.5) contains inherent trade-offs. In general it is desirable that the sets $\mathcal{C}_n$ are as large as possible since they represent the set of initial conditions from where the overall invariance specification can be enforced, and, furthermore, it is easier to enforce invariance of a larger set. However, if one of the left-hand side sets $\mathcal{C}_m$ in (3.5) grows, subsystem n must be robust to a wider range of effects from subsystem m, and is therefore in general able to enforce invariance of a comparatively smaller $\mathcal{C}_n$[19]. This implies that as opposed to a centralized invariant set, there is no notion of maximality when searching for this type of separable invariant set.

[19] Suppose that $\mathcal{C}_n$ is the maximal controlled invariant set inside $\mathcal{X}_n$. Then $\mathcal{C}_n$ grows with the size of $\mathcal{X}_n$, and decreases with the size of $\mathcal{X}_m$.

Chapter overview. This chapter suggests two ways of synthesizing assume-guarantee contracts for invariance specifications, for discrete-time linear systems. As opposed to previous work which has been mostly within the realm of finite systems, polyhedral sets are used to define contracts for systems with continuous state spaces. The first method, presented in Section 3.1, is a centralized approach based on LMIs. Although the centralized computation is performed on the full state space, LMIs can be solved in polynomial time, and the resulting assume-guarantee contracts can be used to decompose a synthesis problem with a more sophisticated specification. The following Section 3.2 outlines a fixed point-based method for determining feasibility of a given contract—and refinement of the contract if it can not be satisfied. Both methods are flexible in terms of information sharing and can incorporate constraints on states and controls, as well as exogenous disturbance. The methods are illustrated with examples in Section 3.3 and the chapter is summarized in Section 3.4.

Related work. Previous work on correct-by-construction decomposition has mostly focused on the invariance problem (Section 1.3.1); assume-guarantee contracts for invariance can be represented by a single time-invariant set for each subsystem. Synthesis of contracts for finite systems was considered by Dallal and Tabuada [32] who presented a decentralized cyclic algorithm that computes *minimally restrictive assumptions* on other subsystems; the procedure can be seen as a gradual refinement of assume-guarantee contracts until consensus is reached among the subsystems. Since there is no structural restriction imposed on the contracts the algorithm is non-conservative, i.e., it retrieves the winning set of the centralized algorithm. However, for the same reason there is no significant gain in scalability since computations are performed on the full state space. Refinement of assume-guarantee contracts for general LTL specifications was considered in [7].

The paper [105] by Rakovic et. al. is perhaps the closest work to the theory in this chapter, where "practical set invariance" is considered via a form of centralized polyhedral contract. This notion of invariance lies in between the completely separated approach in this chapter, and the general contracts in [32], and thus also represents a middle way in terms of conservatism. The approach requires pre-defined template sets and reduces the invariance problem for N interconnected systems to invariance in an N-dimensional system whose states are scaling factors of the template sets.

A related topic is construction of compositional abstractions that mitigate the state space explosion problem [33, 101]. Also here the system is divided into subsystems and each

subsystem treats effects from neighboring subsystems as disturbance. A flexible approach was recently presented in [82] where individual states can "overlap" between subsystems, which allows retaining more precise dynamics at the cost of higher-dimensional subsystems.

3.1. Decomposable Invariant Sets via LMIs

In this section a computational procedure for synthesizing separable controlled invariant sets via LMIs is presented. The standard algorithm for invariance is to iterate the fixed point (1.13); the problem can typically not be solved in a single-shot computation since the problem of joint synthesis of an invariant set and a controller that enforces invariance is bilinear in the decision variables. However, as proposed by Tahir and Jaimoukha in [125], the problem can be linearized via a series of LMI manipulations at the cost of slight conservatism. The method presented here is an extension of this work to a decentralized setting, and to sets with tunable complexity.

The type of system considered in the following consists of N interconnected linear subsystems with states $\mathbf{x}_n(k) \in \mathbb{R}^{n_x^n}$ that affect each other as additive disturbance:

$$\Sigma_{DTn} : \mathbf{x}_n(k+1) = A_{nn}\mathbf{x}_n(k) + \sum_{\substack{m\in[N]\\ m\neq n}} A_{nm}\mathbf{x}_m(k) + B_n\mathbf{u}_n(k) + E_n\mathbf{d}_n(k). \tag{3.6}$$

As can be seen, each subsystem is associated with an input $\mathbf{u}_n(k) \in \mathbb{R}^{n_u^n}$ and an exogenous disturbance $\mathbf{d}_n(k) \in \mathbb{R}^{n_d^n}$; both which do not affect other subsystems. The full system can be written

$$\Sigma_{DT} : \mathbf{x}(k+1) = A\mathbf{x}(k) + B\mathbf{u}(k) + E\mathbf{d}(k), \tag{3.7}$$

for

$$A = \begin{bmatrix} A_{00} & \cdots & A_{0(N-1)} \\ \vdots & \ddots & \vdots \\ A_{(N-1)0} & \cdots & A_{(N-1)(N-1)} \end{bmatrix},\ B = \begin{bmatrix} B_0 & & \\ & \ddots & \\ & & B_{(N-1)} \end{bmatrix},\ E = \begin{bmatrix} E_0 & & \\ & \ddots & \\ & & E_{(N-1)} \end{bmatrix}. \tag{3.8}$$

The inputs of (3.6) are assumed to be bounded as

$$\mathbf{u}_n(k) \in \mathcal{U}_n = \{u_n : H_u^n u_n \leq h_u^n\}, \quad H_u^n \in \mathbb{R}^{\mathcal{N}_u^n \times n_u^n}, \tag{3.9}$$

for all n, and there are symmetric bounds imposed on disturbance;

$$\mathbf{d}_n(k) \in \mathcal{D}_n = \{d_n : -\mathbf{1} \leq H_d^n d_n \leq \mathbf{1}\}, \tag{3.10}$$

where $H_d^n \in \mathbb{R}^{n_d^n \times n_d^n}$ is a non-singular square matrix. Also state constraints of the form

$$\mathbf{x}_n(k) \in \mathcal{X}_n = \{x_n : H_s^n x_n \leq h_s^n\}, \quad H_s^n \in \mathbb{R}^{\mathcal{N}_s^n \times n_x^n}, \tag{3.11}$$

are incorporated. The objective is to compute sets $\mathcal{C}_n \subset \mathcal{X}_n$ that satisfy (3.5). The special symmetric description

$$\mathcal{C}_n = \{x_n : -\mathbf{1} \leq Z_n H_x^n x_n \leq \mathbf{1}\}$$

is imposed on the sets, where $Z_n \in \mathbb{R}^{\mathcal{N}_x^n \times n_x^n}$ is an arbitrary given matrix and $H_x^n \in \mathbb{R}^{n_x^n \times n_x^n}$ is non-singular. The matrix Z_n should be thought of as a set of "generators" that define a symmetric polyhedron that is linearly transformed by H_x^n to form $\mathcal{C}_n$. A natural way to select Z_n is to pick (randomly or evenly spaced) vectors from the unit sphere in $\mathbb{R}^{n_x^n}$. Let Z be the block diagonal matrix formed by all generators:

$$Z = \begin{bmatrix} Z_0 & & \\ & \ddots & \\ & & Z_{N-1} \end{bmatrix}.$$

The following result states that if a set of LMIs are satisfied, then separable controlled invariant sets and controllers that enforce control invariance can be extracted. LMIs can be solved in polynomial time via semi-definite programming [22].

Theorem 3.1. *Let $\mathcal{N}_x = \sum_{n\in[N]} \mathcal{N}_s^n$, $\mathcal{N}_s = \sum_{n\in[N]} \mathcal{N}_s^n$, $\mathcal{N}_u = \sum_{n\in[N]} \mathcal{N}_u^n$ and consider the following LMIs*

$$\hat{K} = \begin{bmatrix} \hat{K}_0 \\ \vdots \\ \hat{K}_{N-1} \end{bmatrix}, \; H_x^{-1} = \begin{bmatrix} (H_x^0)^{-1} & & \\ & \ddots & \\ & & (H_x^{N-1})^{-1} \end{bmatrix}, \; \Lambda = \begin{bmatrix} \lambda_0 I_{n_x^0} & & \\ & \ddots & \\ & & \lambda_{N-1} I_{n_x^{N-1}} \end{bmatrix} \succ 0, \tag{3.12a}$$

$$\begin{bmatrix} \Xi_j - \Phi_j^{-1} & \Omega_j^1 - \begin{bmatrix} H_x^{-1} & 0 \\ 0 & H_x^{-1} \end{bmatrix} & \Omega_j^2 - \begin{bmatrix} H_x^{-1} & 0 \\ 0 & H_x^{-1} \end{bmatrix} & \Psi_j^T \begin{bmatrix} Z^T e_j \\ Z^T e_j \end{bmatrix} \\ * & 2\begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda \end{bmatrix} - \Gamma_j & \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda \end{bmatrix} + (\Omega_j^1)^T - \Psi_j & 0 \\ * & * & \Omega_j^2 + (\Omega_j^2)^T - \Xi_j & 0 \\ * & * & * & \lambda_{n(j)} - \mathbf{1}^T D_x^j \mathbf{1} - \mathbf{1}^T D_d^j \mathbf{1} \end{bmatrix} \succ 0, \qquad \forall j \in [\mathcal{N}_x], \tag{3.12b}$$

$$\begin{bmatrix} \Gamma_j & \Psi_j \\ * & \Xi_j \end{bmatrix} \succ 0, \qquad \forall j \in [\mathcal{N}_x], \tag{3.12c}$$

$$D_x^j \succ 0, \; \begin{bmatrix} \begin{bmatrix} Z^T D_x^j Z & 0 \\ * & D_d^j \end{bmatrix} \begin{bmatrix} -\frac{1}{2}(H_x^{-T} A^T + \hat{K}^T B^T) & 0 \\ 0 & -\frac{1}{2} H_d^{-T} E^T \end{bmatrix} \\ * \qquad\qquad \begin{bmatrix} \Phi_j^{-1} \end{bmatrix} \end{bmatrix} \succ 0, \qquad \forall j \in [\mathcal{N}_x], \tag{3.12d}$$

$$D_s^k \succ 0, \; \begin{bmatrix} Z^T D_s^k Z & -\frac{1}{2} H_x^{-T} H_s^T e_k \\ * & e_k^T h_s - \mathbf{1}^T D_s^k \mathbf{1} \end{bmatrix} \succ 0, \qquad \forall k \in [\mathcal{N}_s], \tag{3.12e}$$

$$D_u^l \succ 0, \; \begin{bmatrix} Z^T D_u^l Z & -\frac{1}{2} \hat{K}^T H_u^T e_l \\ * & e_l^T h_u - \mathbf{1}^T D_u^l \mathbf{1} \end{bmatrix} \succ 0, \qquad \forall l \in [\mathcal{N}_u], \tag{3.12f}$$

where $n(j)$ is the smallest index n such that $j < \sum_{k=0}^{n} \mathcal{N}_x^k$. Consider also structural requirements on some of the variables; namely that D_x^j, D_s^k, and D_u^l are diagonal, and that Φ_j^{-1}, Γ_j and Ξ_j are symmetric.

Provided that these conditions hold, the product of the sets $\mathcal{C}_n = \{x_n : -\mathbf{1} \leq Z_n H_x^n x_n \leq \mathbf{1}\}$ is controlled invariant, and the state feedback controllers $\mathbf{u}_n(k) = \hat{K}_n H_x \mathbf{x}_n(k)$ enforce invariance.

Additional structural requirements can be imposed on the feedback matrix $\hat{K}$ to model information constraints. In particular, if the (n, m)'th block of $\hat{K}$ is constrained to zero, then the controller for subsystem n does not depend on the state of subsystem m. If $\hat{K}$ is block-diagonal the controllers do not depend on the states of other subsystems which means that each subsystem is robustly controlled invariant with respect to the state of other subsystems, as well as with respect to the exogenous disturbance. A proof of the result is given in Appendix B.1.

While the above result yields feedback controllers that enforce invariance, there is typically additional control freedom that can be leveraged to pursue additional control tasks while also maintaining invariance.

Theorem 3.2. *Consider a solution of* (3.12) *and subsystem n. Let J_n^1 be the set of subsystems whose state information is available (non-zero block in $\hat{K}_n$), and let J_n^2 be the set of subsystems whose state is not available (zero block in $\hat{K}_n$), such that $J_n^1 \cup J_n^2 = [N] \setminus \{n\}$. Then the set of $\mathbf{u}_n(k)$ such that*

$$
\begin{aligned}
H_u^n \, \mathbf{u}_n(k) &\leq h_u^n, \\
Z_n H_x^n B \, \mathbf{u}_n(k) &\leq \mathbf{1} - Z_n H_x^n \left(A_{nn} \mathbf{x}_n(k) + \sum_{m \in J_n^1} A_{nm} \mathbf{x}_m(k) \right) \\
&\quad - \sum_{m \in J_n^2} \max_{x_m \in \mathcal{C}_m} Z_n H_x^n A_{nm} x_m - \max_{d_n \in \mathcal{D}_n} Z_n H_x^n E_n d_n, \\
-Z_n H_x^n B \, \mathbf{u}_n(k) &\leq \mathbf{1} + Z_n H_x^n \left(A_{nn} \mathbf{x}_n(k) + \sum_{m \in J_n^1} A_{nm} \mathbf{x}_m(k) \right) \\
&\quad + \sum_{m \in J_n^2} \min_{x_m \in \mathcal{C}_m} Z_n H_x^n A_{nm} x_m + \min_{d_n \in \mathcal{D}_n} Z_n H_x^n E_n d_n,
\end{aligned}
\tag{3.13}
$$

is non-empty for any $\prod_{n \in [N]} \mathbf{x}_n(k) \in \prod_{n \in [N]} \mathcal{C}_n$, and any such $\mathbf{u}_n(k)$ enforces invariance of $\prod_{n \in [N]} \mathcal{C}_n$. The min *and* max *should be taken row-wise (c.f. equation* (1.7)*).*

Proof. Invariance is enforced from $\prod_{n\in[N]} \mathbf{x}_n(k)$ if there for each $n \in [N]$ exist $\mathbf{u}_n(k) \in \mathcal{U}_n$ such that for all $\mathbf{d}_n(k) \in \mathcal{D}_n$ and all $\mathbf{x}_m \in \mathcal{C}_m$ for $m \in J_2^n$;

$$-\mathbf{1} \leq Z_n H_x^n \left(A_{nn}\mathbf{x}_n(k) + \sum_{m\neq n} A_{nm}\mathbf{x}_m(k) + B_n\mathbf{u}_n(k) + E_n\mathbf{d}_n(k) \right) \leq \mathbf{1}. \tag{3.14}$$

Existence of such a $\mathbf{u}_n(k)$ that depends on states in J_1^n but not on those in J_2^n follows from Theorem 3.1. The first inequality in (3.13) ensures that $\mathbf{u}_n(k) \in \mathcal{U}_n$ and the remaining two inequalities follow from reshuffling the two inequalities in (3.14) and considering the worst-case effects of non-measurable disturbance. □

3.2. Decomposable Invariant Sets via Polyhedral Computation

Next, an iterative approach to computation of separable invariant sets is presented. As opposed to above, this method can account for nonlinear system interconnections as well as additive interconnections.

The type of system considered below consists of N interconnected systems of the form

$$\Sigma_{DTn}: \ \mathbf{x}_n(k+1) = \left(A_{nn} + \sum_{m\neq n} A_{nm}(\mathbf{x}_m(k)) \right) \mathbf{x}_n(k) + B_n\mathbf{u}_n(k) + \sum_{m\neq n} F_{nm}(\mathbf{x}_m(k)), \tag{3.15}$$

where the mappings $x_m \mapsto A_{nm}(x_m)$ and $x_m \mapsto F_{nm}(x_m)$ are potentially nonlinear.

Remark 3.1. *The dynamics can readily be extended with additive exogenous disturbance $E_n\mathbf{d}_n(k)$ for a more general description*

$$\mathbf{x}_n(k+1) = \left(A_{nn} + \sum_{m\neq n} A_{nm}(\mathbf{x}_m(k)) \right) \mathbf{x}_n(k) + B_n\mathbf{u}_n(k) + \sum_{m\neq n} F_{nm}(\mathbf{x}_m(k)) + E_n\mathbf{d}_n(k).$$

This extra term is omitted to simplify notation; it can also be incorporated as an additional subsystem in the $\sum_{m\neq n} F_{nm}$ term.

The computation of invariant sets is based on iterating the fixed point (1.13); to achieve this the backwards reachability operator from Section 1.2.2 must be extended to account for system interconnections and nonlinearities which is the objective of the following sections.

Fix a given subsystem n and assume that for all $m \neq n$ there is a finite collection $\{f_{nm}^j\}_{j\in[q_{nm}]}$ of q_{nm} non-constant functions $f_{nm}^j : \mathcal{C}_m \to \mathbb{R}$, linearly independent on $\mathcal{C}_m$, such that $A_{nm}(x_m)$ can be represented as $A_{nm}(x_m) = \sum_{j\in[q_{nm}]} f_{nm}^j(x_m)A_{nm}^j$ and $F_{nm}(x_m) = \sum_{j\in[q_{nm}]} f_{nm}^j(x_m)F_{nm}^j$, where A_{nm}^j and F_{nm}^j are constant matrices for all $j \in [q_{nm}]$. Note that such a collection $\{f_{nm}^j\}_{j\in[q_{nm}]}$ with at most $n^2 + n$ members always exists, but q_{nm} can in general be much smaller than $n^2 + n$. Define the function $f_{nm} : \mathcal{C}_m \to \mathbb{R}^{q_{nm}}$ as

$$f_{nm} : x_m \mapsto [f_{nm}^0(x_m)\ f_{nm}^1(x_m) \dots f_{nm}^{q_{nm}-1}(x_m)]^T,$$

which accounts for all the nonlinearities from subsystem m to subsystem n.

For information set indices J_n^1 (information available) and J_n^2 (information unavailable), the following generalized backwards reachable set operator is required to compute the n'th invariant set:

$$\mathrm{Pre}^{\Sigma_{DT^n}}(X) = \left\{ \begin{array}{l} x_n : (\forall m \in J_n^1,\ \forall x_m^1 \in \mathcal{C}_m)(\exists u_n \in \mathcal{U}_n)\ (\forall m \in J_n^2,\ \forall x_m^2 \in \mathcal{C}_m) \\ \left(A_{nn} + \sum_{m\neq n} \sum_{j\in[q_{nm}]} f_{nm}^j(x_m)A_{nm}^j \right) x_n + B_n u_n + \sum_{m\neq n} \sum_{j\in[q_{nm}]} f_{nm}^j(x_m)F_{nm}^j \in X \end{array} \right\}.$$

That is, the subsystems in J_n^1 are treated as a *measurable* disturbance, so the control is allowed to depend on these state values. On the contrary, the only information about the *non-measurable* subsystems in J_n^2 is that they are contained in the sets $\mathcal{C}_m$ so the control action must be robust with respect to all possible values in these sets.

3.2.1. Removal of Nonlinearities via Convexification

The next lemma shows that the nonlinearities associated with f_{nm} can be accounted for in a linear system where the nonlinearities have been moved to a set description $f_{nm}(\mathcal{C}_m)$. In addition, there is no loss of precision if the convex hull of $f_{nm}(\mathcal{C}_m)$ is considered. Once this result is established, it is extended to a re-formulation of the backwards reachable set operator in Corollary 3.1 below.

Lemma 3.1. *Let X be a convex set, then*

$$\forall x_m \in \mathcal{C}_m, \quad \left(A_{nn} + \sum_{j\in[q_{nm}]} f_{nm}^j(x_m)A_{nm}^j \right) x_n + \sum_{j\in[q_{nm}]} f_{nm}^j(x_m)F_{nm}^j \in X, \tag{3.16}$$

if and only if

$$\forall d_{nm} \in \text{conv}(f_{nm}(\mathcal{C}_m)), \quad \left(A_{nn} + \sum_{j \in [q_{nm}]} d_{nm}^j A_{nm}^j \right) x_n + \sum_{j \in [q_{nm}]} d_{nm}^j F_{nm}^j \in X. \tag{3.17}$$

Proof. (3.16) can be rewritten as

$$\forall d_{nm} \in f_{nm}(\mathcal{C}_m), \quad \left(A_{nn} + \sum_{j \in [q_{nm}]} d_{nm}^j A_{nm}^j \right) x_n + \sum_{j \in [q_{nm}]} d_{nm}^j F_{nm}^j \in X. \tag{3.18}$$

The implication from (3.17) to (3.16) is now obvious since $f_{nm}(\mathcal{C}_m) \subset \text{conv}\,(f_{nm}(\mathcal{C}_m))$. Suppose that (3.16) holds but not (3.17). Then there is $d_{nm} \in \text{conv}(f_{nm}(\mathcal{C}_m))$ such that

$$\left(A_{nn} + \sum_{j \in [q_{nm}]} d_{nm}^j A_{nm}^j \right) x_n + \sum_{j \in [q_{nm}]} d_{nm}^j F_{nm}^j \notin X.$$

By definition of the convex hull, $d_{nm} = \sum_l \alpha_l d_l$ for $\alpha_l \geq 0$, $\sum_l \alpha_l = 1$, and $d_l \in f_{nm}(\mathcal{C}_m)$. But from (3.18);

$$\begin{aligned} &\left(A_{nn} + \sum_{j \in [q_{nm}]} d_{nm}^j A_{nm}^j \right) x_n + \sum_{j \in [q_{nm}]} d_{nm}^j F_{nm}^j = \\ &= \sum_l \alpha_l \left(\left(A_{nn} + \sum_{j \in [q_{nm}]} d_l^j A_{nm}^j \right) x_n + \sum_{j \in [q_{nm}]} d_l^j F_{nm}^j \right) \in X \end{aligned}$$

by convexity of X—a contradiction. □

Corollary 3.1. *For a convex $\mathcal{U}_n$, the backwards reachable set operator can be written as follows:*

$$\text{Pre}^{\Sigma_{DT_n}}(X) = \left\{ \begin{array}{l} x_n : (\forall m \in J_n^1,\ \forall d_{nm} \in \text{conv}(f_{nm}(\mathcal{C}_m)))\ (\exists u_n \in \mathcal{U}_n) \\ \qquad (\forall m \in J_n^2,\ \forall d_{nm} \in \text{conv}(f_{nm}(\mathcal{C}_m))) \\ \left(A_{nn} + \sum_{m \neq n} \sum_{j \in [q_{nm}]} d_{nm}^j A_{nm}^j \right) x_n + B_n u_n + \sum_{m \neq n} \sum_{j \in [q_{nm}]} d_{nm}^j F_{nm}^j \in X \end{array} \right\}. \tag{3.19}$$

Proof (sketch). The corollary follows by twice applying Lemma 3.1. The first application to convexify the sets associated with J_n^2 is straightforward since other parts (i.e. terms related to J_n^1 and u_n) can be taken as constant. To substitute the sets associated with J_n^1, non-relevant parts of the expression can be moved to the set description X by eliminating the inner $\forall$ and $\exists$ quantifiers, to arrive at an expression like in Lemma 3.1. By convexity of $\mathcal{U}_n$ the modified set X is convex. □

Corollary 3.2. *If* $\text{conv}(f_{nm}(\mathcal{C}_m))$ *is replaced by* V_{nm} *in* (3.19) *for* $V_{nm} \supset \text{conv}(f_{nm}(\mathcal{C}_m))$*, then an inner approximation of* $\text{Pre}^{\Sigma_{DT^n}}(X)$ *is obtained.*

As a consequence of these results, the nonlinearities in f_{nm} can be moved to the set description if the convexified image of $\mathcal{C}_m$ under f_{nm} can be computed or over-approximated. Utilizing over-approximations of the sets $\text{conv}\,(f_{nm}(\mathcal{C}_m))$ results in inner approximations of $\text{Pre}^{\Sigma_{DT^n}}(X)$. Such inner approximations still yield sound (but not complete) winning sets when employed in fixed-point computations (c.f. Section 1.3).

Next, two explicit techniques for computing an over-approximation of the convex hull $\text{conv}(f(\mathcal{C}))$ for a multivariate function $f : \mathcal{C} \to \mathbb{R}^{n_d}$ and a convex set $\mathcal{C} \subset \mathbb{R}^p$ are presented.

Convex-Hull Computation With Monotone Functions

The first technique computes an over-approximation of $\text{conv}(f(\mathcal{C}))$ provided that f satisfies certain monotonicity conditions and that $\mathcal{C}$ is a hyper rectangle.

Monotonicity is defined with respect to a given *cone*, which is a set $K \subset \mathbb{R}^n$ such that $x, y \in K$ implies $x + y \in K$, and such that $x \in K$ implies $tx \in K$ for all scalars $t \geq 0$. A cone K induces a partial ordering $\leq_K$ on $\mathbb{R}^n$ given by

$$x \leq_K y \quad \iff \quad y - x \in K.$$

An *interval* with respect to a cone K is a convex set X for which there exist extreme points $x^-, x^+ \in X$ such that $x^- \leq_K x \leq_K x^+$ for all $x \in X$; intervals are denoted $[x^-, x^+]_K$. For instance, when the cone K corresponds to one of the orthants in $\mathbb{R}^n$, intervals are hyper rectangles.

Given two cones $K_1 \subset \mathbb{R}^p$ and $K_2 \subset \mathbb{R}^q$, a function $f : \mathbb{R}^p \to \mathbb{R}^q$ is *monotone* on the set X with respect to K_1 and K_2 if for all $x, y \in X$, $x \leq_{K_1} y$ implies that $f(x) \leq_{K_2} f(y)$. A simple condition to verify monotonicity of a function $f^i : \mathbb{R}^p \to \mathbb{R}$ is the sign stability of its gradient:

Proposition 3.1. *Given a hyper rectangle $\mathcal{C} \subset \mathbb{R}^p$, a mapping $f^i : \mathcal{C} \to \mathbb{R}$ is monotone on $\mathcal{C}$ if each element of the gradient of f^i maintains its sign on $\mathcal{C}$. That is, for all j, there exists $\sigma_j \in \{0, 1\}$ such that*

$$(-1)^{\sigma_j} \frac{\partial f^i}{\partial x_j} \geq 0 \quad \forall v \in \mathcal{C}, \tag{3.20}$$

then f^i is monotone with respect to the cones $K_1^i = \{x = [x^1, \ldots, x^p]^\intercal \in \mathbb{R}^p \mid (-1)^{\sigma_j} x_j \geq 0\}$, $K_2 = [0, \infty)$. Moreover, if all components f^i, $i \in [q]$, of a function $f : \mathcal{C} \to \mathbb{R}^q$ are monotone with respect to some cones K_1^i and $K_2 = [0, \infty)$, then $f(\mathcal{C}) \subset \prod_{i\in[p]}[f^i(\underline{x}^i), f^i(\overline{x}^i)]$, where $\underline{x}^i$ and $\overline{x}^i$ are extreme points of $\mathcal{C}$ with respect to the order induced by K_1^i.

Note that since $\prod_{i\in[p]}[f^i(\underline{x}^i), f^i(\overline{x}^i)]$ in the above proposition is a convex set, it contains $\text{conv}(f(\mathcal{C}))$. Given f and $\mathcal{C}$, if the conditions in the above proposition fail, i.e., the gradients of component functions of f are not sign stable on $\mathcal{C}$, but there is a finite cover $\{\mathcal{C}_l\}_{l\in[L]}$ of $\mathcal{C}$ with $\mathcal{C} = \bigcup_{l\in[L]} \mathcal{C}_l$ and each $\mathcal{C}_l$ is a hyper rectangle where the conditions of the proposition hold, then it is still possible to find a convex over-approximation of $\text{conv}(f(\mathcal{C}))$. This is done by over-approximating the convex hull of each $f(\mathcal{C}_l)$ with a hyper rectangle using the proposition above, and then taking the convex hull of these hyper rectangles. The next result shows that, under certain conditions, it is possible to obtain an arbitrarily tight over-approximation of $\text{conv}(f(\mathcal{C}))$ by covering $\mathcal{C}$ with hyperboxes of decreasing size. The idea is illustrated in the left part of Figure 3.4.

Theorem 3.3. *Let $\{\mathcal{C}_l\}_{l\in[L]}$ be a hyper rectangular cover of $\mathcal{C}$, i.e., $\mathcal{C} = \bigcup_{l\in[L]} \mathcal{C}_l$ and each $\mathcal{C}_l$ is a hyper rectangle. Assume that the gradients of the component functions of f are sign stable on each $\mathcal{C}_l$. Also, let $[\mathcal{C}_l]_\epsilon$ be a finite hyper rectangular cover of $\mathcal{C}_l$, where the size of the maximum edge of each hyper rectangle $R \in [\mathcal{C}_l]_\epsilon$ is at most ϵ and $\mathcal{C}_l = \bigcup_{R\in[\mathcal{C}_l]_\epsilon} R$. Then, $\text{conv}(f(\mathcal{C})) = \lim_{\epsilon\to 0} \text{conv}(\{\{f(R)\}_{R\in[\mathcal{C}_l]_\epsilon}\}_{l\in[L]})$.*

Proof. The result follows from the following claims: i) for any sets A and B, $\text{conv}(A \cup B) = \text{conv}(\text{conv}(A) \cup \text{conv}(B))$, and ii) for sets A and B such that $h(A, B) \leq \epsilon$, it holds that $h(\text{conv}(A), \text{conv}(B)) \leq \epsilon$, where h is the Hausdorff distance function.

Claim i) follows directly from the definition of convex hulls. To show ii), take any $a \in \text{conv}(A)$; it can be written as $a = \sum_{i\in I} \alpha_i a_i$ with $a_i \in A$ and $\sum_{i\in I} \alpha_i = 1$, $\alpha_i \geq 0$. For each a_i there is a $b_i \in B$ s.t. $\|a_i - b_i\| \leq \epsilon$. Evidently, $b = \sum_{i\in I} \alpha_i b_i \in \text{conv}(B)$, and $\|a - b\| = \|\sum_i \alpha_i (a_i - b_i)\| \leq \sum_{i\in I} \alpha_i \|a_i - b_i\| \leq \epsilon$, which shows $h(\text{conv}(A), \text{conv}(B)) \leq \epsilon$. □

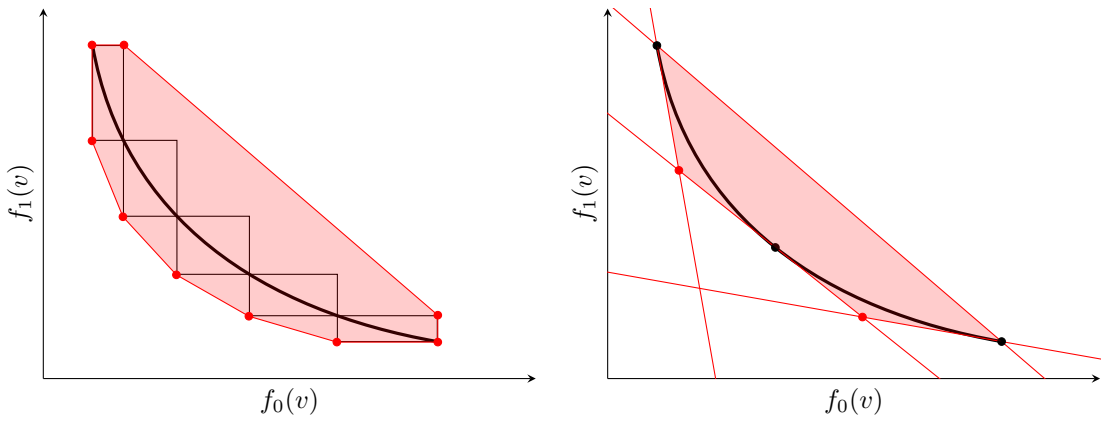

Figure 3.4: Convex hull over-approximation of images of a function. The convex hull is constructed around a nonlinear curve $f : [a, b] \to \mathbb{R}^2$ using the monotonicity (left) and gradient (right) methods.

Convex-Hull Computation With Convex Projections

Ideas similar to those for monotonicity can also be used when the component functions f^i of f are convex or concave on some convex polyhedral sets $\mathcal{C}_l$ that cover the set $\mathcal{C}$. Note that when f^i is convex (concave), the minimum (maximum) of f^i on each $\mathcal{C}_l$ can be found by convex programming and the maximum (minimum) can be found by evaluating the function f^i on the vertices of $\mathcal{C}_l$. Such a strategy again creates a collection of hyper rectangles that over-approximate $f(\mathcal{C})$. However, the convex hull of these hyper rectangles can potentially have a large number of vertices. Next, attention is restricted to a very special case with a single parameter, valid for the lane keeping application studied later, for which over-approximations with smaller numbers of vertices can be computed.

Consider the special case of a map $f = (f^0, f^1) : [\underline{x}, \overline{x}] \to \mathbb{R}^2$ where f^1 is convex in f^0 over $[\underline{x}, \overline{x}]$, i.e., the function $f^1 \circ (f^0)^{-1} : f^0([\underline{x}, \overline{x}]) \to \mathbb{R}$ is a convex function. In this case, the line connecting $f(\underline{x})$ to $f(\overline{x})$ lies completely above the graph of $f([\underline{x}, \overline{x}])$. Furthermore, any tangent line to $f([\underline{x}, \overline{x}])$ lies completely below the graph of $f([\underline{x}, \overline{x}])$. The right part of Figure 3.4 illustrates the idea, where one hyperplane constraint is constructed using the former fact, and three hyperplane constraints are constructed using the latter fact. In particular, the former constraint is of the form

$$-mf^0(x) + f^1(x) \leq f^1(\underline{x}) - mf^0(\underline{x}), \quad m = \frac{f^1(\overline{x}) - f^1(\underline{x})}{f^0(\overline{x}) - f^0(\underline{x})},$$

and the latter constraints are of the form

$$m_0 f^0(x) + m_1 f^1(x) \geq m_0 f^0(x_i) + m_1 f^1(x_i),$$

where $m_0 \frac{\mathrm{d}f^0}{\mathrm{d}x}(x_i) + m_1 \frac{\mathrm{d}f^1}{\mathrm{d}x}(x_i) = 0$ and $x_i \in [\underline{x}, \overline{x}]$.

This method can also be applied in higher dimensions, i.e., when $f : [\underline{x}, \overline{x}] \to \mathbb{R}^q$, provided that the components $f^0, \ldots, f^{q-1}$ of f pairwise satisfy the convexity property described above. In this case, two-dimensional sets $\mathcal{P}_{ij}$ can be computed for all i, j with $i \neq j$, and then lifted to q dimensions to obtain $\mathrm{conv}(f(\mathcal{C})) \subset \mathcal{P} = \{[x^0, \ldots, x^{q-1}]^\intercal : (x^i, x^j) \in \mathcal{P}_{ij} \; \forall \; i \neq j\}$.

3.2.2. Backwards Reachable Set for Systems With Measurable and Non-Measurable Disturbance

Assuming that the convexification and over-approximation steps in Section 3.2.1 have been carried out, the effect of each subsystem on other subsystems is represented by a (finite) convex polyhedron. To somewhat simplify notation, consider the viewpoint of a single subsystem; the effects on this subsystem from other subsystems take the form either of measurable disturbance or non-measurable disturbance, depending on information availability. Collect these effects in variables $\mathbf{d}^1$ and $\mathbf{d}^2$ and consider dynamics Σ_{DT} defined as

$$\Sigma_{DT} : \; \mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{d}^1(k), \mathbf{u}(k), \mathbf{d}^2(k)),$$

for

$$f(x, d^1, u, d^2) = \left(A + \sum_{j \in [n_{d^1}]} d^1_j A^1_j + \sum_{j \in [n_{d^2}]} d^2_j A^2_j \right) x + Bu + \sum_{j \in [n_{d^1}]} d^1_j F^1_j + \sum_{j \in [n_{d^2}]} d^2_j F^2_j,$$

together with sets $\mathcal{D}_i = \{d^i : H_{d^i} d^i \leq h_{d^i}\}$ and $\mathcal{U} = \{u : H_u u \leq h_u\}$ that describe disturbance and input constraints, for $H_x \in \mathbb{R}^{\mathcal{N}_x \times n_x}, H_{d^i} \in \mathbb{R}^{\mathcal{N}_{d^i} \times n_{d^i}}$, and $H_u \in \mathbb{R}^{\mathcal{N}_u \times n_u}$. Here $\mathbf{d}^1$ models measurable disturbance and $\mathbf{d}^2$ non-measurable disturbance; that is, the input $\mathbf{u}(k)$ can depend on $\mathbf{d}^1(k)$ but not on $\mathbf{d}^2(k)$. Note that (3.17) can be written in this form when $\mathrm{conv}(f_{nm}(\mathcal{C}_m))$ is a set defined by linear inequalities.

The goal of this subsection is to compute the backwards reachable set

$$\mathrm{Pre}^{\Sigma_{DT}}(X) = \left\{x : \forall d^1 \in \mathcal{D}^1 \; \exists u \in \mathcal{U} \; \forall d^2 \in \mathcal{D}^2, \; f(x, d^1, u, d^2) \in X\right\}$$

for a set $X = \{x : H_x x \leq h_x\}$.

An exact way to compute $\mathrm{Pre}^{\Sigma_{DT}}$ is provided in the following result:

Theorem 3.4.

$$\mathrm{Pre}^{\Sigma_{DT}}(X) = \bigcap_{d^1 \in V(\mathcal{D}^1)} \mathrm{proj}_u \left\{(x, u) : H_x f(x, d^1, u, d^2) \leq h_x, H_u u \leq h_u \;\; \forall d^2 \in V(\mathcal{D}^2)\right\},$$

where $V(\mathcal{D}^i)$ is the (finite) set of vertices of $\mathcal{D}^i$.

Proof. Follows immediately from quantifier elimination via intersection ($\forall$) and projection ($\exists$). □

Thus, $\mathrm{Pre}^{\Sigma_{DT}}(X)$ can be computed via $|V(\mathcal{D}^1)|$ projections of $(n_x + n_u)$-dimensional polyhedra, each with $\mathcal{N}_x|V(\mathcal{D}^2)| + \mathcal{N}_u$ inequalities. The sets of vertices $V(\mathcal{D}^i)$ is finite but can potentially be very large—just the problem of enumerating the vertex set can be computationally challenging. An alternative way of computing an inner approximation of $\mathrm{Pre}^{\Sigma_{DT}}$ that does not require vertex enumeration is therefore presented next.

Theorem 3.5. *The following set is an inner approximation of* $\mathrm{Pre}^{\Sigma_{DT}}(X)$*:*

$$\left\{\begin{array}{l}
x : \exists y_0 \in \mathbb{R}^{\mathcal{N}_{d^2} \times \mathcal{N}_x}, \{K_y^j\}_{j \in [n_{d^1}]} \in \mathbb{R}^{\mathcal{N}_{d^2} \times \mathcal{N}_x}, u_0 \in \mathbb{R}^{n_u}, K_u \in \mathbb{R}^{n_u \times n_{d^1}} \\
\quad z_1, \{z_{2,i}\}_{i \in [n_{d^2}]}, \{z_{3,i}\}_{i \in [n_{d^2}]}, \{z_{4,i}\}_{i \in [\mathcal{N}_{d^2}]} \in \mathbb{R}^{\mathcal{N}_{d^1} \times \mathcal{N}_x}, z_5 \in \mathbb{R}^{\mathcal{N}_{d^1} \times \mathcal{N}_u}, \; s.t. \\
\quad h_{d^1}^T z_1 \leq (h_x - H_x A x - H_x B u_0 - y_0^T h_{d^2})^T, \; h_{d^1}^T z_{4,i} \leq e_i^T y_0 \\
\quad H_{d^1}^T z_1 = \left(\left[(K_y^0)^T h_{d^2} \quad \cdots \quad (K_y^{n_{d^1}-1})^T h_{d^2}\right] + H_A^1 (I_{n_d^1} \otimes x) + H_F^1 + H_x B K_u\right)^T, \\
\quad h_{d^1}^T z_{2,i} \leq e_i^T (H_A^2 (I_{n_d^2} \otimes x) + H_F^2 - y_0^T H_{d^2})^T, \\
\quad h_{d^1}^T z_{3,i} \leq -e_i^T (H_A^2 (I_{n_d^2} \otimes x) + H_F^2 - y_0^T H_{d^2})^T, \\
\quad H_{d^1}^T z_{2,i} = \begin{bmatrix} e_i^T H_{d^2}^T K_y^0 \\ \vdots \\ e_i^T H_{d^2}^T K_y^{n_{d^1}-1} \end{bmatrix}, \; H_{d^1}^T z_{3,i} = -\begin{bmatrix} e_i^T H_{d^2}^T K_y^0 \\ \vdots \\ e_i^T H_{d^2}^T K_y^{n_{d^1}-1} \end{bmatrix}, \; H_{d^1}^T z_{4,i} = \begin{bmatrix} -e_i^T K_y^0 \\ \vdots \\ -e_i^T K_y^{n_{d^1}-1} \end{bmatrix} \\
\quad h_{d^1}^T z_5 \leq (h_u - H_u u_0)^T, \; H_{d^1}^T z_5 = K_u^T H_u^T, \\
\quad z_1 \geq 0, \; z_{2,i} \geq 0, \; z_{3,i} \geq 0, \; z_{4,i} \geq 0, \; z_5 \geq 0.
\end{array}\right\}$$

where for $i = 1, 2$*;*

$$H_A^i = \begin{bmatrix} H_x A_0^i & \cdots & H_x A_{n_d^i-1}^i \end{bmatrix}, \quad \textit{and} \quad H_F^i = \begin{bmatrix} H_x F_0^i & \cdots & H_x F_{n_d^i-1}^i \end{bmatrix}, \tag{3.21}$$

and $\otimes$ *is the Kronecker product.*

Similarly to Theorem 3.4, this set is the projection of a polyhedron. The approximation in the result above stems from restricting $\mathbf{u}(k)$ to be linearly dependent on $\mathbf{d}^1(k)$, and a similar restriction for a slack variable y. The number of variables in this implicit set description is

$$n_{tot} = n_x + n_u + n_u n_{d^1} + \mathcal{N}_x \mathcal{N}_{d^2}(1 + n_{d^1}) + \mathcal{N}_x \mathcal{N}_{d^1}(1 + 2n_{d^2} + \mathcal{N}_{d^2}) + \mathcal{N}_u \mathcal{N}_{d^1}, \tag{3.22}$$

and the number of equalities and inequalities are

$$\mathcal{N}_{eq} = n_{d^1} \mathcal{N}_x (1 + 3n_{d^2}) + n_{d^1} \mathcal{N}_u \quad \text{and} \quad \mathcal{N}_{iq} = \mathcal{N}_x (1 + 3n_{d^2}) + \mathcal{N}_u, \tag{3.23}$$

respectively. As a consequence of Theorem 3.5 the backwards reachable set can be inner approximated via a single projection of a n_{tot}-dimensional polyhedron onto its first n_x dimensions. While these numbers seem large, the principal culprits are cross terms that disappear if only one type of disturbance (either measurable or non-measurable) is present. A proof is given in Appendix B.2.

3.2.3. Contract refinement

The computational procedures above can be employed to evaluate whether a contract of the form

$$\bigwedge_{\substack{m \in [N] \\ m \neq n}} \square \left(\mathbf{x}_m \in \mathcal{X}_m \right) \implies \square \left(\mathbf{x}_n \in \mathcal{X}_n \right)$$

can be upheld, by computing controlled invariant sets $\mathcal{C}_n \subset \mathcal{X}_n$. In case this is not possible, i.e. one or more of the sets $\mathcal{C}_n$ turn out as the empty set, the original contract needs to be refined. Based on the observation that it is easier for subsystem n to enforce invariance of $\mathcal{X}_n$ if $\mathcal{X}_m$ is smaller; the following heuristics can be applied for contract refinement:

1. For all n such that $\mathcal{C}_n \neq \emptyset$, set $\mathcal{X}_n = \mathcal{C}_n$.

2. If $\mathcal{C}_n = \emptyset$ or step 1. did not result in additional nonempty sets, shrink all $\mathcal{X}_n$ for which $\mathcal{C}_n \neq 0$ by a predefined factor.

The computations of $\mathcal{C}_n$ can be guaranteed to terminate by adding a pre-defined shrinking factor [34, 112].

3.3. Examples

Below two numerical examples are provided. The first example demonstrates how separable invariant sets synthesized via the LMIs in Section 3.1 can be used to decouple a synthesis problem. Secondly, decoupled synthesis as in Section 3.2 is considered for Adaptive Cruise Control and Lane Keeping, for a model that exhibits nonlinear interconnections.

3.3.1. Synthesis of Separable Invariant Sets for Modular Control Design

Consider a scenario involving a tethered UAV and a ground vehicle, where each vehicle is given a surveillance task that requires visiting certain regions infinitely often. The tether can be used to power the UAV to significantly increase the duration it is airborne, however it induces dynamic coupling between the UAV and the ground vehicle. This coupling is modeled as a spring. For simplicity, the vehicles are modeled as double integrators and their motion is constrained to one dimension:

$$\begin{bmatrix} \mathbf{x}^1(k+1) \\ \mathbf{v}^1(k+1) \\ \mathbf{x}^2(k+1) \\ \mathbf{v}^2(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ \gamma & 1 & -\gamma & 0 \\ 0 & 0 & 1 & 1 \\ -\gamma & 0 & \gamma & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}^1(k) \\ \mathbf{v}^1(k) \\ \mathbf{x}^2(k) \\ \mathbf{v}^2(k) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u}^1(k) \\ \mathbf{u}^2(k) \end{bmatrix}. \tag{3.24}$$

The objective is to find separable invariant sets for the two subsystems $(\mathbf{x}^1, \mathbf{v}^1)$ and $(\mathbf{x}^2, \mathbf{v}^2)$ and then let each system perform additional control objectives while still guaranteeing overall invariance. Using a small coupling $\gamma = 0.1$ and a bound $\|\mathbf{u}(k)\|_\infty \leq 0.3$, it takes 1.7s to compute the invariant sets depicted in Figure 3.5 that consist of 5 zonotope generators each.

Next additional control objectives are pursued inside these sets in a modular fashion. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be the invariant set pertaining to the subsystem $(\mathbf{x}^1, \mathbf{v}^1)$ and $(\mathbf{x}^2, \mathbf{v}^2)$ respectively.

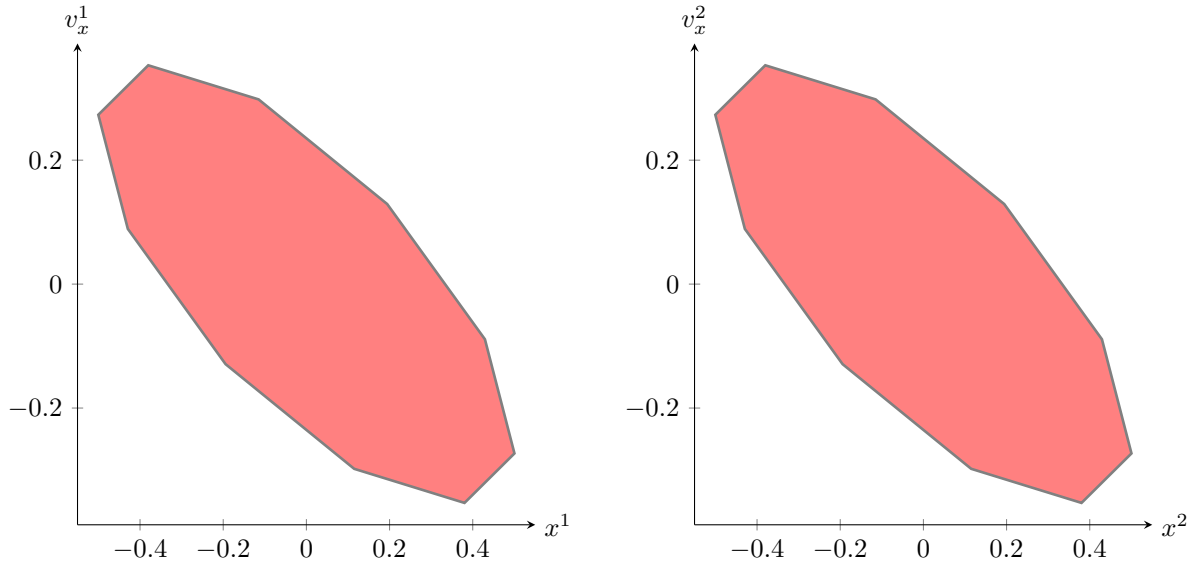

Figure 3.5: Invariant sets for toy robot/UAV example.

Consider a surveillance-like task where the ground robot is required to visit the goal sets $G_1^+, G_1^- \subset \mathcal{C}_1$ infinitely often; whereas the UAV is required to visit $G_2^+, G_2^- \subset \mathcal{C}_2$ infinitely often, where the goal sets are taken to be $G_1^{\pm} = \{(\pm x^1, v^1) : x^1 \in [0.2, 0.35]\}$ and $G_2^{\pm} = \{(\pm x^2, v^2) : x^2 \in [0.05, 0.18]\}$. The local objectives can be expressed in LTL as $\varphi_i = \square\lozenge G_i^+ \wedge \square\lozenge G_i^-$ for $i = 1, 2$.

Synthesis for the local objectives is done via fixed-point calculations (1.15) to synthesize the local controllers, but any LTL synthesis method could potentially be employed. Figure 3.6 depicts $G_i^{\pm}$, together with robust (with respect to "disturbance" induced from the other subsystem) backwards reachable sets of G_i^- contained inside $\mathcal{C}_i$ in lighter green (the backwards reachable sets from G_i^+ are symmetric). Since the sets from where G_i^- is reachable eventually cover G_i^+, together with the fact that G_i^+ is reachable from G_i^- by symmetry, the specification is realizable. The left illustration in Figure 3.7 shows trajectories of a simulation where both systems satisfy their control objectives, while simultaneously countering the "disturbance" they cause each other.

Now, assume the task specification for the UAV changes to $\varphi_2' = \square\lozenge G_2' \wedge \square\lozenge G_2''$, with $G_2' = \{(x^2, v^2) : x^2 \in [-0.18, -0.05]\}$ and $G_2'' = \{(x^2, v^2) : x^2 \in [0.18, 0.33]\}$. By construction of the invariant sets, if a new controller can be synthesized for the UAV, the new specification $\varphi_1 \wedge \varphi_2'$ is guaranteed to be satisfied without making any changes to the controller of the ground robot, despite the potentially different "disturbance" inputs it gets

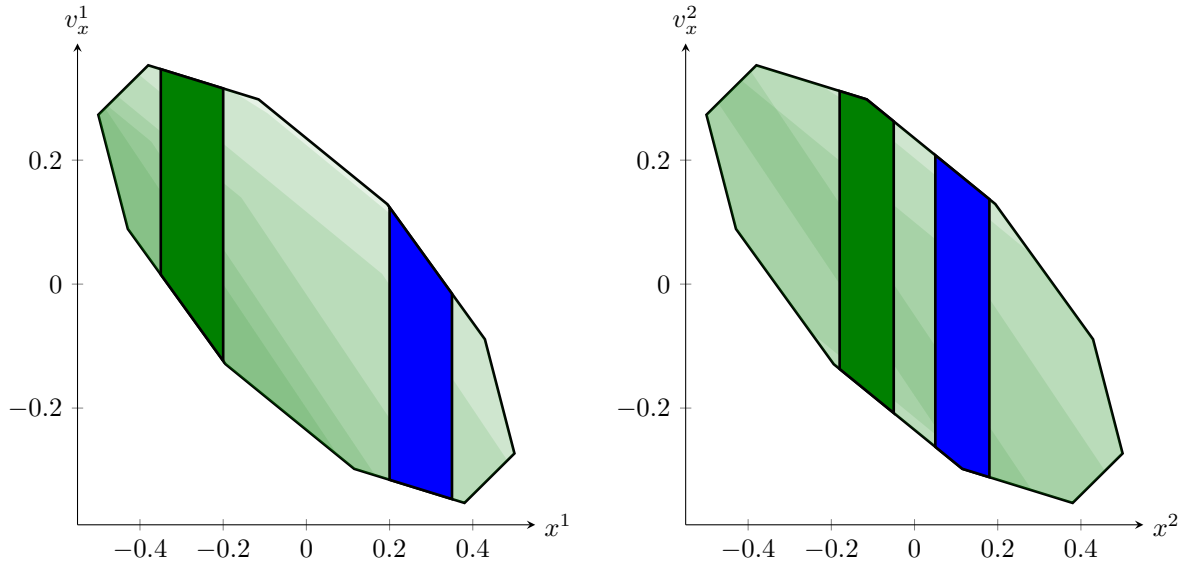

Figure 3.6: Synthesis inside separable invariant sets. For $i = 1$ (left) and $i = 2$ (right), the goal sets G_i^-, G_i^+ are shown in green, blue. The light green sets depict regions from where G_i^- is reachable.

from the UAV. The right illustration in Figure 3.7 shows the trajectories of a simulation of this new scenario where both subsystems satisfy their control objectives.

An interesting feature of this problem is that if the invariant sets are increased in size, these control objectives can no longer be realized using this kind of decentralized controllers. As the sizes of the sets increase, more control effort must be reserved for countering the increasing "disturbance" from the other subsystem and hence there is less freedom to pursue additional control objectives.

3.3.2. Composition of Lane Keeping and Adaptive Cruise Control

In this section, the method presented in Section 3.2 is applied to synthesize a Lane Keeping (LK) controller and an Adaptive Cruise Controller (ACC) whose composition is guaranteed to be safe. Consider the following models for longitudinal motion

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{u}\\ \mathbf{h}\end{bmatrix} = \begin{bmatrix}-f_1/m & 0\\ -1 & 0\end{bmatrix}\begin{bmatrix}\mathbf{u}\\ \mathbf{h}\end{bmatrix} + \begin{bmatrix}\mathbf{F_w}/m - f_0/m - \boldsymbol{\nu}\mathbf{r}\\ v_L\end{bmatrix}; \tag{3.25}$$

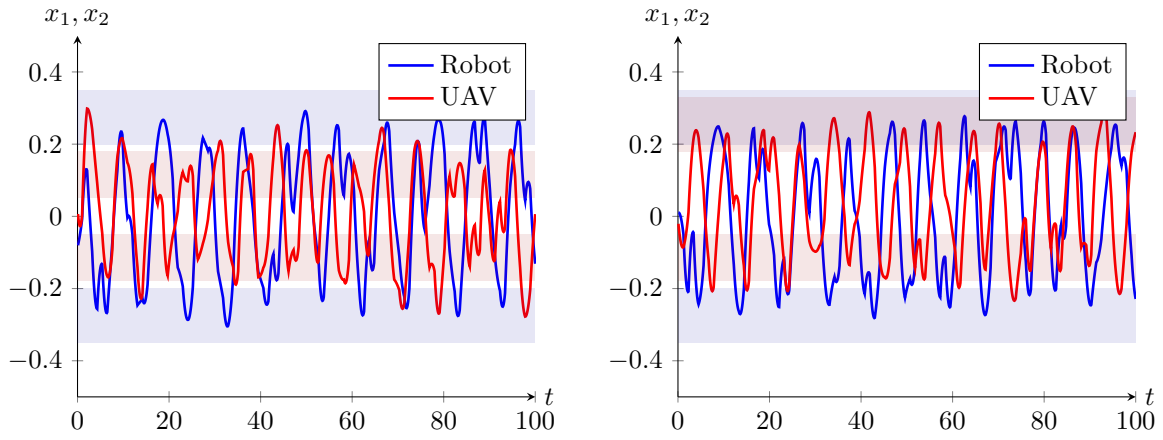

Figure 3.7: Re-synthesis inside separable invariant sets. Subsystem trajectories that visit given regions (marked with red, blue) infinitely often. Re-synthesis can be done separately for the different subsystems thanks to the inherent robust modularity. The two plots show simulation for different local control objectives.

and for lateral dynamics

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\mathbf{y}\\ \boldsymbol{\nu}\\ \boldsymbol{\Delta\Psi}\\ \mathbf{r}\end{bmatrix} = \underbrace{\begin{bmatrix}0 & 1 & u & 0\\ 0 & -\frac{C_{\alpha f}+C_{\alpha r}}{m\mathbf{u}} & 0 & \frac{bC_{\alpha r}-aC_{\alpha f}}{m\mathbf{u}}-\mathbf{u}\\ 0 & 0 & 0 & 1\\ 0 & \frac{bC_{\alpha r}-aC_{\alpha f}}{I_z\mathbf{u}} & 0 & -\frac{a^2C_{\alpha f}+b^2C_{\alpha r}}{I_z\mathbf{u}}\end{bmatrix}}_{A_{LK}}\begin{bmatrix}\mathbf{y}\\ \boldsymbol{\nu}\\ \boldsymbol{\Delta\Psi}\\ \mathbf{r}\end{bmatrix} + \begin{bmatrix}0\\ \frac{C_{\alpha f}}{m}\\ 0\\ a\frac{C_{\alpha f}}{I_z}\end{bmatrix}\boldsymbol{\delta_f} + \begin{bmatrix}0\\ 0\\ -1\\ 0\end{bmatrix}\mathbf{r_d}. \quad (3.26)$$

The goal of the ACC is to compute the applied longitudinal force $\mathbf{F_w}$ so that either a desired longitudinal speed $\mathbf{u} = u_{des}$ is achieved, or so that the headway $\mathbf{h}$ stays above some minimal value, where the latter objective is higher prioritized. Correspondingly, the goal of LK is to control the steering angle $\boldsymbol{\delta_f}$ in a way that prevents lane departure.

Both these models are linearized versions of nonlinear force-balance equations. In the following the physical meaning of each state is briefly described; see [86] and [111] for details. For the ACC model, the two states $\mathbf{u}$ and $\mathbf{h}$ represent longitudinal velocity and headway. The headway is the distance to a lead car, which is assumed to travel at velocity v_L. The LK model has four states $\mathbf{y}, \boldsymbol{\nu}, \boldsymbol{\Delta\Psi}$ and $\mathbf{r}$, which describe lateral displacement from the center of the lane, lateral velocity, yaw angle, and yaw rate. The input to the LK system is the steering angle $\boldsymbol{\delta_f}$, and there is also an exogenous disturbance $\mathbf{r_d}$ coming from the curvature of the road. The road curvature is assumed to be bounded by $|\mathbf{r_d}| \leq 0.05$,

Table 3.1: Parameter values for ACC and LK models.

m	1650 kg	f_0	-24 N	f_1	19 Ns/m
a	1.11 m	$C_{\alpha f}$	133000 N	I_z	2315 kg m^2
b	1.59 m	$C_{\alpha r}$	98800 N	v_L	26 m/s

Table 3.2: State constraints for ACC and LK models.

State	Lower bound	Upper bound	State	Lower bound	Upper bound
$\mathbf{u}$	25 m/s	30 m/s	$\boldsymbol{\nu}$	-1.2 m/s	1.2 m/s
$\mathbf{h}$	42 m	∞	$\boldsymbol{\Delta\Psi}$	-0.05 rad	0.05 rad
$\mathbf{y}$	-0.9 m	0.9 m	$\mathbf{r}$	-0.3 rad/s	0.3 rad/s

which is in line with the maximal recommended curvature of freeways in Michigan [109].

As can be seen, the state $\mathbf{u}$ pertaining to the ACC system appears in the system matrix of (3.26). Conversely, the states $\boldsymbol{\nu}$ and $\mathbf{r}$ from the LK system appear in the offset term of the ACC system. Furthermore, both of these interdependencies are nonlinear.

In the following, the parameter values given in Table 3.1 are assumed and for comfort reasons control input bounds of $\mathbf{F_w} \in [-3m, 2m]$ and $\boldsymbol{\delta_f} \in [\pi/90, \pi/90]$ are imposed. For combined safety and comfort reasons, the state constraints given in Table 3.2 are posited. The bound $\mathbf{y} \in [-0.9, 0.9]$ prevents lane departure, while the bound $\mathbf{h} \geq 42$ implies a time headway[20] of 1.4 s at $\mathbf{u} = 25$ m/s, which is a common recommendation.

If a LK controller can be found that keeps the states $\mathbf{y}, \boldsymbol{\nu}, \boldsymbol{\Delta\Psi}, \mathbf{r}$ within the bounds in Table 3.2 whenever $\mathbf{u}$ is within its bounds, and conversely for the ACC system, safety is guaranteed. To apply the methods from Section 3.2 the continuous-time models are time-discretized using Euler forward with a time step of 0.1s, which preserves the base $\{f^i\}_{i=1}^q$ from which the system matrices are composed. When implementing a correct discrete-time controller in continuous time, inter-sample constraint violations may occur, as well as differences from the omission of higher-order terms in the Euler forward approximation (as opposed to the exact exponential map). It is possible to correct for this by adding certain robustness margins [47, 75], but those details are omitted here.

Solving the LK part The entries of A_{LK} in (3.26) are composed of two linearly independent nonlinearities: u and $1/u$. Using the method in Section 3.2.1 yields the polyhedron

[20]Time headway, or *time to collision*, is defined as $\tau = \mathbf{h}/\mathbf{u}$.

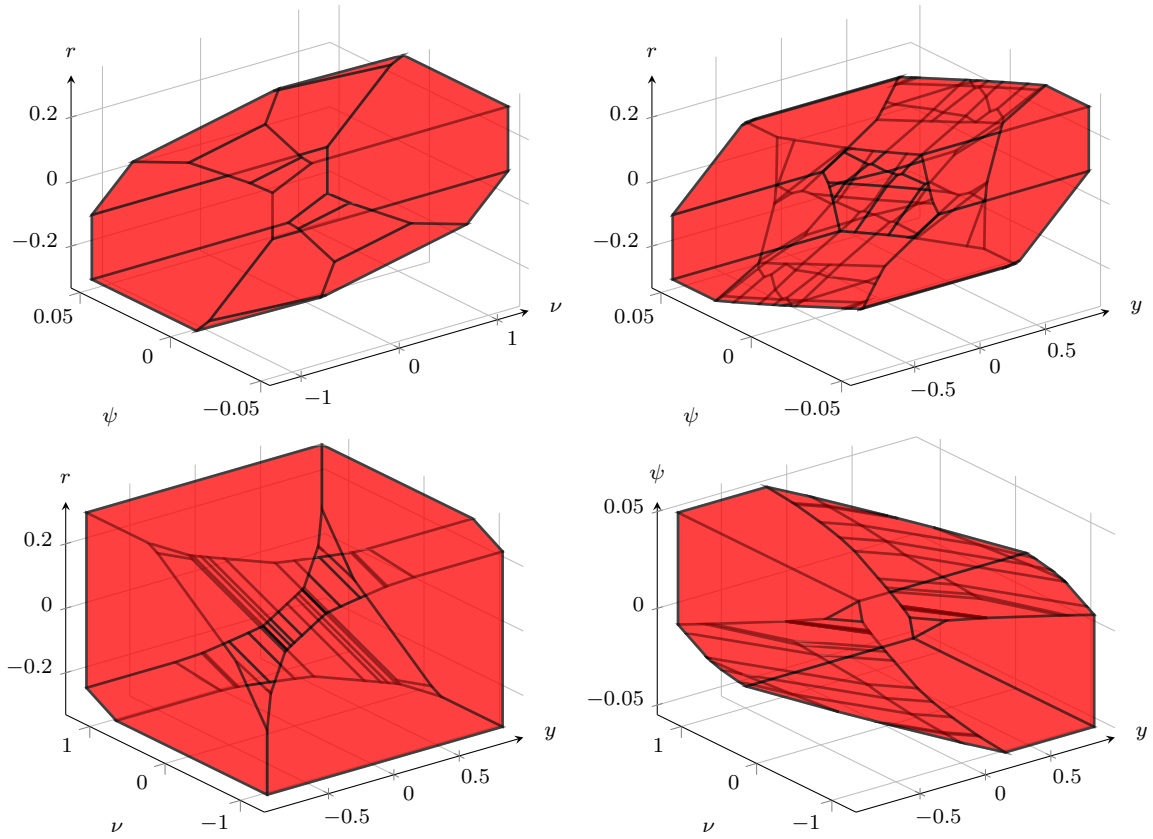

Figure 3.8: Projections of the four-dimensional controlled invariant set for the LK system.

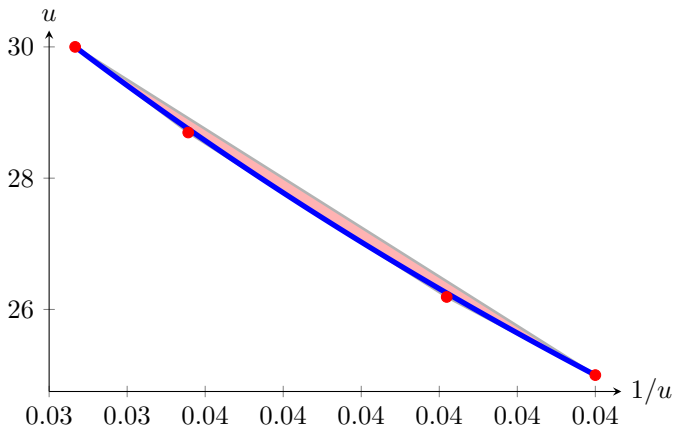

Figure 3.9: Over-approximation of nonlinearities in ACC subsystem. All possible values of $[1/u, u]$ (in blue) are covered by a polyhedron $\mathcal{P}_{LK}$ with four vertices (in red).

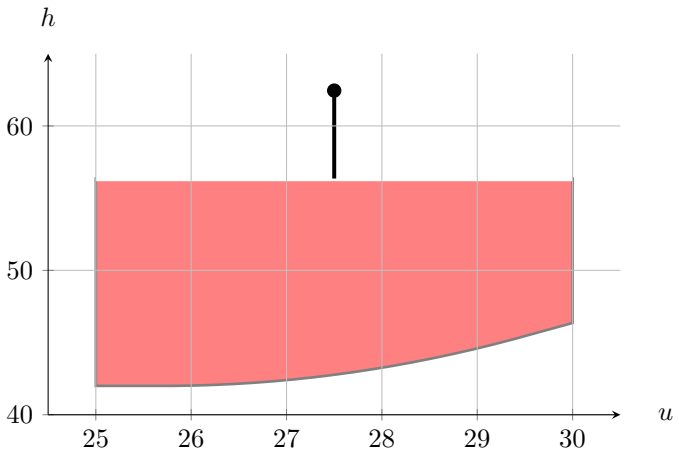

Figure 3.10: Controlled invariant set for the ACC subsystem.

$\mathcal{P}_{LK} \subset \mathbb{R}^2$ depicted in Figure 3.9 that represents an over-approximation of all values the function $u \mapsto [u, 1/u]$ can take when u varies in the interval $[25, 30]$. Applying the invariant set computation for the four vertex systems using the bounds for the LK states as the initial set gives the 4-dimensional polyhedron $\mathcal{C}_{LK}$ depicted in Figure 3.8, which is controlled invariant for all $u \in [25, 30]$.

Solving the ACC part In the ACC system (3.25), there is one offset term $\boldsymbol{\nu}\mathbf{r}$ that is a function of the state of the LK system. The goal is to find a polyhedron $\mathcal{P}_{ACC} \subset \mathbb{R}$ that is an over-approximation of the range of the function $f : [\nu, r] \mapsto \nu r$ where $\nu \in [-1.2, 1.2]$ and $r \in [-0.3, 0.3]$. The exact range of this function is easily seen to be $[-0.36, 0.36]$.

Once the two sets $\mathcal{C}_{ACC}$ and $\mathcal{C}_{LK}$ that satisfy (3.5) are obtained, controllers that guarantee invariance can be implemented by choosing for a given state $\mathbf{x}_{ACC} \in \mathcal{C}_{ACC}$ and $\mathbf{x}_{LK} \in \mathcal{C}_{LK}$ an input $\mathbf{u}_{ACC}$ such that $A(\mathbf{x}_{LK})\mathbf{x}_{ACC} + B\mathbf{u}_{ACC} + F(\mathbf{u}_{LK}) \in \mathcal{C}_{ACC}$, and conversely for $\mathbf{x}_{LK}$. Finding such inputs amounts to enforcing certain linear constraints which by construction are feasible. In particular, at each time step an input satisfying these constraints is computed by solving a convex quadratic optimization problem. In Figure 3.11 a simultaneous simulation of the ACC and LK systems is shown, where the road curvature $\mathbf{r_d}$ is in the form of a sinusoidal signal with maximal amplitude 0.05. As expected, all bounds are respected throughout the simulation.

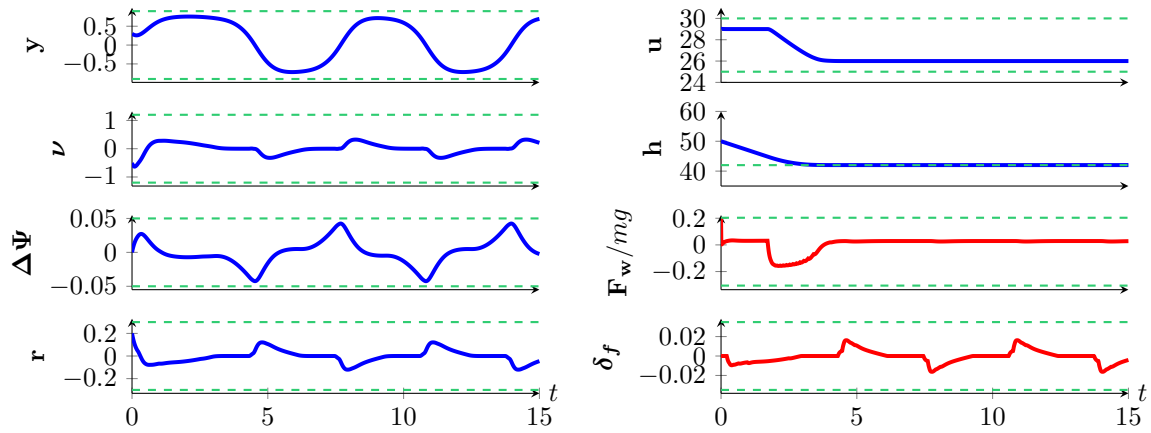

Figure 3.11: Simultaneous simulation of the LK and ACC subsystems on a curvy road. States are shown in blue, and computed control inputs in red. The dashed green lines indicate state and control bounds that are guaranteed to hold. The ACC controller is programmed to maintain a desired speed of 29 m/s when possible. However, it is forced to slow down at $t = 2$ to avoid future violation of the headway constraint.

3.4. Conclusions

This chapter was concerned with computation of separable controlled invariant sets, or, in other words: assume-guarantee contracts for invariance. The formal relationships between the subsystems exhibited by such sets allow decomposition of formal synthesis problems as shown in the example in Section 3.3.1. Two methods for computation of such sets were proposed; one based on LMIs, and one obtained from a generalization of the iterative fixed-point computation in Section 1.3.1. The latter method can treat nonlinear interconnections via convex over-approximations of function images.

While the dimensionality is reduced, there are still numerical challenges due to polyhedra that become overly complex. The toughest numerical barrier in linear reachability is polyhedral projection. The standard algorithm is to employ Fourier Motzkin elimination that results in a highly redundant description. Unfortunately, the best known algorithm [26] for redundancy elimination still lacks a reliable implementation and thus presents an easy way to at least marginally improve scalability. To further improve scalability of the iterative method it would be desirable to control the complexity of the involved polyhedra; for instance by working with inner approximations. Another potential direction for improvement is to avoid the projection step and work with implicit set representations, since it is well known that the projection $\text{proj}(\mathcal{P})$ of a polyhedron $\mathcal{P}$ may have exponen-

tially many more inequalities than $\mathcal{P}$ [23]. While implicit representations may facilitate the synthesis phase, additional on-line computational burden is introduced—checking set membership requires solving a LP feasibility problem. The key technical challenge is to find a termination criterion for the iterative algorithms, either by pre-computing an upper bound on the number of iterations, or by checking inclusion of implicitly defined sets.

At a higher level, also the contract refinement heuristics (Section 3.2.3) can potentially be strengthened with theoretical guarantees. The refinement procedure seeks to resolve the inherent trade-off: a larger set for subsystem n makes the n'th synthesis problem easier, but simultaneously makes it more difficult for other subsystems. Motivated by [32], an instrumental ingredient could be computation of minimally restrictive assumptions, i.e., the maximal disturbance one subsystem can tolerate from other subsystems.

Chapter 4.

Exploiting Symmetry in Counting Problems

In this chapter a different approach to tackle high-dimensionality is explored: exploitation of symmetries. The system under consideration is the aggregation of N almost-homogeneous n_x-dimensional switched subsystems—making the aggregate system a $N \times n_x$-dimensional switched system with 2^N aggregate modes. Though N can be very large, the system exhibits symmetries in the dynamics which can be exploited to enable synthesis for a special type of symmetric constraints called *counting constraints*. Like the previous chapter, the solution approach is abstraction-based, but with the crucial difference that a single abstraction is used for all subsystems, thus roughly reducing the number of states in the abstraction from $\mathcal{O}(1/\eta^{Nn_x})$ in the naïve approach to $\mathcal{O}(1/\eta^{n_x})$, where η the precision. Mild heterogeneity among the subsystems can be accounted for by assuming certain stability conditions. The idea of utilizing a single abstraction for multiple subsystems allows synthesis problems of very high dimension that exhibit these symmetries to be solved—the method is demonstrated on a $10,000$-dimensional switched system with $2^{10,000}$ modes. If there are several distinct classes of subsystems (i.e., strong heterogeneity), one abstraction can be constructed for each class; thus abstracting not only the state space but also the space of system configurations as displayed in Figure 4.1.

This work is motivated by the TCL coordination problem. TCLs include air conditioners, water heaters, refrigerators, etc., that operate around a temperature set point. TCL owners are typically indifferent to small temperature perturbations and accept temperatures in a range around the set point; this range is called the *dead band*. The idea behind TCL coordination is that an electric utility company can leverage this flexibility–which becomes

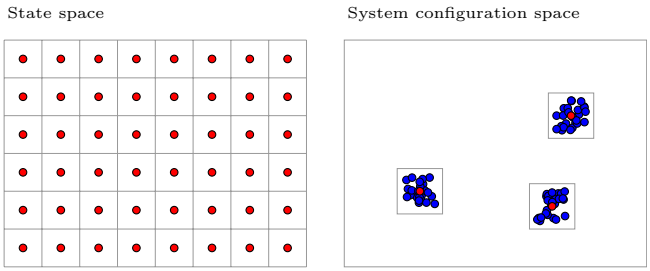

Figure 4.1: Abstraction in state space and in system configuration space. In the state space to the left, a discrete point is selected to represent all continuous states in the same cell. Similarly, to the right, three representative systems (in red) have been chosen to represent the three classes of mildly heterogeneous systems.

meaningful for large collections of TCLs–to shape aggregate demand on the grid.

The fundamental problem in TCL coordination is to simultaneously control local safety constraints (i.e., maintain each TCL in its deadband), and global aggregate constraints. These constraints are special instances of *counting constraints*, which is the type of constraint considered in this chapter. The TCL coordination example will be used throughout the chapter to motivate the discussion:

Example 4.1. *Let* $\{\mathbf{x}_n\}_{n\in[N]}$ *be the states of a family of* N *switched systems of the following form:*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_n(t) = f_{\sigma_n(t)}\left(\mathbf{x}_n(t), \mathbf{d}_n(t)\right), \quad \sigma_n(t) \in \{\texttt{on}, \texttt{off}\}, \tag{4.1}$$

where $f_{\texttt{on}}$ *and* $f_{\texttt{off}}$ *are functions* $\mathbb{R} \times \mathcal{D} \to \mathbb{R}$ *for some bounded disturbance set* $\mathcal{D}$*. For a desired dead band* $[\underline{a}, \overline{a}]$*, and bounds* $[\underline{K}, \overline{K}]$ *on the aggregate number of TCLs that are in mode* **on***, synthesize switching protocols* $\{\pi_n\}_{n\in[N]}$ *such that the generated trajectories satisfy*

$$\mathbf{x}_n(t) \in [\underline{a}, \overline{a}] \text{ for all } n \in [N] \text{ and all } t \in \mathbb{R}_+, \tag{4.2}$$

$$\text{For all } t \in \mathbb{R}_+, \textbf{ at least } \underline{K} \text{ and } \textbf{at most } \overline{K} \text{ of the TCLs are in mode } \texttt{on}. \tag{4.3}$$

Chapter overview. The general counting problem is introduced in Section 4.1 along with an abstraction procedure that transforms a continuous-state counting problem into a finite-state one. A solution approach to the finite-state counting problem is presented in Section 4.2 in the form of a linear feasibility problem that can either be solved as a LP, or as an ILP. Results in Section 4.3 specify what inferences that can be made about the

finite-time counting problem from (in)-feasibility of different configurations of the proposed feasibility problem, and suggest rounding procedures to convert non-integer solutions into integer solutions. Extension of the theory to the multi-class setting is briefly presented in Section 4.4. To highlight the applicability of the method, one numerical example and one application-motivated example are provided in Section 4.5 before the chapter is summarized in Section 4.6.

Related work. Previous work on the TCL scheduling problem has resulted in control algorithms based on broadcasting a universal set-point temperature [43], Markov chain "bin" models [30, 37], and priority stacks [53], with the objective to track a power signal. However, hard constraints on aggregate power consumption (i.e., the number of TCLs in mode `on`) has not been taken into account, which is crucial to avoid overloading the grid or under-utilizing the power generated by renewable resources.

A solution to a problem reminiscent of the TCL problem is presented in [6, 8], where LP-based schedule search for bounded-rate systems is considered. While that approach allows for more general set constraints and has polynomial complexity in the number of permitted aggregate modes, it still becomes impractical for large-scale systems since the number of aggregate modes grows exponentially. Also related is recent work on intersection clearing controllers [3, 27], where integer programs are used to find a finite-horizon schedule that allows cars to clear an intersection. This work can be seen as a marriage between the "bin" abstraction approach previously used in TCL literature, and schedule search via integer programming—generalized to arbitrary incrementally stable systems and an infinite horizon.

A particular instance of the (mode-)counting problem has been studied in more detail: the case with one-dimensional subsystems with two modes. It turns out that in this case an analytical solution is admitted for the infinite-horizon case even with strong heterogeneity [92]. In the case that the infinite-horizon problem is infeasible, the time of constraint satisfaction can be maximized [91].

4.1. The Counting Problem

Before introducing the counting problem, the concept of a counting constraint is defined.

Definition 4.1. *A **counting constraint** (X, R) for a collection of N switched subsystems*

numbered from 0 *to* $N-1$*, with the same state space* $\mathcal{X}$ *and input space* $\mathbb{U}$*, is a set* $X = X^{\mathcal{X}} \times X^{\mathbb{U}}$ *with* $X^{\mathcal{X}} \subset \mathcal{X}$ *and* $X^{\mathbb{U}} \subset \mathbb{U}$*, and a bound* R*. The counting constraint is* ***satisfied*** *by state-input pairs* (x_n, μ_n)*,* $n \in [N]$*, if the aggregate number of pairs that fall in* X *is less than or equal to* R*:*

$$\sum_{n\in[N]} \mathbb{1}_X(x_n, \mu_n) \leq R.$$

Remark 4.1. *For simplicity of exposition attention is restricted to counting sets that are the product of a set* $X^{\mathcal{X}}$ *in output space and a set* $X^{\mathbb{U}}$ *in mode space. However, the solution method can handle more general subsets of* $\mathcal{X} \times \mathbb{U}$*.*

This notion of counting constraint extends that in previous work [89] by allowing X to be a subset of $\mathcal{X} \times \mathbb{U}$, instead of being restricted to a singleton member of $\mathbb{U}$. This generalization permits counting constraints that include the state space, as opposed to only counting the number of systems using each mode (i.e., *mode*-counting). A local safety constraint $\mathbf{x}_n(t) \notin S$ for all $n \in [N]$ can be expressed with the counting constraint $(S \times \mathbb{U}, 0)$, therefore there is no need to treat local safety constraints separately as in the cited paper. Furthermore, if a lower bound is desired, a counting constraint for the complement X^C of X can be specified:

$$\sum_{n\in[N]} \mathbb{1}_X(x_n, \mu_n) \geq R \Longleftrightarrow \sum_{n\in[N]} \mathbb{1}_{X^C}(x_n, \mu_n) \leq N - R.$$

Example 4.2. *The TCL scheduling constraints* (4.2) *and* (4.3) *fall into the class of counting constraints. The following counting constraints specify that the number of TCLs that are in mode* `on` *at time* t *should be in the interval* $[\underline{K}, \overline{K}]$*:*

$$\sum_{n\in[N]} \mathbb{1}_{\mathbb{R}\times\{\texttt{on}\}}(\mathbf{x}_n(t), \sigma_n(t)) \leq \overline{K}, \quad \sum_{n\in[N]} \mathbb{1}_{\mathbb{R}\times\{\texttt{off}\}}(\mathbf{x}_n(t), \sigma_n(t)) \leq N - \underline{K}. \tag{4.4}$$

In addition, the dead band constraints (4.2) *can be imposed by the counting constraint*

$$\sum_{n\in[N]} \mathbb{1}_{[\underline{a},\overline{a}]^C\times\{\texttt{on},\texttt{off}\}}(\mathbf{x}_n(t), \sigma_n(t)) \leq 0. \tag{4.5}$$

Consider now a family of N identical subsystems with dynamics given by a transition system and some initial conditions $\{\mathbf{x}_n(0)\}_{n\in[N]}$. The problem considered in this chapter

is the following:

Problem 4.1. *Consider N subsystems, all governed by the same transition system $\mathcal{T} = (\mathcal{X}, \mathbb{U}, \longrightarrow, h)$, where $\mathbb{U}$ is finite, and assume that initial states $\{\mathbf{x}_n(0)\}_{n\in[N]}$ and L counting constraints $\{X_l, R_l\}_{l\in[L]}$ are given. Synthesize individual switching protocols $\{\pi_n\}_{n\in[N]}$ such that the generated actions $\sigma_n(0)\sigma_n(1)\sigma_n(2)\dots$ and trajectories $\mathbf{x}_n(0)\mathbf{x}_n(1)\mathbf{x}_n(2)\dots$ for $n \in [N]$ satisfy the counting constraints*

$$\sum_{n\in[N]} \mathbb{1}_{X_l}\left(\mathbf{x}_n(k), \mu_n(k)\right) \leq R_l, \quad \forall k \in \mathbb{Z}_+,\ \forall l \in [L]. \tag{4.6}$$

An instance of this problem is referred to as

$$(N, \mathcal{T}, \{\mathbf{x}_n(0)\}_{n\in[N]}, \{X_l, R_l\}_{l\in[L]}). \tag{4.7}$$

Naïvely, if $\mathcal{X}$ is a n_ξ-dimensional space, the overall system can be viewed as a $N \times n_\xi$-dimensional hybrid system with $|\mathbb{U}|^N$ modes and is thus beyond the reach of common control synthesis techniques that do not exploit the symmetry. However, symmetry is present: it does not matter which subsystems contribute to the summation in (4.6) for the constraint to be satisfied, so Problem 4.1 exhibits subsystem permutation symmetry in its specification. In addition, there is also subsystem permutation symmetry in the dynamics since all subsystem states are governed by the same transition system. As an implication there is no need to keep track of identities of individual subsystems—they are all equivalent from both a dynamics- as well as from a specification-point of view. This symmetry (i.e., permutation invariance) is the key to solving large-scale counting problems.

The goal of this work is to synthesize controllers that satisfy counting constraints for a collection of continuous-time switched systems with states $\mathbf{x}_n(t)$, $n \in [N]$, with dynamics governed by

$$\Sigma_{Sw}:\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_n(t) = f_{\sigma_n(t)}(\mathbf{x}_n(t), \mathbf{d}_n(t)),\ \sigma_n : \mathbb{R} \to \mathbb{U}, \tag{4.8}$$

for $\mathbf{x}_n(t) \in \mathbb{R}^{n_x}$ and a disturbance signal $\mathbf{d}_n(t)$ assumed to take values in a set $\mathcal{D}$. Although the system description is identical for all N systems in the family, the disturbance signals $\mathbf{d}_n$ are not, and can therefore model mild heterogeneity, including parameter variations across system models or modeling inaccuracies. The vector fields and the disturbance signals are assumed to satisfy standard assumptions for existence and uniqueness of solutions. In addition, it is assumed that the vector fields exhibit the following form of stability [10]:

Assumption 4.1. *For each $\mu \in \mathbb{U}$, the vector field $f_\mu(x, d)$ is C^1 in x and continuous in d. Furthermore, the nominal system $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = f_\mu(\mathbf{x}, \mathbf{0})$ is forward complete and incrementally stable. That is, there exists a $\mathcal{KL}$-function β_μ such that*

$$\|\phi_{f_\mu}(t, x; \mathbf{0}) - \phi_{f_\mu}(t, x'; \mathbf{0})\|_\infty \leq \beta_\mu \left(\|x - x'\|_\infty, t\right). \tag{4.9}$$

Note that the vector field $f_\mu(x, d)$ being C^1 in x implies that $f_\mu(x, d)$ is Lipschitz in x (with some Lipschitz constant K_μ) when the states are constrained to a compact set. Furthermore, assume that the disturbance signal is continuous, and that its effect is bounded in the following sense:

Assumption 4.2. *The disturbance signals $\mathbf{d}_n : \mathbb{R} \to \mathcal{D}$ are continuous. Furthermore for all $\mu \in \mathbb{U}$, compared to the nominal vector fields without disturbance, the effect of the disturbance is less than $\bar{\delta}_\mu$:*

$$\|f_\mu(x, d) - f_\mu(x, 0)\|_\infty \leq \bar{\delta}_\mu,$$

for all $d \in \mathcal{D}$.

The counting problem was introduced only for transition systems, which comprise time-sampled continuous-state systems but not systems with continuous time evolution such as (4.8). The analogue of Problem 4.1 for a continuous-time system could easily be stated. One key difference is that in a time-sampled system mode switches can only happen at the sampling instants $\tau\mathbb{Z}_+$, where τ is the sampling time; thus the actuation is constrained. Furthermore, a solution where state counting constraints are violated in between samplings (but satisfied at sample instants) is still a valid solution to the instance (4.7). If such inter-sample violations are unacceptable, the state-parts of counting sets can be contracted by some margin determined by the dynamics to ensure satisfaction for all $t \in \mathbb{R}_+$ [75].

In the following, the time-sampled analogue of (4.8) is constructed as a FTS in order to leverage the notion of bisimilarity for transition systems. The domain is also restricted to a bounded set $\mathcal{X} \subset \mathbb{R}^{n_x}$, resulting in the following space-restricted τ-sampled counterpart of Σ_{Sw} confined to $\mathcal{X}$:

$$\mathcal{T}_\tau = (\mathcal{X}, \mathbb{U}, \underset{\tau}{\longrightarrow}, \mathrm{Id}_{\mathbb{R}^d}), \tag{4.10}$$

where

$$x \xrightarrow[\tau]{\mu} x' \text{ if and only if } \exists \mathbf{d} : [0, \tau] \to \mathcal{D} \text{ s.t. } x' = \phi_{f_\mu}(\tau, x; \mathbf{d}).$$

The instance

$$(N, \mathcal{T}_\tau, \{\mathbf{x}_n(0)\}_{n\in[N]}, \{X_l, R_l\}_{l\in[L]}) \tag{4.11}$$

of Problem 4.1 is referred to as the *continuous-state counting problem*. Since $\mathcal{T}_\tau$ is a non-deterministic, infinite transition system, this is a difficult problem to solve. In the following a Deterministic Finite Transition System (DFTS) $\mathcal{T}_{\tau,\eta}$ that is approximately bisimilar to $\mathcal{T}_\tau$ is constructed in order to map the problem into a finite instance.

4.1.1. Abstraction Procedure

The abstraction procedure below creates a finite-state model of Σ_{Sw}. Under the assumptions outlined above, approximate bisimilarity between the continuous-state system and its finite-state abstraction is established, which enables results in Section 4.1.2 relating the solvability of the corresponding counting problem instances.

For a state discretization parameter $\eta > 0$ define an abstraction function $\alpha_\eta : \mathcal{X} \mapsto \mathcal{X}$ as

$$\alpha_\eta(x) = \eta \cdot \left\lfloor \frac{x}{\eta} \right\rfloor + \frac{\eta}{2}\mathbf{1}. \tag{4.12}$$

This function is constant on hyperboxes of side η, and its image of the compact set $\mathcal{X}$ is finite. The abstraction function defines the, (τ, η)-discretized counterpart of Σ_{Sw} as the transition system

$$\mathcal{T}_{\tau,\eta} = \left(\alpha_\eta(\mathcal{X}), \mathbb{U}, \underset{\tau,\eta}{\longrightarrow}, \mathrm{Id}_{\mathbb{R}^d}\right),$$

where

$$\xi \underset{\tau,\eta}{\xrightarrow{\mu}} \xi' \quad \text{iff } \alpha_\eta(\phi_{f_\mu}(\tau, \xi; \mathbf{0})) = \xi'.$$

In essence, the domain $\mathcal{X}$ is partitioned into uniform boxes of size η that represent discrete states. As illustrated in Figure 4.2, transition relations are established by simulating each mode, without disturbance, during a time τ, starting at the centers ξ of the boxes. As opposed to $\mathcal{T}_\tau$, the resulting transition system $\mathcal{T}_{\tau,\eta}$ is deterministic—for each state ξ and action μ there exists (at most) one successor state ξ'.

Similarly to a result from [102] it can now be shown that an abstraction constructed in this way is bisimilar to the time-sampled system (4.10) if a certain inequality holds.

Proposition 4.1. *Assume that Assumptions 4.1 and 4.2 hold, i.e., for all $\mu \in \mathbb{U}$ there are Lipschitz constants K_μ, $\mathcal{KL}$-functions β_μ, and disturbance effect bounds $\overline{\delta}_\mu$ associated with*

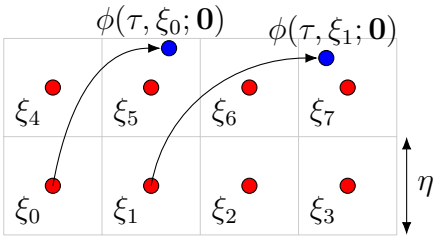

Figure 4.2: Illustration of how the abstraction $\mathcal{T}_{\tau,\eta}$ is constructed. The center of each cell of side η represents a discrete state ξ_i. A transition $\xi_0 \to \xi_5$ is established since the flow for time τ starting at ξ_0 ends up in the cell corresponding to ξ_5, and similarly for $\xi_1 \to \xi_7$.

the modes of (4.8)*. If for all* $\mu \in \mathbb{U}$*,*

$$\beta_\mu(\epsilon, \tau) + \frac{\overline{\delta}_\mu}{K_\mu}\left(e^{K_\mu \tau} - 1\right) + \frac{\eta}{2} \leq \epsilon, \tag{4.13}$$

then $\mathcal{T}_{\tau,\eta} \cong_\epsilon \mathcal{T}_\tau$*.*

Proof. Consider the relation $\xi \sim x$ iff $\|\xi - x\|_\infty \leq \epsilon$. It evidently satisfies 1) of Definition 1.11. Item 2) of the same definition is trivially satisfied since all transitions of $\mathcal{T}_{\tau,\eta}$ are present in $\mathcal{T}_\tau$ as the special case without disturbance.

What remains is to show that item 3) holds. To this end, assuming that $\xi \sim x$, it must be shown that $\xi' \sim \phi_{f_\mu}(\tau, x; \mathbf{d})$ for all $\mathbf{d} : [0, \tau] \to \mathcal{D}$, where $\xi' \in \alpha_\eta(\mathcal{X})$ is the μ-successor of ξ in $\mathcal{T}_{\tau,\eta}$.

It follows from the construction of $\mathcal{T}_{\tau,\eta}$ that $\|\xi' - \phi_{f_\mu}(\tau, \xi; \mathbf{0})\|_\infty \leq \eta/2$. Furthermore, under the continuity assumptions, it is known that the flow $\phi_{f_\mu}(t, x; \mathbf{0})$ of $f_\mu(x, 0)$ and the flow $\phi_{f_\mu}(t, x, \mathbf{d})$ for any $\mathbf{d}$ such that $\|f_\mu(x, 0) - f_\mu(x, \mathbf{d}(t))\|_\infty \leq \overline{\delta}_\mu$ satisfy $\|\phi_{f_\mu}(t, x; \mathbf{0}) - \phi_{f_\mu}(t, x; \mathbf{d})\|_\infty \leq (\overline{\delta}_\mu / K_\mu)(e^{K_\mu t} - 1)$ [58].

Thus, it follows that for any such $\mathbf{d} : [0, \tau] \to \mathcal{D}$,

$$\begin{aligned}
&\|\xi' - \phi_{f_\mu}(\tau, x; \mathbf{d})\|_\infty \leq \|\phi_{f_\mu}(\tau, x; \mathbf{d}) - \phi_{f_\mu}(\tau, x; \mathbf{0})\|_\infty + \|\phi_{f_\mu}(\tau, x; \mathbf{0}) - \phi_{f_\mu}(\tau, \xi; \mathbf{0})\|_\infty \\
&\quad + \|\phi_{f_\mu}(\tau, \xi; \mathbf{0}) - \xi'\|_\infty \leq \frac{\overline{\delta}_\mu}{K_\mu}\left(e^{K_\mu \tau} - 1\right) + \beta_\mu\left(\|\xi - x\|_\infty, \tau\right) + \frac{\eta}{2} \\
&\quad \leq \frac{\overline{\delta}_\mu}{K_\mu}\left(e^{K_\mu \tau} - 1\right) + \beta_\mu\left(\epsilon, \tau\right) + \frac{\eta}{2} \leq \epsilon.
\end{aligned}$$

Hence $\xi' \sim \phi_{f_\mu}(\tau, x; \mathbf{d})$ which completes the proof. □

It is known that the trajectories of ϵ-approximately bisimilar systems remain within distance ϵ of each other [48]. This fact is the key to the following two results that establish relations between existence of solutions of the counting problem in the continuous-state and finite-state settings.

4.1.2. Counting Problems and Abstractions

If ϵ-approximate bisimilarity holds, trajectories of $\mathcal{T}_{\tau,\eta}$ and $\mathcal{T}_\tau$ are guaranteed to remain ϵ-close when corresponding control actions are chosen. Therefore, solvability of the counting problem is equivalent for the two up to an approximation margin ϵ. To make precise statements, consider functions $\mathcal{G}^{\pm\epsilon}$ that expand (resp. contract) a counting set $X = X^{\mathcal{X}} \times X^{\mathbb{U}}$ in the state space domain before abstraction:

$$\mathcal{G}^{+\epsilon}(X^{\mathcal{X}} \times X^{\mathbb{U}}) = \alpha_\eta \left(X^{\mathcal{X}} \oplus \mathcal{B}_\infty(0,\epsilon)\right) \times X^{\mathbb{U}},$$
$$\mathcal{G}^{-\epsilon}(X^{\mathcal{X}} \times X^{\mathbb{U}}) = \alpha_\eta \left(X^{\mathcal{X}} \ominus \mathcal{B}_\infty(0,\epsilon)\right) \times X^{\mathbb{U}}.$$

The next results describe how solutions of the counting problem can be mapped between the time-sampled system $\mathcal{T}_\tau$ and the time-state abstracted system $\mathcal{T}_{\tau,\eta}$.

Theorem 4.1. *Let $\mathcal{T}_\tau$ and $\mathcal{T}_{\tau,\eta}$ be the time-sampled and time-state discretized systems constructed from a system of the form (4.8), such that $\mathcal{T}_\tau$ and $\mathcal{T}_{\tau,\eta}$ are ϵ-approximately bisimilar. Let α_η be the abstraction function for $\mathcal{T}_{\tau,\eta}$.*

If there exists a solution to the instance

$$\left(N, \mathcal{T}_{\tau,\eta}, \{\alpha_\eta(\mathbf{x}_n(0))\}_{n\in[N]}, \{\mathcal{G}^{+\epsilon}(X_l), R_l\}_{l\in[L]}\right) \tag{4.14}$$

of Problem 4.1, then there exists a solution to the instance

$$(N, \mathcal{T}_\tau, \{\mathbf{x}_n(0)\}_{n\in[N]}, \{X_l, R_l\}_{l\in[L]}). \tag{4.15}$$

Proof. Let $\{\pi_n\}_{n\in[N]}$ be individual switching protocols that solve (4.14) by generating trajectories $\zeta_n(0)\zeta_n(1)\ldots$ and actions $\sigma_n(0)\sigma_n(1)\ldots$ for $\mathcal{T}_{\tau,\eta}$. Due to bisimilarity and the $\eta/2$-proximity of initial conditions, the individual trajectories $\zeta_n(0)\zeta_n(1)\ldots$ of $\mathcal{T}_{\tau,\eta}$ and the individual trajectories $\mathbf{x}_n(0)\mathbf{x}_n(1)\ldots$ of $\mathcal{T}_\tau$ satisfy $\|\zeta_n(k) - \mathbf{x}_n(k)\|_\infty \leq \epsilon$ for all $k \in \mathbb{Z}_+$

when the input sequence $\sigma_n(0)\sigma_n(1)\ldots$ is implemented for both systems. By assumption,

$$\sum_{n\in[N]} \mathbb{1}_{\mathcal{G}^{+\epsilon}(X_l)}\left(\zeta_n(k),\sigma_n(k)\right) \leq R_l.$$

Thus for a counting set $X_l = X_l^x \times X_l^u$,

$$\mathbf{x}_n(k) \in X_l^x \implies \zeta_n(k) \in X_l^x \oplus \mathcal{B}_\infty(0,\epsilon) \implies \zeta_n(k) \in \alpha_\eta\left(X_l^x \oplus \mathcal{B}_\infty(0,\epsilon)\right),$$

where the last step follows from knowing that $\zeta_n(k)$ only takes values x such that $\alpha_\eta(x) = x$. Thus,

$$\sum_{n\in[N]} \mathbb{1}_{X_l}\left(\mathbf{x}_n(k),\sigma_n(k)\right) \leq \sum_{n\in[N]} \mathbb{1}_{\mathcal{G}^{+\epsilon}(X_l)}\left(\zeta_n(k),\sigma_n(k)\right) \leq R_l,$$

which shows that the corresponding counting constraint in (4.15) is satisfied. □

Theorem 4.2. *Under the same assumptions as in Theorem 4.1, if there is no solution to the instance*

$$\left(N, \mathcal{T}_{\tau,\eta}, \{\alpha_\eta(\mathbf{x}_n(0))\}_{n\in[N]}, \left\{\mathcal{G}^{-(\epsilon+\frac{\eta}{2})}(X_l), R_l\right\}_{l\in[L]}\right) \tag{4.16}$$

of Problem 4.1, then there is no solution to the instance

$$\left(N, \mathcal{T}_\tau, \{\mathbf{x}_n(0)\}_{n\in[N]}, \{X_l, R_l\}_{l\in[L]}\right). \tag{4.17}$$

Proof. Suppose for contradiction that there is a solution to (4.17) but not to (4.16). Let $\{\pi_n\}_{n\in[N]}$ be individual switching policies that solve (4.17) by generating trajectories $\mathbf{x}_n(0)\mathbf{x}_n(1)\ldots$ and actions $\sigma_n(0)\sigma_n(1)\ldots$ for $\mathcal{T}_\tau$. Due to bisimilarity and the $\eta/2$-proximity of initial conditions, the individual trajectories $\zeta_n(0)\zeta_n(1)\ldots$ of $\mathcal{T}_{\tau,\eta}$ and the individual trajectories $\mathbf{x}_n(0)\mathbf{x}_n(1)\ldots$ of $\mathcal{T}_\tau$ satisfy $\|\zeta_n(k)-\mathbf{x}_n(k)\| \leq \epsilon$ for all $k \in \mathbb{Z}_+$ when the actions $\sigma_n(0)\sigma_n(1)\ldots$ are implemented for both systems. For a set A,

$$\alpha_\eta\left(A \ominus \mathcal{B}_\infty\left(0,\frac{\eta}{2}\right)\right) \subset \{x \in A : \alpha_\eta(x) = x\} \subset A.$$

Thus,

$$\begin{aligned}\zeta_n(t) \in \alpha_\eta\left(X_l^x \ominus \mathcal{B}_\infty\left(0,\epsilon+\frac{\eta}{2}\right)\right) &\implies \mathbf{x}_n(t) \in X_l^x \ominus \mathcal{B}_\infty\left(0,\epsilon\right)\\ &\implies \mathbf{x}_n(t) \in \left(X_l^x \ominus \mathcal{B}_\infty\left(0,\epsilon\right)\right) \oplus \mathcal{B}_\infty(0,\epsilon) \subset X_l^x.\end{aligned}$$

It follows that

$$\sum_{n\in[N]} \mathbb{1}_{\mathcal{G}^{-(\epsilon+\eta/2)}(X_l)}(\zeta_n(t),\sigma_n(t)) \leq \sum_{n\in[N]} \mathbb{1}_{X_l}(\mathbf{x}_n(t),\sigma_n(t)) \leq R_l,$$

thus $\{\pi_n\}_{n\in[N]}$ is a solution also for (4.16)—a contradiction. □

Example 4.3. *The constraints of the TCL scheduling are discretized. The mode-counting constraints* (4.4) *lack a state space part, and are therefore not modified. However, the local safety constraints* (4.5) *must be tightened to*

$$\sum_{n\in[N]} \mathbb{1}_{[\underline{a}+\epsilon,\overline{a}-\epsilon]^C\times\{\texttt{on},\texttt{off}\}}(\mathbf{x}_n(k),\sigma_n(k)) \leq 0,$$

in order for a solution of the finite-state instance to be mappable to a valid solution of the continuous-state instance of Problem 4.1.

4.2. Solving the Discrete-State Counting Problem

Having described the reduction of a continuous-state instance of Problem 4.1 to a deterministic finite-state instance, a solution procedure is presented for the latter. Consider a DFTS $\mathcal{T}_{DFTS} = (\mathbb{X}, \mathbb{U}, \longrightarrow, h)$; it may be the result of a continuous-state abstraction or just represent a discrete structure onto which a counting problem is defined. A DFTS can alternatively be viewed as a directed graph $G = (\mathbb{X}, \longrightarrow)$, where $\mathbb{X}$ is a set of nodes and $\longrightarrow$ is a set of edges, and each edge is labeled with an action from $\mathbb{U}$. This dual viewpoint enables graph-theoretical concepts to be leveraged to investigate the aggregate system; in the following the transition system viewpoint and the graph viewpoint are used interchangeably.

Before proceeding, recall some standard graph notions for a directed graph $G = (\mathbb{X}, \longrightarrow)$. A *path* in G is a list of edges $(\xi_0,\xi_1)(\xi_1,\xi_2)\ldots(\xi_{J-1},\xi_J)$. The distance between two nodes ξ_0 and ξ_J is the length of the shortest path connecting the two. The graph diameter $\mathrm{diam}(G)$ is the longest distance between two nodes in the graph. If the first and last nodes in a path are equal, i.e. $\xi_J = \xi_0$, the path is a *cycle*. A cycle is *simple* if it visits every node at most one time. For a subset of nodes $D \subset \mathbb{X}$ (or the corresponding subgraph, which is used interchangeably), it is said to be *strongly connected* if for each node pair $(\xi,\xi') \in D$, there exists a path from ξ to ξ'. Any directed graph can be decomposed into strongly connected

components. The *period* of a subgraph D is the greatest common divisor of all cycles in D. A subgraph D is called *aperiodic* if it has period one.

Next, aggregate dynamics are introduced which take the form of a constrained linear system, and interesting connections between properties of the graph G and reachability properties of the aggregate system are explored.

4.2.1. Aggregate Dynamics as a Linear System

Consider a total of N subsystems whose dynamics are governed by a DFTS $\mathcal{T}_{DFTS} = (\mathbb{X}, \mathbb{U}, \longrightarrow, h)$. Let aggregate states labeled $\mathbf{w}_\xi$ for $\xi \in \mathbb{X}$ describe the number of individual subsystems that are *at node* ξ. By also introducing control actions $\mathbf{r}_\xi^\mu$ that represent the number of subsystems *at node* ξ *using action* μ, the aggregate dynamics can be written as

$$\mathbf{w}_\xi(k+1) = \sum_{\mu \in \mathbb{U}} \sum_{\xi' \in \mathcal{N}_\xi^\mu} \mathbf{r}_{\xi'}^\mu(k), \quad \xi \in \mathbb{X}, \tag{4.18}$$

where $\mathcal{N}_\xi^\mu$ is the index set of predecessors of ξ under the action μ:

$$\mathcal{N}_\xi^\mu = \{\xi' \in \mathbb{X} : (\xi', \mu, \xi) \in \longrightarrow\}.$$

The control actions $\mathbf{r}_\xi^\mu$ are constrained such that for all $\xi \in \mathbb{X}$ and $\mu \in \mathbb{U}$,

$$\mathbf{r}_\xi^\mu(k) \geq 0, \quad \sum_{\xi \in \mathbb{U}} \mathbf{r}_\xi^\mu(k) = \mathbf{w}_\xi(k), \tag{4.19}$$

which ensures the continued positivity of the states: $\mathbf{w}_\xi(k+1) \geq 0$ for $\xi \in \mathbb{X}$. Furthermore, the invariant $\sum_{\xi \in \mathbb{X}} \mathbf{w}_\xi = N$ holds over time, where N is the total number of subsystems. In the following, the compact notation

$$\Sigma_{agg} : \ \mathbf{w}(k+1) = B\mathbf{r}(k),$$

is employed to denote this aggregate system. Here B is composed of the incidence matrices $A^\mu, \mu \in \mathbb{U}$ of the mode-transition graphs. The state space $\mathcal{W}$ and (state-dependent)

admissible control set $\mathcal{R}$ of this system are then

$$\mathcal{W} = \left\{ \mathbf{w} \in \mathbb{Z}_+^K : \|\mathbf{w}\|_1 = N \right\},$$
$$\mathcal{R}(\mathbf{w}) = \left\{ \mathbf{r} \in \mathbb{Z}_+^{KM} : \text{ (4.19) holds for } (\mathbf{w}, \mathbf{r}) \right\}.$$

4.2.2. Graph Properties

Next properties of the directed graph $G = (\mathbb{X}, \longrightarrow)$ are connected to reachability properties of the aggregate dynamics.

First a concept of controllability on a subset of nodes D is defined for the aggregate dynamics Σ_{agg}. Similarly as for controllability of linear systems on a subspace, controllability of Σ_{agg} on D means that the system can be steered between any two aggregate states with support on D.

Definition 4.2. *A subset of nodes $D \subset \mathbb{X}$ is **completely controllable** for Σ_{agg} if for any two states $\mathbf{w}$, $\mathbf{w}'$ with support[21] on D such that $\|\mathbf{w}\|_1 = \|\mathbf{w}'\|_1$, there exists a finite horizon T, states $\{\mathbf{w}(k)\}_{k=0}^{T}$, and controls $\{\mathbf{r}(k)\}_{k=0}^{T-1}$ satisfying (4.19), such that $\mathbf{w}(0) = \mathbf{w}$, $\mathbf{w}(T) = \mathbf{w}'$, and $\mathbf{w}(k+1) = B\mathbf{r}(k)$ for $k \in [T]$.*

Theorem 4.3. *If a strongly connected component D is aperiodic, it is completely controllable for Σ_{agg}.*

Proof. It is known that the incidence matrix A_D of an aperiodic, strongly connected graph D is primitive [117], i.e., there exists an integer T such that all entries of A_D^T are positive. This means that for each node pair (ξ_j, ξ_l), there exists a path of length T that connects them. Thus, by sending p_{jl} systems along paths $\xi_j \to \xi_l$ such that $\sum_l p_{jl} = \mathbf{w}_j$ and $\sum_j p_{jl} = \mathbf{w}'_l$, the state at time T is equal to $\mathbf{w}'$. Aggregate controls $\mathbf{r}(k)$ that realize these paths can be defined by switching the correct number of systems at each node over time. □

In the case of periodicity, it is not possible to reach every state since the parity structure of the initial state is preserved along the trajectories. However, within this restriction, the system is still controllable in a certain sense. If a strongly connected component D has period P, its nodes can be labeled with a function $L_P : D \to [P]$ such that a node ξ with $L_P(\xi) = p$ only has edges to nodes ξ' with $L_P(\xi') = (p+1) \mod P$. Let $D_0, \ldots, D_{P-1}$ be the subsets of nodes induced by the equivalence relation $\xi \sim \xi'$ iff $L_P(\xi) = L_P(\xi')$.

[21] A state $\mathbf{w}$ having support on D means that $\mathbf{w}_\xi = 0$ for $\xi_k \notin D$.

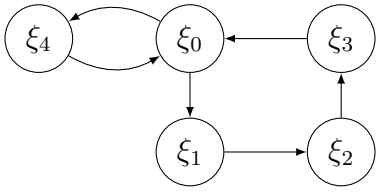

Figure 4.3: A FTS with two cycles and period 2. The cycles are $\{\xi_0, \xi_1, \xi_2, \xi_3\}$ and $\{\xi_0, \xi_4\}$ of length 2 and 4, respectively. Since $\gcd(2, 4) = 2$, the period of the system is 2.

Corollary 4.1. *The subsets of nodes D_p for $p \in [P]$ as constructed above are completely controllable for Σ_{agg}.*

Proof. The nodes in D_p can be connected with edges that correspond to paths of length P in D. By construction, the resulting graphs are aperiodic, so the previous result applies. □

It is well-known that if a discrete-time linear system is completely controllable, its reachable set for a time $k \geq n$, where n is the system dimension, is the entire state-space; otherwise, it is an affine subspace that depends on k and the initial state. The preceding results show a corresponding property for the aggregate dynamics Σ_{agg}—a linear system with input constraints evolving on an integer lattice. The controllability in this setting is entirely characterized by the properties of the graph representing the abstraction. For the controllable case (aperiodic graph), the reachable set of Σ_{agg} at a time $k \geq T$, where T is the controllability horizon from the proof of Theorem 4.3, is the set of all positive integer-valued vectors satisfying $\|\mathbf{w}\|_1 = N$. In case of periodicity with a period P, it is the intersection of this lattice set with an affine subspace that depends on ($k \mod P$) and the parity structure of the initial state, as illustrated in the following example.

Example 4.4. *Consider the FTS in Figure 4.3, it consists of two cycles of length 2 and 4, so the period is 2. The two equivalence classes induced by the periodicity are $\{\xi_0, \xi_2\}$ and $\{\xi_1, \xi_3, \xi_4\}$; all subsystems in the first class move to the second, and vice versa. Consider an initial state such that*

$$\mathbf{w}_{\xi_0}(0) + \mathbf{w}_{\xi_2}(0) = c_0, \quad \mathbf{w}_{\xi_1}(0) + \mathbf{w}_{\xi_3}(0) + \mathbf{w}_{\xi_4}(0) = c_1.$$

Due to periodicity this parity structure is preserved in the sense that

$$\mathbf{w}_{\xi_0}(k) + \mathbf{w}_{\xi_2}(k) = \begin{cases} c_0, & \text{if } k \text{ is even,} \\ c_1, & \text{if } k \text{ is odd,} \end{cases} \qquad \mathbf{w}_{\xi_1}(k) + \mathbf{w}_{\xi_3}(k) + \mathbf{w}_{\xi_4}(k) = \begin{cases} c_1, & \text{if } k \text{ is even,} \\ c_0, & \text{if } k \text{ is odd.} \end{cases}$$

However, within this restriction Corollary 4.1 implies that any assignment is reachable when k is large enough.

4.2.3. Graph Assignments

Attention is restricted to solution trajectories that consist of a finite prefix phase and a periodic suffix phase defined on cycles. The main idea is to steer the subsystems to cycles during the prefix part, and to let them follow periodic trajectories over the cycles in the suffix part. Controllability of the aggregate dynamics plays a role in what type of periodic trajectories the subsystems can be steered to. Periodic trajectories enable guarantees that hold over an infinite time horizon, using a suffix that is finitely described.

In the following, cycles C are assumed to be of the form

$$C = (\xi_0, \mu_0, \xi_1)(\xi_1, \mu_1, \xi_2) \dots (\xi_{I-1}, \mu_{I-1}, \xi_0), \tag{4.20}$$

of length given by $|C| = I$, where $(\xi_i, \mu_i, \xi_{i+1}) \in \longrightarrow$.

In order to map from an index i of a specific cycle C to the corresponding graph nodes, let $\Phi_C^{\mathbb{X}} : [|C|] \to \mathbb{X}$ be a function that for a cycle C of the form (4.20) maps the cycle index i to the corresponding node ξ_i in the graph G; that is, $\Phi_C^{\mathbb{X}}(i) = \xi_i$. Similarly, define a function $\Phi_C^{\mathbb{U}}$ that specifies the outgoing mode in C for cycle index i; that is, $\Phi_C^{\mathbb{U}}(i) = \mu_i$.

Next the concept of a *cycle assignment* is defined; an assignment distributes "weights" (or numbers) of subsystems to each node along a cycle. If the graph represents an abstraction, the assignment counts the number of subsystems whose continuous states are in the vicinity of the abstract states on this cycle.

Definition 4.3. *An **assignment** to a cycle C is a function $\gamma : [|C|] \to \mathbb{R}_+$.*

Definition 4.4. *An **integer assignment** to a cycle C is an assignment γ to C such that $\gamma(i) \in \mathbb{Z}_+$ for $i \in [|C|]$.*

For a subsystem assigned to a cycle, its state circulates along the abstract states on this

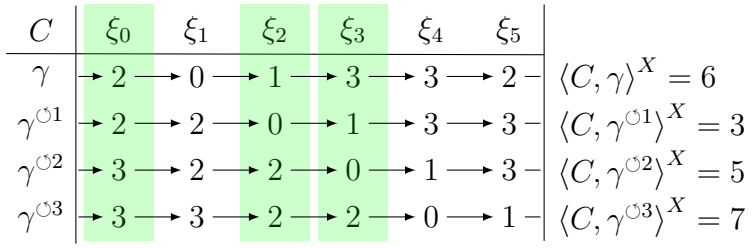

Figure 4.4: Illustration of cycles and circulating assignments. The diagram represents a cycle C comprising nodes $\xi_0, \ldots, \xi_5$ matched with a circulating assignment $\gamma = [2, 0, 1, 3, 3, 2]$. The right column shows how the X-count varies as the assignment γ circulates for a counting set $X \supset \{\xi_0, \xi_2, \xi_3\}$, i.e. matching the cycle indices 0, 2, and 3.

cycle as time progresses (provided that the appropriate actions are chosen). The movement corresponds to a circular shift of the assignment.

Definition 4.5. *For an assignment γ its k-step **circulation**, denoted $\gamma^{\circlearrowleft k} : [|C|] \to \mathbb{R}_+$, is the shifted function*

$$\gamma^{\circlearrowleft k}(i) = \gamma\left((i-k) \mod |C|\right).$$

The periodicity is manifested by the relation $\gamma^{\circlearrowleft(|C|+k)} = \gamma^{\circlearrowleft k}$. To capture how counting quantities vary during the circulation, the following notation for the matching of a cycle and a circulated assignment is used.

Definition 4.6. *For a cycle C and an assignment $\gamma : [|C|] \to \mathbb{R}_+$ the X-count at time k of a counting set X is defined as*

$$\left\langle C, \gamma^{\circlearrowleft k} \right\rangle^X = \sum_{i \in [|C|]} \mathbb{1}_X\left(\Phi_C^{\mathbb{X}}(i), \Phi_C^{\mathbb{U}}(i)\right) \gamma^{\circlearrowleft k}(i). \tag{4.21}$$

If the cycle includes elements from a set X (typically coming from a counting constraint), (4.21) counts the number of subsystems contributing to the counting constraint at time k assuming that the subsystems make up the assignment γ at time zero and follow the cycles. The concept is illustrated in Figure 4.4.

Example 4.5. *For a given node $\xi \in \mathbb{X}$ and time $k \in \mathbb{Z}_+$, the expression*

$$\left\langle C, \gamma^{\circlearrowleft k} \right\rangle^{\{\xi\} \times \mathbb{U}} \tag{4.22}$$

represents the number of subsystems assigned to ξ by the assignment γ when it has circulated k steps in C. To illustrate, for the example in Figure 4.5 on page 105, (4.22) *evaluates to the following values for $\xi_0, \ldots, \xi_5$ and $k = 0, \ldots, 4$:*

$$\begin{array}{c} \\ k=0 \\ k=1 \\ k=2 \\ k=3 \\ k=4 \end{array} \begin{array}{c} \begin{array}{cccccc} \xi_0 & \xi_3 & \xi_4 & \xi_5 & \xi_2 & \xi_1 \end{array} \\ \begin{pmatrix} 0 & 3 & 4 & 4 & 3 & 2 \\ 0 & 2 & 3 & 4 & 4 & 3 \\ 0 & 3 & 2 & 3 & 4 & 4 \\ 0 & 4 & 3 & 2 & 3 & 4 \\ 0 & 4 & 4 & 3 & 2 & 3 \end{pmatrix} \end{array}.$$

The next definition introduces functions that for given cycle-assignment pairs return the highest attained count over all circulations.

Definition 4.7. *The **maximal X-count** for a cycle C with assignment γ, denoted $\Psi^X(C, \gamma)$, is the maximal number of subsystems simultaneously in X when γ is circulated around C:*

$$\Psi^X(C, \gamma) = \max_{k \in \mathbb{Z}_+} \left\langle C, \gamma^{\circlearrowleft k} \right\rangle^X.$$

An illustration is provided in Figure 4.5 for an example cycle-assignment pair. The maximal X-count can also be computed as the maximum entry in a matrix-vector product

$$\Psi^X(C, \gamma) = \left\| B_C^X \gamma \right\|_\infty,$$

where $B_C^X \in \{0, 1\}^{|C| \times |C|}$ is a circulant binary matrix

$$B_C^X[i, j] = \mathbb{1}_X \left(\Phi_C^{\mathbb{X}}(k_{ij}), \Phi_C^{\mathbb{U}}(k_{ij}) \right), \; k_{ij} = (i - j) \mod |C|,$$

taking the general form

$$B_C^X = \begin{bmatrix} b_0 & b_1 & \ldots & b_{|C|-1} \\ b_1 & b_2 & \ldots & b_0 \\ \vdots & \vdots & \iddots & \vdots \\ b_{|C|-1} & b_0 & \ldots & b_{|C|-2} \end{bmatrix}.$$

Definition 4.8. *The **maximal joint X-count** for a set of cycles $\{C_j\}_{j \in [J]}$ with assignments $\{\gamma_j\}_{j \in [J]}$, denoted $\Psi^X\left(\{C_j\}_{j \in [J]}, \{\gamma_j\}_{j \in [J]}\right)$, is the maximal number of subsystems*

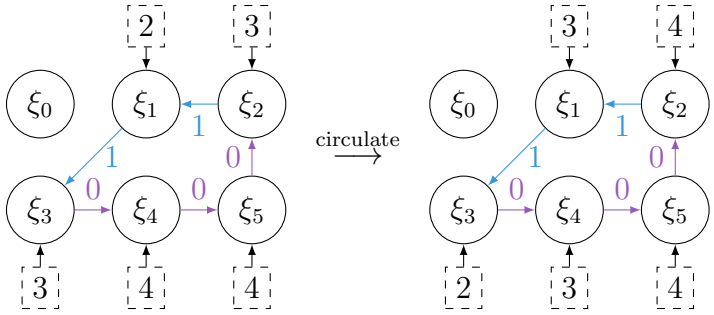

Figure 4.5: Illustration of how the maximal counts are attained during assignment circulation. The assignment $\gamma = [3, 4, 4, 3, 2]$ of length 5 is matched to a cycle $C = (\xi_3, 0, \xi_4), (\xi_4, 0, \xi_5), (\xi_5, 0, \xi_2), (\xi_2, 1, \xi_1), (\xi_1, 1, \xi_3)$ of the same length, with two modes 0 (purple) and 1 (blue). Let $X_0 = \mathbb{X} \times \{0\}$ and $X_1 = \mathbb{X} \times \{1\}$ count the total number of subsystems in each mode, irrespective of spatial position. On the left, the X_0-count is then 3+4+4 = 11, while the X_1-count is 3+2=5. After circulating the assignment one step, as displayed to the right, the X_0-count is 2+3+4 = 9 and the X_1-count is 4+3 = 7. Over all possible rotations, the maximal X_0-count is 11, so $\Psi^{X_0}(C, \gamma) = 11$. Similarly, $\Psi^{X_1}(C, \gamma) = 8$.

simultaneously in X when the assignments $\{\gamma_j\}_{j\in[J]}$ are synchronously circulated around the cycles $\{C_j\}_{j\in[J]}$:

$$\Psi^X\left(\{C_j\}_{j\in[J]}, \{\gamma_j\}_{j\in[J]}\right) = \max_{k\in\mathbb{Z}_+} \sum_{j\in[J]} \left\langle C_j, \gamma_j^{\circlearrowleft k}\right\rangle^X .$$

While the characterization in Definition 4.8 is useful in subsequent proofs, the maximal joint X-count can also be expressed as the maximal element in a matrix-vector product.

Proposition 4.2. *The maximal joint X-count satisfies*

$$\Psi^X\left(\{C_j\}_{j\in[J]}, \{\gamma_j\}_{j\in[J]}\right) = \left\| \sum_{j\in[J]} (\mathbf{1}_{k_j} \otimes B^X_{C_j})\gamma_j \right\|_\infty , \tag{4.23}$$

where $k_j = \mathrm{lcm}\left(\{|C_j|\}_{j\in[J]}\right)/|C_j|$, lcm is the least common multiple, $\mathbf{1}_k$ is the length k column vector of all ones, and $\otimes$ is the Kronecker product.

4.2.4. Solution to the Discrete-State Counting Problem

Consider now an instance

$$(N, \mathcal{T}_{DFTS}, \{\zeta_n(0)\}_{n\in[N]}, \{X_l, R_l\}_{l\in[L]}) \tag{4.24}$$

of Problem 4.1, where $\mathcal{T}_{DFTS}$ is a Deterministic Finite Transition System. Let $\mathbf{w}(0)$ be the aggregate initial state counting the number of subsystems at each node:

$$\mathbf{w}_\xi(0) = \sum_{n\in[N]} \mathbb{1}_{\{\xi\}}\left(\zeta_n(0)\right), \quad \xi \in \mathbb{X}. \tag{4.25}$$

Attention is restricted to solutions of a particular form; it is shown in later sections that this is without loss of generality.

Definition 4.9. *A control strategy for an aggregate initial state* $\mathbf{w}(0)$ *is of* ***prefix-suffix type*** *if it consists of a finite number of inputs* $\mathbf{r}(0), \ldots, \mathbf{r}(T-1)$*, and a set of cycles* $\{C_j\}_{j\in[J]}$ *with assignments* $\{\gamma_j\}_{j\in[J]}$ *such that the cycles are populated with their respective cycle assignments at time* T*.*

Given aggregate initial states $\mathbf{w}(0)$, a set $\{C_j\}_{j\in[J]}$ of cycles in G, and a prefix horizon T, prefix-suffix solutions can be extracted from feasible points of the following linear feasibility problem:

$$
\begin{aligned}
&\text{find } \gamma_0, \dots, \gamma_{J-1} \text{ (cycle assignments)},\\
&\qquad \mathbf{r}(0), \dots, \mathbf{r}(T-1) \text{ (aggregate inputs)},\\
&\qquad \mathbf{w}(1), \dots, \mathbf{w}(T) \text{ (aggregate states)},\\
&\text{s.t.} \sum_{\xi \in \mathbb{X}} \sum_{\mu \in \mathbb{U}} \mathbb{1}_{X_l}(\xi, \mu)\, \mathbf{r}_\xi^\mu(k) \le R_l, && k \in [T], l \in [L], && \text{(4.26a)}\\
&\qquad \Psi^{X_l}(\{C_j\}_{j\in[J]}, \{\gamma_j\}_{j\in[J]}) \le R_l, && l \in [L], && \text{(4.26b)}\\
&\qquad \mathbf{w}(k+1) = B\mathbf{r}(k), && k \in [T], && \text{(4.26c)}\\
&\qquad \mathbf{w}_\xi(T) = \sum_{j\in[J]} \langle C_j, \gamma_j \rangle^{\{\xi\}\times\mathbb{U}}, && \xi \in \mathbb{X}, && \text{(4.26d)}\\
&\qquad \sum_{\mu\in\mathbb{U}} \mathbf{r}_\xi^\mu(k) = \mathbf{w}_\xi(k), && k \in [T], \xi \in \mathbb{X}, && \text{(4.26e)}\\
&\qquad \mathbf{r}_\xi^\mu(k) \ge 0, && k \in [T], \xi \in \mathbb{X}, \mu \in \mathbb{U}. && \text{(4.26f)}
\end{aligned}
$$

The number of variables and (in)equalities in (4.26) is

$$
\mathcal{O}\left(T|\mathbb{X}||\mathbb{U}| + \sum_{j\in[J]} |C_j|\right) \quad \text{and} \quad \mathcal{O}\left(LT + L\,\mathrm{lcm}\left(\{|C_j|\}_{j\in[J]}\right) + T|\mathbb{X}|\right),
$$

respectively (not counting positivity constraints that solvers handle easily). Crucially, these numbers do not depend on the total number of subsystems N which makes the approach suitable for large N and moderate-sized graphs.

The set $\{C_j\}_{j\in[J]}$ of cycles is an input to the optimization problem (4.26), while the solver selects good cycles within the input set by setting the assignments of the rest to zero. Section 4.3 presents theoretical results regarding when an input cycle set is "sufficiently rich" not to compromise the existence of solutions. On the other hand, in practice, randomly sampling a set of cycles of given size and using them as input seems to be a good strategy in many problems.

4.2.5. Size Reductions of the Linear Program

As pointed out next, there are certain ways to further decrease the number of variables and/or inequalities—sometimes without loss of generality.

First of all, state-mode pairs in G with mandated 0-count can be pruned from G to decrease the number of aggregate variables. Specifically, a pruned graph $\tilde{G}$ can be constructed in the following way: if $(\xi, \mu) \in X_l$ and $R_l = 0$, then remove the action μ at ξ. If this results in nodes in $\tilde{G}$ with no outgoing edges, prune these nodes together with incoming edges and repeat until all nodes have at least one valid action. This procedure is equivalent to finding the largest controlled invariant set contained in X_l^C.

Secondly, the following result drastically reduces the number of inequalities in (4.26b) representing suffix counting constraints.

Proposition 4.3. *The joint X-count for two cycles C_0 and C_1 with co-prime length, i.e.* $\gcd(|C_0|, |C_1|) = 1$*, can be computed as*

$$\Psi^X\left(\{C_0, C_1\}, \{\gamma_0, \gamma_1\}\right) = \Psi^X\left(C_0, \gamma_0\right) + \Psi^X\left(C_1, \gamma_1\right).$$

Proof. If $|C_0|$ and $|C_1|$ are coprime, by the Chinese Remainder Theorem, the equations $k_1 = k \mod |C_1|$ and $k_2 = k \mod |C_2|$ have a unique solution $k < |C_0||C_1|$ for every pair $k_1 < |C_1|, k_2 < |C_2|$. It follows that every circulation of γ_0 in C_0 at some point coincides with every circulation of γ_1 in C_1; hence the maximal joint X-count is equal to the sum of individual maximal X-counts for the two cycles. □

Therefore, for sets of cycles $\{C_j\}_{j\in[J]}$ and $\{C_{j'}\}_{j'\in[J']\setminus[J]}$ such that $\gcd(|C_j|, |C_{j'}|) = 1$ for all pairs $j \in [J]$, $j' \in [J'] \setminus [J]$, it holds that

$$\begin{aligned}\Psi^X\left(\{C_j\}_{j\in[J']}, \{\gamma_j\}_{j\in[J']}\right) = \Psi^X\big(\{C_j\}_{j\in[J]}, \{\gamma_j\}_{j\in[J]}\big) \\ + \Psi^X\big(\{C_j\}_{j\in[J']\setminus[J]}, \{\gamma_j\}_{j\in[J']\setminus[J]}\big).\end{aligned} \tag{4.27}$$

Thus, if the cycle set can be partitioned into sets of cycles with mutually co-prime length, the number of inequalities can be reduced.

Example 4.6. *To exemplify this reduction, consider a set of cycles with lengths ranging from 2 to 20:*

$$\begin{aligned}&\operatorname{lcm}([21]) = 232792560,\\ &\operatorname{lcm}([21] \setminus \{11, 13, 17, 19\}) + 11 + 13 + 17 + 19 = 5100.\end{aligned}$$

The number 232792560 is the number of inequalities in the naïve approach (4.23)*, which by* (4.27) *can be drastically reduced to 5100 if the cycles of prime lengths* $11, 13, 17$*, and* 19

are considered separately.

If the number of constraints is still prohibitively large, it can be replaced by a conservative constraint as follows.

Remark 4.2. *The constraint* (4.26b) *can be substituted by the conservative constraint*

$$\sum_{i\in\mathbb{Z}_+} \Psi^{X_l}(\{C_j\}_{j:|C_j|=i}, \{\gamma_j\}_{j:|C_j|=i}) \leq R_l, \quad \forall l \in [L],$$

which groups cycles by cycle length i and disregards effects from periodicity. The number of constraints in this case becomes $L(1 + \|\bigcup_{j\in[J]}\{|C_j|\}\|_1)$ instead of $L \operatorname{lcm}\big(\{|C_j|\}_{j\in[J]}\big)$, where $\bigcup_{j\in[J]}\{|C_j|\})$ is the set of cycle lengths without repetition.

While the total number of subsystems N does not impact the number of inequalities or constraints, it might affect the performance of integer linear program solvers since the number of candidate integer points grows with N. In addition, the converse results in Section 4.3 depend on N—in order to prove infeasiblity of the problem very large horizons T and/or cycle sets may be required. If N is prohibitively large for this purpose it can be scaled down if a certain divisibility condition holds: if there is a common divisor S that divides $\mathbf{w}_\xi(0)$ for all $\xi \in \mathbb{X}$, and that divides R_l for all $l \in [L]$, then there is a 1-1 correspondence between solutions of (4.26) and solutions of its analogue obtained from the substitutions $\mathbf{w}(0) \to \mathbf{w}(0)/S$ and $R_l \to R_l/S$. The correspondence simply consists in scaling $\mathbf{r}$, $\mathbf{w}$, and the assignments γ with the same S.

4.2.6. Control Strategy Extraction

The section is concluded with a switching protocol that solves the instance (4.24) from a feasible solution of (4.26).

Theorem 4.4. *If $\{\mathbf{r}(k)\}_{k\in[T]}$, $\{\mathbf{w}(k)\}_{k\in[T+1]}$, $\{\gamma_j\}_{j\in[J]}$ form a feasible integer solution of* (4.26)*, then input selection according to the switching protocol in Algorithm 3 is recursively feasible, and solves the instance* (4.24)*.*

A proof of the theorem is provided on page 148 in Appendix C.

Remark 4.3. *The switching protocol in Algorithm 3 is memoryless; input construction depends only on the current states $\{\zeta_n(k)\}_{n\in[N]}$ and auxiliary information from the solution*

Algorithm 3: Switching protocol.

Data: Time k, current state $\zeta_n(k)$ for $n \in [N]$
Result: Switching signals $\sigma_n(k)$ for $n \in [N]$

if $k < T$ **then**
  Select $\sigma_n(k)$ s.t. $\forall \xi \in \mathbb{X}, \mu \in \mathbb{U}$; $\sum_{n\in[N]} \mathbb{1}_{\{(\xi,\mu)\}}(\zeta_n(k), \sigma_n(k)) = \mathbf{r}_\xi^\mu(k)$;
else
  Select $\sigma_n(k)$ s.t. $\forall \xi \in \mathbb{X}, \mu \in \mathbb{U}$;
  $\sum_{n\in[N]} \mathbb{1}_{\{(\xi,\mu)\}}(\zeta_n(k), \sigma_n(k)) = \sum_{j\in[J]} \left\langle C_j, \gamma_j^{\circlearrowleft k-T} \right\rangle^{\{(\xi,\mu)\}}$;
end

Table 4.1: Conclusions from (in)feasibility of linear program. Given feasibility or infeasiblity of (4.26) in various configurations, this table lists the inferences that can be made according to results in Section 4.3.

	Feasible	**Cycle set**	T	**Result**	**Why?**
ILP	Yes	Any	Any	Solution	Thm 4.4
LP	Yes	Any	Any	Approx. solution	Thm 4.7,4.8
ILP	No	$\{C : \lvert C\rvert \leq \lvert\mathbb{X}\rvert\binom{\lvert\mathbb{X}\rvert+N-1}{N}\}$	$\binom{\lvert\mathbb{X}\rvert+N-1}{N}$	No solution exists	Thm 4.5
LP	ϵ-No	$\{C : C$ simple$\}$	$\frac{(\mathrm{diam}(G)^2+1)N}{\epsilon}$	No solution exists	Thm 4.6

of (4.26)*. However, for implementation central coordination is required at each time step. The coordination requirement can be relaxed by simulating the system up to time T and assigning an individual prefix and suffix to each subsystem. Then decentralized open-loop controllers can be constructed that realize these individual prefix-suffix paths and mimic the performance of the centralized protocol without communication requirements.*

4.3. Analysis of Linear Program

Theorem 4.4 establishes that solving (4.26) provides a correct solution. Next completeness of the solution approach is discussed, along converse result detailing what information that can be obtained from (in)feasibility of (4.26) when it is solved as an ILP, and when integer constraints are relaxed to obtain a more efficiently solvable LP. Table 4.1 summarizes the results of this section.

i	0	1	2	3	4	5
γ	2	0	1	4	3	2
$\bar{\gamma}_{\{12\}}$	2	2	2	2	2	2
$\bar{\gamma}_{\{3,9\}}$	1	3	1	3	1	3
$\bar{\gamma}_{\{2,4,6\}}$	1	2	3	1	2	3

Figure 4.6: Averaging of assignments. A non-average assignment $\gamma : [6] \to \mathbb{Z}_+$, and three averaged assignment with periods 1 (aperiodic), 2, and 3. All assignments have total weight 12, i.e. $\|\gamma\|_1 = \|\bar{\gamma}_{\{12\}}\|_1 = \|\bar{\gamma}_{\{3,6\}}\|_1 = \|\bar{\gamma}_{\{2,4,6\}}\|_1 = 12$. Note that average assignments are not necessarily integral.

4.3.1. Converse Results

The first result states that the restriction to prefix-suffix form is without loss of generality, provided that the prefix horizon is sufficiently large, and that the suffix cycle set is sufficiently rich. A proof is given on page 150 in Appendix C.

Theorem 4.5. *Suppose that there is a solution to the instance* (4.24)*. Then there is a feasible integer solution to* (4.26) *with a prefix length T of at most $\binom{|\mathbb{X}|+N-1}{N}$ and a suffix consisting of cycles of length at most $|\mathbb{X}|\binom{|\mathbb{X}|+N-1}{N}$.*

The number of cycles up to a given length can be very large; the next result restricts the analysis to a much smaller set. The key observation is that an assignment can be "averaged" over its cycle without violating any counting bounds. The averaging idea is illustrated in Figure 4.6 and captured in the following definition.

Definition 4.10. *For a cycle C, a graph period P dividing $|C|$, and total weights $\{N_p\}_{p\in P}$, the P-average assignment $\bar{\gamma}_{\{N_p\}_{p\in[P]}}$ is defined as*

$$\bar{\gamma}_{\{N_p\}_{p\in[P]}}(i) = \frac{N_{(i \mod P)}}{|C|/P}, \quad \forall i \in [|C|].$$

In the case $P = 1$, this assignment has constant counting bounds for any cycle, more precisely;

$$\left\langle C, \bar{\gamma}_{N_0}^{\circlearrowleft k} \right\rangle^X = = \frac{N_0}{|C|} \langle C, \mathbf{1} \rangle^X \tag{4.28}$$

for all k, where $\langle C, \mathbf{1} \rangle^X$ simply counts the number of node-action pairs in C that are in

the counting set X. As a consequence, for any assignment γ, it holds that

$$\Psi^X\left(C, \bar{\gamma}_{\|\gamma\|_1}\right) \leq \Psi^X(C, \gamma). \tag{4.29}$$

In other words, if the averaged assignment for a given total weight does not satisfy counting bounds, no assignment does.

A more general result (Lemma C.1 in Appendix C) shows that a cyclic integer suffix can be mapped into a (possibly non-integer) suffix defined on simple cycles via averaging. It turns out that these averaged assignments are reachable given an infinitesimal relaxation of the counting bounds, since they preserve the parity structure of the initial condition. The reachability results in Section 4.2.1 do not take counting constraints into account; when such constraints are present they may conflict with reachability. Nevertheless, by introducing an arbitrarily small relaxation of the counting constraints, it can be ensured that reachability is preserved. The magnitude of the relaxation does however impact the worst-case time required to control the aggregate system to a new state.

Theorem 4.6. *Suppose that there exists an integer solution to the instance*

$$(N, \mathcal{T}_{DFTS}, \{\zeta_n(0)\}_{n\in[N]}, \{X_l, R_l\}_{l\in[L]}). \tag{4.30}$$

Let $\mathrm{diam}(G)$ *be the diameter of the directed graph* G*. Then, if every counting constraint* (X_l, R_l) *is relaxed with an absolute factor* ϵ *to* $(X_l, R_l + \epsilon)$*, the non-integer version of the linear program* (4.26) *with prefix horizon* $\frac{(\mathrm{diam}(G)^2+1)N}{\epsilon}$ *and the cycle set consisting of all simple cycles, is feasible.*

A proof is provided on page 152 in Appendix C.

4.3.2. Rounding of Non-Integer Solution

If the linear program (4.26) is too large to be solvable as an integer program it may still be possible to solve it using a standard LP solver and round the result to obtain an integer solution. One option is to use a probabilistic discrepancy-minimizing rounding procedure (e.g. [78]) which allows specified relationships to be preserved after rounding; thus introducing a counting constraint violation but maintaining the validity of the solution (e.g. dynamics, prefix-suffix connection). Here a heuristic to round only the suffix part of the solution is proposed, together with an analysis of its performance under certain

assumptions on cycle structure. For a given integer suffix, the prefix part of (4.26) can be solved to find a matching prefix—a problem that is typically much smaller.

Counting constraints may be violated as a result of the rounding but the magnitude of the worst-case violation can be bounded. Given an aperiodic graph and (non-integer) cycle-assignment pairs $\{C_j, \gamma_j\}_{j\in[J]}$ that satisfy the counting constraints, consider the following rounding procedure:

1. Assign an integer number of subsystems to each cycle that is close to the original weight, i.e., find integers N_j s.t. $\sum_{j\in[J]} N_j = \sum_{j\in[J]} \|\gamma_j\|_1$ and s.t. N_j is close to $\|\gamma_j\|_1$. This can easily be achieved in a way s.t. $|N_j - \|\gamma_j\|_1| \leq 1$.

2. Find individual integer assignments with total weight N_j that are close to the average assignments, i.e., find integer assignments $\tilde{\gamma}_j$ s.t. $\|\tilde{\gamma}_j\|_1 = N_j$ and s.t. γ_j is close to $\bar{\gamma}_{N_j}$.

   To this end, let κ_1 and κ_2 be the quotient and remainder when dividing N_j by $|C_j|$, i.e. $\kappa_1 = \lfloor N_j/|C_j| \rfloor$ and $\kappa_2 = N_j \mod |C_j|$. Then let $d = |C_j|/\kappa_2$. Consider the pseudo-periodic assignment $\tilde{\gamma}_j$ defined as follows:

   $$\begin{aligned} &\tilde{\gamma}_j(i) = \kappa_1 + 1, \text{ for } i \in \{\lfloor dk \rfloor\}_{k\in[\kappa_2]}, \\ &\tilde{\gamma}_j(i) = \kappa_1, \text{ otherwise.} \end{aligned} \tag{4.31}$$

   This assignment is pseudo-periodic in the sense that the 1's are evenly distributed with distance d before they are rounded to integer points, as illustrated in Figure 4.7. It is easy to see that $\|\tilde{\gamma}_j\|_1 = N_j$.

3. Find a prefix using (4.26) that steers to $\{C_j, \tilde{\gamma}_j\}_{j\in[J]}$.

If the graph is periodic the rounding can be done so as to preserve the parity structure of the original solution and guarantee reachability, but the details are omitted here. Assuming aperiodicity the counting bounds for the rounded solution are compared with the counting bounds for the original solution to obtain a bound on the worst-case counting constraint violation. The first result assumes a certain cycle structure, namely that all nodes that contribute to the counting set X are placed in sequence in the cycle. Such structure is often present as connected counting regions tend to lead to consecutive parts in cycles.

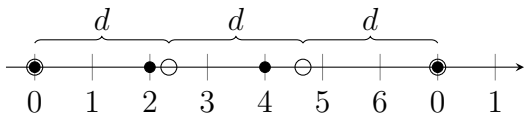

Figure 4.7: Pseudo-periodic assignment rounding. The figure shows how pseudo-periodic assignment rounding is performed for an example with $|C_j| = 7$ and $\|\gamma_j\|_1 = 3$. The subsystems are first placed in non-integer positions separated by a distance d (open circles), and are then rounded into integer positions (filled circles). The final pseudo-periodic assignment is $[1, 0, 1, 0, 1, 0, 0]$.

Proposition 4.4. *Let $R_C^X = \langle C, \mathbf{1} \rangle^X$ be the number of nodes in a cycle C that contribute to X-counting. If all such nodes are consecutive, i.e., $(\xi_i, \mu_i) \in X$ for $i \in [R_C^X]$ and $(\xi_j, \mu_j) \notin X$ for $j \in [|C|] \setminus [R_C^X]$, then the rounding procedure* (4.31) *satisfies*

$$\Psi^X(C_j, \tilde{\gamma}_j) \leq \Psi^X(C_j, \bar{\gamma}_{N_j}) + 1.$$

Proof. When the assignment $\tilde{\gamma}_j$ circulates in C_j, exactly R_C^X contiguous indices of $\tilde{\gamma}_j$ contribute to the X-count. The number of contributing indices with value $\kappa_1 + 1$ (as opposed to value κ_1) can be bounded. Let $[i_0, i_0 + R_C^X - 1]$ be a (circular) sequence representing R_C^X contributing indices. Consider Figure 4.7; any point that ends up in the sequence after left-rounding must satisfy $dk \in [i_0, i_0 + R_C^X)$. Since each left-closed, right-open interval of length d captures exactly one point of the form dk there are at most $\lceil R_C^X/d \rceil$ such points. Therefore by (4.28),

$$\begin{aligned}\Psi^X(C_j, \tilde{\gamma}_j) &\leq \kappa_1 R_C^X + \left\lceil \frac{R_C^X}{d} \right\rceil \leq \kappa_1 R_C^X + \frac{R_C^X}{d} + 1 \\ &\leq \frac{R_C^X}{|C_j|} \underbrace{(\kappa_1 |C_j| + \kappa_2)}_{N_j} + 1 = \Psi^X(C_j, \bar{\gamma}_{N_j}) + 1.\end{aligned}$$

□

Corollary 4.2. *If C_j has at most p X-segments (segments of consecutive nodes in X), the rounding* (4.31) *satisfies*

$$\Psi^X(C_j, \tilde{\gamma}_j) \leq \Psi^X(C_j, \bar{\gamma}_{N_j}) + p.$$

Proof. Follows from applying Proposition 4.4 to each of the p segments. □

These results are now incorporated into a bound on the counting constraint violation in the overall rounding procedure. Again, this bound does not depend on the total number of subsystems N. The difference between the original counting bounds and their relaxed counterparts therefore becomes insignificant as N grows.

Theorem 4.7. *For a given counting constraint (X, R) satisfied by $\{\gamma_j\}_{j\in[J]}$, the following bound holds for the overall suffix rounding procedure*

$$\sum_{j\in[J]} \Psi^X(C_j, \tilde{\gamma}_j) \leq R + J + \sum_{j\in[J]} p_j^X,$$

where p_j^X is the number of X-segments in the j'th cycle. Thus the relaxed counting constraint $(X, R + J + \sum_{j\in[J]} p_j^X)$ is guaranteed to be satisfied by the rounded suffix $\{\tilde{\gamma}_j\}_{j\in[J]}$.

Proof. By the rounding procedure, Corollary 4.2, and (4.29),

$$\Psi^X(C_j, \tilde{\gamma}_j) \leq \Psi^X(C_j, \bar{\gamma}_{N_j}) + p_j^X \leq \Psi^X(C_j, \bar{\gamma}_{\|\gamma_j\|_1}) + 1 + p_j^X \leq \Psi^X(C_j, \gamma_j) + 1 + p_j^X.$$

Summing over $j \in [J]$ yields the result. □

If the structure required in Proposition 4.4 is not present, the following is a worst-case bound on the counting constraint violation due to rounding.

Theorem 4.8. *The rounding* (4.31) *satisfies*

$$\Psi^X(C_j, \tilde{\gamma}_j) \leq \Psi^X\left(C_j, \bar{\gamma}_{N_j}\right) + \frac{|C_j|}{4}.$$

Proof. Assume that $N_j < |C_j|$, since any multiple of $|C_j|$ can be assigned as the exact average assignment. The number of nodes contributing to the X-count is upper bounded by $\min(R_C^X, N_j)$, hence,

$$\begin{aligned}\Psi^X(C_j, \tilde{\gamma}_j) - \Psi^X\left(C_j, \bar{\gamma}_{N_j}\right) &\leq \min(R_C^X, N_j) - \frac{N_j}{|C_j|} R_C^X \\ &= |C_j| \min\left(\frac{R_C^X}{|C_j|}\left(1 - \frac{N_j}{|C_j|}\right), \frac{N_j}{|C_j|}\left(1 - \frac{R_C^X}{|C_j|}\right)\right) \leq \frac{|C_j|}{4},\end{aligned}$$

where the last step follows from $\max_{a,b\in[0,1]} \min(ab, (1-a)(1-b)) = 1/4$. □

These rounding bounds can be precomputed (using all cycles used in the LP instead of all cycles with non-zero assignments) and the counting constraints can be strengthened accordingly to ensure that the original constraints are satisfied after rounding.

4.4. Strong Heterogeneity

Above, mild heterogeneity in the continuous dynamics was considered, which enabled the construction of a single abstraction capturing all possible behaviors. If there is significant heterogeneity in the collection of subsystems this may no longer be possible while maintaining a good approximation.

To alleviate this shortcoming, the synthesis method can be extended to a *multi-class* setting where each subsystem belongs to a particular class, and each class has only mild heterogeneity among its members. One abstraction per class can then be constructed, and the counting problem can be solved jointly for the different classes. In addition, counting constraints can be extended to capture class identity. For instance, it can be posited that at least R_1 subsystems of class 1 be present in a given area, or that no more than R_2 subsystems of class 2 are in a particular mode.

To formalize these ideas, consider DFTSs $\mathcal{T}^m = (\mathbb{X}^m, \mathbb{U}^m, \longrightarrow^m, h^m)$ for $m \in [M]$. Then the multi-class counting problem is as follows:

Problem 4.2. *Consider N subsystems divided in M classes such that class h has N_m members, and $\sum_{m\in[M]} N_m = N$. Subsystems in class h are governed by the the transition system $\mathcal{T}^m$. Assume that for all $m \in [M]$, initial states $\{\mathbf{x}_n^m(0)\}_{n\in[N_m]}$ are given.*

Given L counting constraints $\{\prod_{m\in[M]} X_l^m, R_l\}_{l\in[L]}$ such that $X_l^m \subset \mathbb{X}^m \times \mathbb{U}^m$, synthesize switching protocols $\{\pi_n^m\}_{n\in[N_m]}$ such that the generated inputs $\sigma_n^m(0)\sigma_n^m(1)\sigma_n^m(2)\dots$ and trajectories $\mathbf{x}_n^m(0)\mathbf{x}_n^m(1)\mathbf{x}_n^m(2)\dots$ satisfy the counting constraints

$$\sum_{m\in[M]} \sum_{n\in[N_m]} \mathbb{1}_{X_l^m}(\mathbf{x}_n^m(k), \sigma_n^m(k)) \leq R_l, \quad \forall k \in \mathbb{Z}_+,\ \forall l \in [L].$$

The linear program (4.26) can easily be extended to the multi-class setting, at the cost of additional variables and constraints. Assuming similar abstraction parameters and cycle selections, a problem with two classes has roughly twice as many variables as a problem with a single class. The next section includes an example showcasing how a multi-class counting problem can account for parameter heterogeneity by constructing two abstractions

for a family of continuous-time systems.

4.5. Examples

The method is showcased on two examples; a numerical example and the TCL scheduling problem. The examples are computed with a prototype implementation available at `https://github.com/pettni/mode-count`, which uses Gurobi [50] as the underlying ILP solver.

4.5.1. Numerical Example

The first example is the following nonlinear system:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_1 &= -2(\mathbf{x}_1 - \mathbf{u}) + \mathbf{x}_2, \\ \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_2 &= -(\mathbf{x}_1 - \mathbf{u}) - 2\mathbf{x}_2 - \mathbf{x}_2^3. \end{aligned} \tag{4.32}$$

It can be shown that for a fixed $\mathbf{u}$ the system is incrementally stable and that the $\mathcal{KL}$-function

$$\beta(r,t) = \sqrt{2}r \left\| \exp\left(\begin{bmatrix} -2 & 1 \\ -1 & -2 \end{bmatrix} t \right) \right\|_2$$

satisfies (4.9). Introduce two modes μ_1 and μ_2 corresponding to the fixed inputs $\mathbf{u} \equiv -1$ and $\mathbf{u} \equiv 1$.

For a large N, consider mode-counting constraints $(X_{\mu_1}, 0.55N)$ and $(X_{\mu_2}, 0.55N)$ for $X_{\mu_s} = \mathcal{X}^d \times \{s\}$ stating that at most 55% of the subsystems can use the same dynamical mode at any given time. In addition, consider the balancing objective that no more than 55% of the subsystems can be in either one of the sets $X_1 = \{(x_1, x_2) : x_1 \geq 0\}$ or $X_2 = \{(x_1, x_2) : x_1 \leq 0\}$; expressed by the counting constraints $(X_1, 0.55N)$ and $(X_2, 0.55N)$. Furthermore, all subsystems should repeatedly visit both these sets.

Consider the domain $\mathcal{X}^d = \{(x_1, x_2) : x_1 \in [-2, 2], x_2 \in [-1.5, 1.5]\}$. Using abstraction parameters $\eta = 0.05$, $\tau = 0.32$ an abstraction that is 0.1-approximately bisimilar to the time-discretization of (4.32) can be computed. In order to guarantee that the counting constraints are satisfied, the counting sets must be expanded as $\tilde{X}_1 = \{(x_1, x_2) : x_1 \geq -0.1\}$ and $\tilde{X}_2 = \{(x_1, x_2) : x_1 \leq 0.1\}$. The finite-state counting problem was then solved with randomized initial conditions and a horizon $T = 10$, for a cycle set consisting of 200

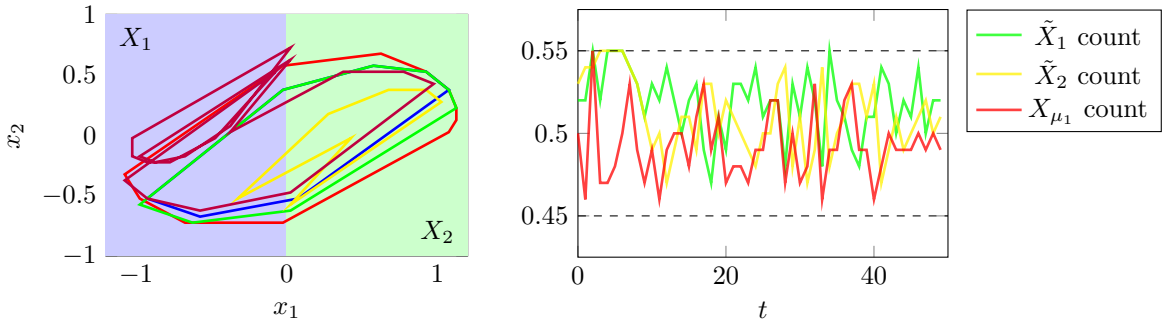

Figure 4.8: Cycles and counting bounds in the numerical example. Left: Illustration of counting sets X_1 and X_2 together with selected cycles making up the suffix solution. Right: Fraction of total number of systems present in $\tilde{X}_1$, $\tilde{X}_2$ and X_{μ_1} over time. Whereas the sets X_{μ_1} and X_{μ_2} are mutually exclusive (the fractions sum to 1), the sets $\tilde{X}_1$ and $\tilde{X}_2$ are not due to the expansion to account for approximate bisimilarity.

randomized cycles. Each cycle was also required to visit both $\tilde{X}_1^C$ and $\tilde{X}_2^C$ in order to achieve the second objective. The problem was solved for $N = 10^k$ for $k = 2, \ldots, 9$; the solving times are shown in Table 4.2 and illustrate that the difficulty of this problem is largely independent of N.

Table 4.2: Average solution times over 10 randomized trials using the Gurobi ILP solver.

N	10^2	10^3	10^4	10^5	10^6	10^7	10^8	10^9
Time (s)	7.8	11.1	12.0	12.8	11.5	13.2	13.0	11.0

Figure 4.8 illustrates some of the cycles that make up the suffix part of the solution and the number of subsystems that are in the counting sets over time in an example trajectory.

4.5.2. Application Example: TCL Scheduling

The following model is used to describe the dynamics of the temperature θ_i of an individual TCL [53]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\theta_i(t) = -a_i\left(\theta_i(t) - \theta_i^a\right) - b_i P_i^m \mathbb{1}_{\{\mathtt{on}\}}\left(\sigma_i(t)\right). \tag{4.33}$$

Consider two distinct populations of TCLs, i.e., that every set of parameters $(a_i, b_i, \theta_i^a, P_i^m)$ is $\overline{\delta}$-close to one of two nominal parameter configurations. Each nominal parameter configuration represents a mildly heterogeneous class (c.f. Section 4.4). The parameter values

Table 4.3: TCL example: parameter values for the two classes. Note that the time discretizations need to be identical for concurrent execution.

Parameter	Class 1	Class 2
Nominal $[a, b, \theta_a, P_m]$	$[2, 2, 32, 5.6]$	$[2.2, 2.2, 32, 5.9]$
Space discretization η	0.002	0.0015
Time discretization τ	0.05	0.05
Error bound $\overline{\delta}$	0.025	0.025

for the nominal configurations are listed in Table 4.3 along with the abstraction parameters η and τ used for each class and the allowed deviation $\overline{\delta}$ from these nominal values.

It can easily be shown that the $\mathcal{KL}$-function

$$\beta(r, s) = re^{-sa_i}.$$

satisfies (4.9) with respect to (4.33). In addition, it satisfies the approximate bisimulation inequality (4.13) for an approximation level $\epsilon = 0.2$ and the Lipschitz constant a_i for (4.33); thus the results from Section 4.1.2 apply. The constraints for the TCL problem have been introduced earlier in (4.4)-(4.5). All subsystems were constrained to remain in the temperature interval $[21.3, 23.7]$; taking the approximation into account this implies that the corresponding constraint for the discrete problem became $\theta_i(t) \in [21.5, 23.5]$.

Initial conditions were randomly selected for 10,000 systems of each class and 50 random cycles were sampled for each class[22], before (4.26) was solved for a prefix length of 20 steps (corresponding to one hour). Randomized additive model errors δ_i such that $|\delta_i| \leq \overline{\delta}$ were also introduced to represent mild in-class heterogeneity. Figure 4.9 shows simulated trajectory densities for two different mode-`on`-counts, one *maximal* count of 6,000 and one *minimal* count of 6,700. For comparison, the analytical minimal upper bound is 5,522 and the analytical maximal lower bound is 7,118 as computed from formulas in [92]. These bounds are not attained here due to approximation errors stemming from the approximate bisimulation, incomplete cycle selection, etc. Figure 4.10 shows mode-`on`-counts during the same simulations. As can be seen, the imposed counting bounds are satisfied.

Remark 4.4. *It would be straightforward to also introduce between-class heterogeneity in the counting constraints. Such heterogeneity could be used to model power consumption*

[22] To promote diversity in the cycle sets, the cycles were selected in order to have different fractions of time in mode `on`.

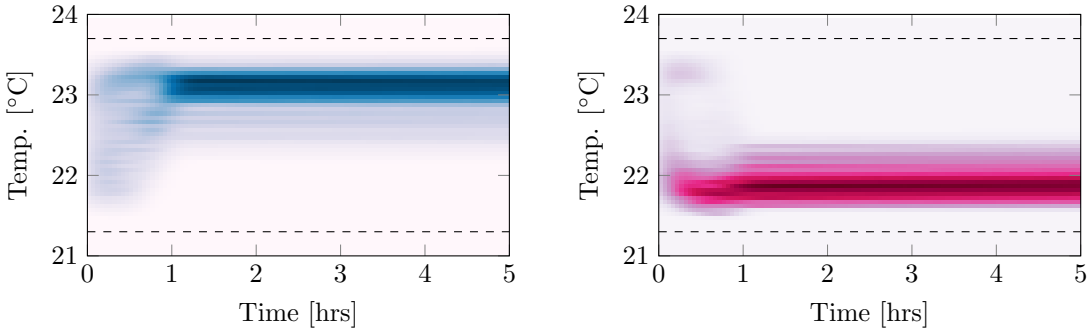

Figure 4.9: Density of TCLs in different parts of the temperature spectrum over time. Left: maximal mode-on-count of 6,000. Right: minimal mode-on-count of 6,700. Colored parts represent regions occupied by a larger fraction of the 20,000 subsystems. The state counting constraint (4.5) guarantees that no subsystem exits the interval [21.3, 23.7] which is marked with dashed lines. The first hour represents the prefix part of the solution which steers the initial state to the periodic suffix. As can be seen, for a low desired mode-on-count, the subsystems group in the upper end of the temperature spectrum. The reason is that the difference between inside and outside temperature is lower, so the TCLs can have a shorter average duty cycle since less warming takes place.

that varies between classes and thus allow a more realistic power consumption constraint to be enforced.

4.6. Summary

This chapter was concerned with control synthesis for high-dimensional but permutation-symmetric systems, subject to likewise symmetric counting constraints. The symmetry enables representation of the aggregate system as an integer linear system defined on top of an abstraction constructed from a single subsystem, thus avoiding abstraction of the whole state space. Prefix-suffix solutions were characterized as the feasible set of an integer linear program, and it was shown how to interpret (in)feasibility of the program both in the integer and non-integer case.

As demonstrated, the method can solve problem instances with tens of thousands of states. With the proposed approach, it is possible to impose hard constraints on the overall power consumption of TCLs over an infinite time horizon. Yet, the converse results suggest that "worst-case" discrete-state counting problems become more difficult (requiring

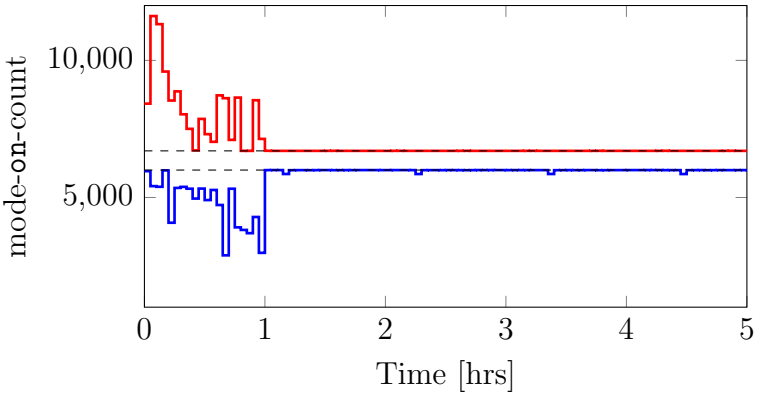

Figure 4.10: Number of TCLs in mode on during the two simulations. As can be seen, the lower bound of 6,700 is enforced for the upper (red) trajectory, while the upper bound 6,000 is enforced for the lower (blue) trajectory.

long horizons and large cycle sets) as N grows. However, the increased difficulty is the result of certain modularity properties that are unlikely to appear in an abstraction graph.

In this work cycles and cycle assignments were used to form the suffix part of a solution. An alternative approach, explored in [115] which focuses on applications to multi-agent planning and coordination, is to introduce binary loop variables that ensure that $\mathbf{w}(k) = \mathbf{w}(T)$ for some $k \leq T$. That approach does not require sampling of a cycle set, and also enables more sophisticated temporal counting specifications (cLTL+) via ILP encodings of LTL, but has the drawback that the relaxed (non-integer) LP lacks a clear interpretation. In addition, the set of cycles in this work can be selected in a way to satisfy certain auxiliary specifications, as highlighted in Section 4.5.1.

The setting in this chapter was synchronous and deterministic; uncertainty on the continuous level was "abstracted away", resulting in a deterministic discrete system. Relaxing these requirements would make the method more applicable in practice, although non-determinism on the discrete level makes the problem significantly more challenging. Future work also includes identifying ways to round the prefix part of a non-integer solution, which would enable approximate solutions of counting problems without the need of an ILP solver.

Exploiting symmetries in other types of systems to achieve scalability in correct-by-construction methods is a promising direction for future work. Another direction is to consider other types of abstractions, including non-deterministic ones, since not all systems admit finite bisimulations. Although the idea of an aggregate system can still be used in

this case, one should either solve a robust uncertain ILP, which could lead to conservative results, or consider reactive feedback solutions, for which different synthesis techniques are required.

Chapter 5.

Summary and Outlook

The value of formal methods and correct-by-construction techniques ultimately hinges on their ability to improve system design. An improvement could either be a shortening the time of development by eliminating faulty behaviors at an earlier stage, or in the form of a "better" final result. What is better varies from case to case, but most systems have high performance and safety requirements, and must be robust to uncertainties and model errors. It is also advantageous if the system is built modularly and if the development process is relatively fast. System development choices are determined by how different actors value these and other objectives and there are typically trade-offs involved: an autonomous car that does not move is perfectly safe, but not a viable product. Formal methods have the potential to achieve better trade-offs in many cases:

- Increased performance without compromising safety. A correct-by-construction approach yields an exact demarcation of the envelope of safe conditions—the winning set. Trajectory optimization can be done within this set to select the best safe trajectory which may be more efficient than a conservative approach. The winning set can also be utilized as a supervisor that corrects unsafe control actions of a performance controller (see e.g. [86]).

- Better understanding of robustness: explicit treatment of specifications and environment models gives a clear picture of the conditions under which the controller can operate safely.

- Modularity via contracts: if subsystems are endowed with assume-guarantee contracts, one subsystem can be replaced with another subsystem satisfying the same contract, without compromising overall system integrity.

- Shortening the development cycle via earlier detection and elimination of design errors.

Despite these potential advantages, formal methods have yet to be widely adopted in industry. Arguably, the two main barriers are scalability and modeling. This thesis addressed the scalability issue and suggested different ways of exploiting problem structure to mitigate the computational burden. The other issue—modeling—is also related to scalability: high-fidelity models are too complex to be amenable to formal methods. It is therefore crucial to find a model that captures all relevant dynamics in a concise form. Motivated by these current shortcomings, below three important directions for future research are discussed.

5.1. Sophisticated Uncertainty Models

Autonomous systems require sophisticated estimation algorithms that provide controllers with accurate state estimates. Increasing the reliability of real systems via formal methods will require integrated efforts both on the estimation side and on the control side of the problem. Such a holistic perspective is to date fairly uncommon in the literature but nevertheless important: more accurate disturbance models may simultaneously increase both safety and performance. Safety will be improved if uncertainties are better understood, and performance can be increased as there is less need for conservatism to counter unmodeled uncertainties. Better disturbance models will also serve to further delineate the environment where the system can safely operate.

In most of the literature on formal methods, if disturbance is at all included it is assumed—like in this thesis—to be confined to a bounded time-invariant set. While this may be appropriate to capture modeling inaccuracies, the most important uncertainties in real systems often take a radically distinct form. Autonomous cars will for instance rely heavily on neural networks that are notoriously difficult to analyze. It is obvious that the "disturbance"—i.e., risk of misclassification—stemming from a neural network is of a different kind: instead of small, nondeterministic, and potentially persistent, it is large, probabilistic[23], and (hopefully) short-lived. Similar types of short-lived disturbance can be found in other problems such as the Lane Keeping problem from Section 3.3.2 where the

[23] An argument could be made whether the probabilistic "risk of misclassification" that is usually employed in the machine learning community is the appropriate measure. Systems such as autonomous cars that are deployed on a massive scale will require a probability of error that is practically zero.

curvature of the road is modeled as disturbance. In that model, the road curvature was assumed to be contained in a box $\{r_d : -0.05 \leq r_d \leq 0.05\}$. However, in reality curves do not go on forever. By modeling the disturbance as short-lived in time the maximal curvature the controller can handle will be larger; i.e., the controller becomes less conservative.

One potential avenue towards formal treatment of large but short-lived disturbances is to use ideas related to the problem of maximizing "time of invariance" [91]. Briefly, if the system ends up in an extraordinary situation where invariance can not be guaranteed over an infinite horizon, the control objective is switched to maximization of the time of invariance. If this "time of invariance" can be extended (with guarantees) up until the moment normal operation resumes, the safety of the system is preserved. In the cited paper ideas from optimal control were leveraged to obtain lower bounds on the achievable time of invariance, together with associated control policies.

5.2. Improvements in Set Computation and Representation

The backbone of synthesis algorithms is the backwards reachability operator—if it can be computed accurately and cheaply any synthesis problem can be solved. Unfortunately, this is usually not the case—even for linear systems the computations become numerically challenging for systems of moderate dimensionality.

Several potential ways to improve scalability for linear systems were given in Section 3.4. From a more general point of view, it may be necessary to sacrifice exactness for scalability. A backwards reachability operator that inner-approximates the true backwards reachable set can still be used to obtain correct—but not maximal—controllers. Approximations that are not contained in the true set (i.e., are not *inner* approximations) will yield false positives in the sense that the specification can not be enforced from certain points inside the approximate "winning set"[24]. While this obviously moves outside of the realm of correct-by-construction, if the numerical efficiency is increased to the extent that otherwise intractable problems become tractable, the resulting "approximate guarantees" may be better than no guarantees at all. For instance, recent advances in sparse functional approximation have been used to dramatically improve scalability in synthesis [5, 49]. In

[24] If the approximation error can be bounded, it can often be compensated for to obtain an inner approximation.

essence, those papers propose a novel way of sparse set representation which enhances the efficiency of reachability computations. This concept applies in the continuous as well as in the finite domain. Part of ongoing work is to explore Binary Decision Diagrams (BDDs) as parsimonious alternatives for representing a finite abstraction. Preliminary results indicate that backwards reachability calculations can be performed faster when using BDDs than when employing traditional sparse matrix representations. In addition, the choice of method to encoding states in the BDD may have a significant effect.

Finally, the division of on-line versus off-line computational requirements needs to be kept in mind when it comes to set computation and representation. The term synthesis—or correct-by-construction—is generally understood as an off-line synthesis step that results in an easy-to-evaluate policy that can be implemented on-line at a high loop frequency. The other extreme is motion planning, where all computation is done on-line at a point when the initial state is known. In the work on counting problems in Chapter 4 the proposed policies are more reminiscent of motion planning than synthesis: a description of a winning set is obtained, but checking whether a state is inside the set requires solving an ILP. Thus, as of now, the proposed method is better suited for systems running at a low control loop frequency. Moving more of the computation off-line will require (approximate) reachability computations for the aggregate dynamics. On the contrary, the discussion in Section 3.4 suggested that a difficult off-line problem (polyhedral projection) can potentially be avoided at the cost of an easier on-line problem (linear programming). The on-line/off-line computational trade-off needs to be adjusted to the problem at hand.

5.3. Additional Structural Properties

The work in this thesis can be summarized as exploitations of problem structure to improve scalability. In Chapter 2 the problem structure was used to refine the state space only in certain areas; in Chapter 3 the compositional nature exhibited by many problems was utilized; and Chapter 4 treated problems with permutation symmetry in both dynamics and specification. The negative results regarding the difficulty of the synthesis problem imply that there is no hope to encounter an algorithm capable of solving the general control synthesis problem. Therefore, to further advance the reach of formal methods, all potential advantages stemming from structural properties must be leveraged. A prominent recent example of work along these lines is [63] where monotonicity properties in both dynamics

and specification were exploited to ease the computational burden. Fortunately, structure and symmetries are omnipresent in real-life systems and there are likely many important discoveries yet to be made.

Appendix A.

Supplements to Chapter 2

A.1. Synthesis Algorithm Proofs

Lemma A.1. *The first component of* (2.10) *is sound and complete.*

Proof. Let $\{X_k\}_{k\geq 0}$ be the sequence of sets generated in the fixed-point computation; it is evident that $X_1 = Z_0$. Let X_∞ be the resulting fixed point.

For soundness, it suffices to show that a trajectory starting in $\xi \in X_{k+1} \setminus X_k$ can be steered to X_k in finite time while remaining in $[\![b]\!]$. There are two cases. First, if $\xi \in [\![b]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}(X_k) \setminus X_k$, then X_k can be reached in one time step by definition of $\mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}$. Secondly, if $\xi \in \mathrm{PGPre}^{\mathcal{T}^+}_{\sharp,\forall}(X_k, [\![b]\!]) \setminus X_k$, then there exists (G, U) with $G \in \mathcal{G}(U)$ such that X_k can be reached by keeping the state inside of $G \cap [\![b]\!]$ using actions in U until a transition to X_k occurs due to the progress property (if $\sharp = \forall$, then $U = \mathbb{U}$ which enables progress for uncontrollable modes). This shows that $X_{k+1} \subset \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(b \,\mathbf{U}\, X_k)$. By induction the soundness of the algorithm follows.

Completeness follows if it is shown that if a state ξ is not in X_∞, it is not in the $(\sharp, \forall)$-winning set of $b \,\mathbf{U}\, Z$ either. Assume for contradiction that ξ is in the $(\sharp, \forall)$-winning set of $b \,\mathbf{U}\, Z$; that is, from ξ there exists a control strategy such that Z is reached in finitely many steps during which $[\![b]\!]$ is kept invariant. This can happen in two different ways. First, there exists a bound K on the number of steps within which Z is guaranteed to be reached. In this case, ξ is in $\tilde{X}_K$ for $\tilde{X}_0 = Z$, $\tilde{X}_{k+1} = [\![b]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}(\tilde{X}_k)$, which contradicts the fact that X_∞ is a fixed point that does not contain ξ due to the $[\![b]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}(X_k)$ term. Secondly, Z is guaranteed to be reached from ξ while remaining in $[\![b]\!]$ but no control strategy can guarantee a bound on the number of steps. In this case, the controlled trajectories are

not guaranteed to avoid a progress group. Take ξ to be in the last progress group G for some action set U before reaching X_∞. This is without loss of generality, as otherwise the prefix part can be handled by the arguments in the first case and inductively applying the arguments for the second case. But then $\xi \in \mathrm{Inv}^{U,G}_{\sharp,\forall}(X_\infty, [\![b]\!]) \subset \mathrm{PGPre}^{\mathcal{G}}_{\sharp,\forall}(X_\infty, [\![b]\!])$, which again contradicts to the fact that X_∞ is a fixed point. □

Lemma A.2. *The second component of* (2.10) *is sound and complete.*

Proof. The following two LTL identities will be used below:

$$\Box\left(\psi_1 \wedge \left(\bigwedge_{i\in I} \Diamond\psi_2^i\right)\right) = \Box\psi_1 \wedge \left(\bigwedge_{i\in I} \Box\left(\mathtt{True}\ \mathbf{U}\ \psi_2^i\right)\right) = \Box\left[\bigwedge_{i\in I}\left[\psi_1\ \mathbf{U}\ \left(\psi_1 \wedge \psi_2^i\right)\right]\right], \quad \text{(A.1)}$$

$$(\psi_3\ \mathbf{U}\ \psi_1) \vee (\psi_3\ \mathbf{U}\ \psi_2) = \psi_3\ \mathbf{U}\ (\psi_1 \vee \psi_2). \quad \text{(A.2)}$$

Denote the specification by φ_1, i.e., $\varphi_1 = (b\ \mathbf{U}\ Z) \vee \Box\left(b \wedge \left(\bigwedge_{i\in I} \Diamond c^i\right)\right)$. A specification ψ_1 is said to be *stronger* than ψ_2 if for any trajectory ζ, $\zeta \models \psi_1 \implies \zeta \models \psi_2$. If ψ_1 is stronger than ψ_2, then $\mathrm{Win}(\psi_1) \subset \mathrm{Win}(\psi_2)$.

To prove soundness, remark that any fixed point $\overline{W}$ satisfies

$$\overline{W} = \bigcap_{i\in I} \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(b\ \mathbf{U}\ \left(Z \cup \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}\left(\overline{W}\right)\right)\right)\right).$$

Consider $\xi \in \overline{W}$, a trajectory starting in ξ can for each i be controlled to reach either Z or $[\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}\left(\overline{W}\right)$ while remaining in $[\![b]\!]$. If the former occurs, $[\![b]\!]\ \mathbf{U}\ Z$, and thus φ_1, is evidently satisfied by the trajectory. If the latter occurs, the set $[\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}\left(\overline{W}\right)$ is eventually reached. Because of the $\mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}\left(\overline{W}\right)$ term, the argument repeats which shows that either $[\![b]\!]\ \mathbf{U}\ Z$ or

$$\Box\left([\![b]\!]\ \mathbf{U}\ \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}\left(\overline{W}\right)\right)\right) = \Box\left([\![b]\!] \wedge \Diamond\left([\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}(\overline{W})\right)\right)$$

is satisfied by the trajectory (the equality follows from (A.1)). This holds for any i, which is a stronger condition than φ_1. Thus the algorithm is sound.

Completeness amounts to showing that $\mathrm{Win}\left(\varphi_1\right) \subset W_\infty$. Consider the specification

$$\varphi_2 = \bigwedge_{i\in I}\left[b\ \mathbf{U}\ \left(Z \cup \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}\left(\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\varphi_1\right)\right)\right)\right)\right].$$

Since φ_1 is a liveness specification, a trajectory satisfying φ_1 must remain in $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\varphi_1)$. Therefore the winning set of φ_1 is equal to the winning set of the specification

$$(b \,\mathbf{U}\, Z) \vee \Box \left(b \wedge \left(\bigwedge_{i \in I} \Diamond \left([\![c^i]\!] \cap \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \right) \right) \right).$$

Using (A.1) above, this is in turn equal to

$$(b \,\mathbf{U}\, Z) \vee \Box \left(\bigwedge_{i \in I} \left[b \,\mathbf{U}\, \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \right) \right] \right),$$

which is stronger than

$$\begin{aligned} (b \,\mathbf{U}\, Z) \vee & \left(\bigwedge_{i \in I} \left[b \,\mathbf{U}\, \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \right) \right] \right) \\ & = \bigwedge_{i \in I} \left[(b \,\mathbf{U}\, Z) \vee \left(b \,\mathbf{U}\, \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \right) \right) \right]. \end{aligned}$$

By (A.2) and since $[\![b]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall} \left(\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \right) = [\![b]\!] \cap \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1)$, this is stronger than φ_2. Thus $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \subset \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_2)$.

Take any set K such that $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \subset K$. The specification φ_2 is stronger than the specification

$$\varphi_K = \bigwedge_{i \in I} \left[b \,\mathbf{U}\, \left(Z \cup \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall} (K) \right) \right) \right],$$

which implies that $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_2) \subset \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_K)$. Furthermore, in general $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\psi_1 \wedge \psi_2)$ is contained in $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\psi_1) \cap \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\psi_2)$, which implies

$$\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_K) \subset T_K = \bigcap_{i \in I} \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} \left(b \,\mathbf{U}\, \left(Z \cup \left([\![b]\!] \cap [\![c^i]\!] \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall} (K) \right) \right) \right).$$

Thus, if $\xi \in \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1)$, then the inclusions $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \subset \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_2) \subset \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_K)$ imply that ξ is never excluded during the algorithm since it consists of iterating the mapping $K \mapsto T_K$. Thus, $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall} (\varphi_1) \subset W_\infty$, which proves completeness. □

Lemma A.3. *The third component of* (2.10) *is sound and complete.*

Proof. Let $\{V_k\}_{k \geq 0}$ be the sequence of sets generated in the fixed-point computation. Let

V_∞ be the resulting fixed point.

First soundness and completeness are shown for $\Box a = \texttt{True}$. Let $\psi = \Diamond\Box b \wedge \left(\bigwedge_{i\in I} \Box\Diamond c^i\right)$. Starting with soundness; $\Box\left(b \wedge \left(\bigwedge_{i\in I} \Diamond c^i\right)\right)$ is stronger than ψ, therefore it follows that $V_1 \subset \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi)$. Consider V_k for $k > 1$. Starting anywhere in V_k, the state can either be controlled to satisfy $b \,\mathbf{U}\, \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}(V_{k-1})$ or $\Box\left(b \wedge \left(\bigwedge_{i\in I} \Diamond c^i\right)\right)$. If the former happens, the state can be controlled to V_{k-1} and an induction argument over k completes the soundness proof.

For completeness, consider the set $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi) \setminus V_\infty$, where V_∞ is the smallest fixed point. From any $\xi_1 \in \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi) \setminus V_\infty$, the state can *not* be controlled to V_∞, otherwise ξ_1 would have been included in V_∞ by an argument analogous to the completeness proof of Lemma A.1. By repeating the argument it follows that an infinite trajectory in $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi) \setminus V_\infty$ can be generated that satisfies the specification ψ. By considering an appropriate suffix of such a trajectory, this implies the existence of $\xi_2 \in \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi) \setminus V_\infty$ from where the specification $\Box\left(b \wedge \left(\bigwedge_{i\in I} \Diamond c^i\right)\right)$ can be enforced. But this is a contradiction since such a ξ_2 is in V_1 and hence in V_∞.

The final step is to incorporate the $\Box a$ term and show necessity and sufficiency of the restriction to V_{inv}. As established above, the algorithm is sound and complete for ψ. All fixed points are ultimately defined in terms of the winning set of $B \,\mathbf{U}\, Z$, so amending $\Box a$ to the top level specification ψ propagates that term down to the *until* level. It is therefore necessary and sufficient to replace each computation of $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(b \,\mathbf{U}\, Z)$ with $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\Box a \wedge (b \,\mathbf{U}\, Z))$. Since $V_{inv} = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\Box a)$, it follows that

$$\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\Box a \wedge (b \,\mathbf{U}\, Z)) = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}((\llbracket b \rrbracket \cap V_{inv}) \,\mathbf{U}\, (Z \cap V_{inv}))$$

which shows the correctness of the restriction technique. □

Lemma A.4. *The first component of* (2.11) *is sound and complete.*

Proof. For soundness, consider a set of fixed points $\overline{X}^J$ for $J \in 2^I$ that satisfy

$$\overline{X}^J = Z \cup \left(\left(\bigcap_{i\in J} \llbracket b^i \rrbracket\right) \cap \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}\left(\bigcup_{K\in 2^J} \overline{X}^K\right)\right).$$

Let $\xi \in \overline{X}^J$. If $\xi \in Z$ the specification is evidently satisfied. Otherwise, $\xi \in \left(\bigcap_{i\in J} \llbracket b^i \rrbracket\right) \cap \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}\left(\bigcup_{K\in 2^J} \overline{X}^K\right)$ which implies that a trajectory starting in ξ can be controlled to

$\overline{X}^{K_1}$ for some $K_1 \subset J$. If Z is not reached, an induction argument results in a chain $J \supset K_1 \subset K_2 \supset \ldots$ which necessarily converges to some non-empty subset K_∞ of J. Thus there is a strategy that eventually enforces $\Box b^i$ for some $i \in J$.

Let $\{X_k^J\}_{k\geq 0}$ be the sequence of sets generated in the fixed-point computation. For completeness, remark that $X_1^{\{i\}} = Z \cup [\![b^i]\!]$. The objective is to show that if $\xi_0 \notin X_{k_0+1}^J$ but $\xi_0 \in X_{k_0}^J$, then nondeterminism can force a trajectory starting in ξ_0 to avoid $X_1^{\{i\}}$ for all $i \in I$ while also avoiding Z. To this end, assume that $\xi_0 \in X_{k_0}^J \setminus X_{k_0+1}^J$. Evidently, $\xi_0 \notin Z$, which implies that $\xi_0 \in \bigcap_{i\in J}[\![b^i]\!]$ and $\xi_0 \notin \mathrm{Pre}_{\sharp,\forall}^{\mathcal{T}^+,\mathbb{U}}\left(\bigcup_{K\in 2^J} X_{k_0}^K\right)$. The latter means that nondeterminism can prevent a transition to $X_{k_0}^K$ for any $K \subset J$. Assume a transition to ξ_1 occurs and that ξ_1 was excluded from $\{X_k^K\}$ for $k = k_1$. Induction over time and set inclusion results in a strictly decreasing sequence $k_0 k_1, \ldots$ and a trajectory $\zeta(0)\zeta(1)\ldots$ where $\zeta(t) \notin \bigcup_{K\in 2^J} X_{k_t}^K$. In particular, $\zeta(t) \notin Z$ and there exists a finite T s.t. $\zeta(T) \notin [\![b^i]\!]$ for any $i \in J$, which shows completeness of the algorithm. □

Lemma A.5. *The second component of* (2.11) *is sound and complete.*

Proof. Let $\{W_k\}_{k\geq 0}$ be the sequence of sets generated in the fixed-point computation. Let W_∞ be the resulting fixed point.

For soundness, remark that $W_1 = \mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}\left(\bigvee_{i\in I}[(b^i \,\mathbf{U}\, Z) \vee \Box b^i]\right)$ which is contained in $\mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}\left(\Diamond Z \vee \left(\bigvee_{i\in I}\Diamond\Box b^i\right)\right)$. Starting in W_k for $k > 1$, a strategy exists that either results in $\Box b^i$ being fulfilled for some i, or such that W_k can be reached. An induction argument completes the soundness proof.

For completeness, assume that $\xi \in \mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}\left(\Diamond Z \vee \left(\bigvee_{i\in I}\Diamond\Box b^i\right)\right) \setminus W_\infty$. Since

$$\mathrm{Pre}_{\sharp,\forall}^{\mathcal{T}^+}(W_\infty) \cup \mathrm{PGPre}_{\sharp,\forall}^{\mathcal{T}^+}(W_\infty, \mathbb{X}) \subset W_\infty,$$

it follows by the completeness argument in the proof of Lemma A.1 that a transition to W_∞ can never be enforced for a trajectory starting in ξ. Since $Z \subset W_\infty$, it follows that nondeterminism can generate a trajectory that never enters W_∞ and which satisfies $\Diamond\Box b^i$ for some i. Taking a suffix of such a trajectory, it follows that there exists $\xi' \in \mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}\left(\Diamond Z \vee \left(\bigvee_{i\in I}\Diamond\Box b^i\right)\right) \setminus W_\infty$ from where $\bigvee_{i\in I}\Box b^i$ can be enforced. But then $\xi' \in W_1 \subset W_\infty$ which is a contradiction. □

Lemma A.6. *The third component of* (2.11) *is sound and complete.*

Proof. Let $\psi = \Diamond a \vee \left(\bigvee_{i\in I}\Diamond\Box b^i\right) \vee \Box\Diamond c$. For soundness, consider a fixed point $\overline{V}$. It has

the property

$$\overline{V} = \mathrm{Win}_{\sharp,\forall}\left(\Diamond\left(\llbracket a\rrbracket \cup \left(\llbracket c\rrbracket \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}\left(\overline{V}\right)\right)\right) \vee \left(\bigvee_{i\in I}\Diamond\Box b^i\right)\right).$$

Starting in $\overline{V}$, there is a strategy to ensure one of $\psi_1 = \Diamond a$, $\psi_2 = \bigvee_{i\in I}\Diamond\Box b^i$, or $\psi_3 = \Diamond\left(\llbracket c\rrbracket \cap \mathrm{Pre}^{\mathcal{T}^+}_{\sharp,\forall}\left(\overline{V}\right)\right)$. If ψ_1 or ψ_2 occur, ψ is evidently satisfied. If ψ_3 occurs, an induction argument shows that $\psi_1 \vee \psi_2 \vee \Box\psi_3 = \psi$ holds.

For completeness, remark that a trajectory that enforces ψ must remain in $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi)$ due to ψ being a liveness property. Furthermore, $\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}\left(\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi)\right) = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi)$ since ψ can be enforced from its pre-image. Therefore,

$$\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi) = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\tilde{\psi}\right), \text{ for } \tilde{\psi} = \Diamond a \vee \left(\bigvee_{i\in I}\Diamond\Box b^i\right) \vee \Box\Diamond\left(\llbracket c\rrbracket \cap \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}\left(\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi)\right)\right).$$

Take any $K \supset \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi)$ and let $\psi_K = \Diamond a \vee \left(\bigvee_{i\in I}\Diamond\Box b^i\right) \vee \Diamond\left(\llbracket c\rrbracket \cap \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\forall}(K)\right)$. It can be seen that $\tilde{\psi}$ is stronger than ψ_K, which implies the inclusion $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi) \subset \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi_K)$. This shows that elements in $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi)$ are never excluded, which consist of iterations of the mapping $K \mapsto \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\psi_K)$. □

A.2. Candidate Set Derivation

This section outlines how a candidate set C of a winning set $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\cdot)$ is recursively determined. Assuming that $\mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\cdot)$ is computed as the fixed point V_∞ of a sequence $\{V_k\}_{k\geq 1}$, C is constructed according to the following rules:

- For an expanding algorithm with first non-trivial set V_1 and fixed point V_∞:

  Addition: add $\mathrm{Pre}_{\sharp,\exists}(V_\infty) \setminus V_\infty$ to the candidate set C.

  Recursion: if V_1 is an output of another fixed-point algorithm, add its candidate set to C.

- For a contracting algorithm with first non-trivial set V_1 and fixed point V_∞:

  Addition: add $V_1 \setminus V_\infty$ to the candidate set C.

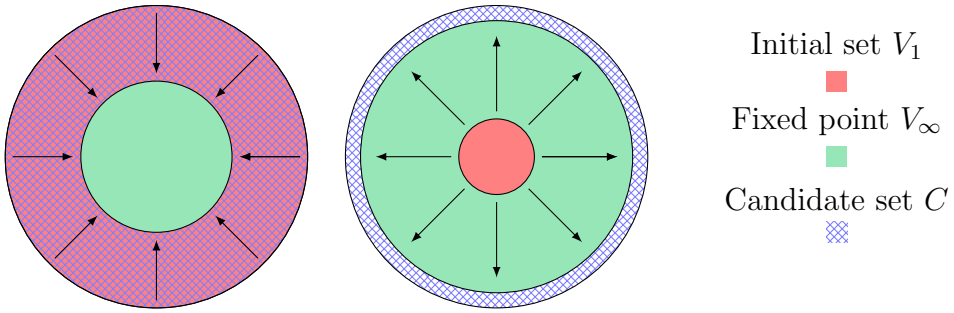

Figure A.1: Candidate sets for contracting and expanding algorithms. For a contracting algorithm (left), the candidate set is selected as the difference between the initial set and the final fixed point. For an expanding algorithm, the candidate set consists of states that are adjacent to the final fixed point but that were not included in the fixed-point computation.

Recursion: if V_1 is an output of another fixed-point algorithm, add its candidate set to C.

The rationales behind these rules are illustrated in Figure A.1. Firstly, the fixed point may always be enlarged by starting with a larger initial set, thus this object is pursued in both cases. For an expanding algorithm, also states that are adjacent to its fixed point V_∞ are added in the hope that a refinement may reveal states that can enlarge it further. There is no need to consider smaller sets V_i since these are already contained in V_∞. For a contracting algorithm, $V_1 \setminus V_\infty$ is added in the hope that refinement may reveal control options that allow the fixed point V_∞ to be enlarged.

There are ways to further tune the candidate sets that may be suitable for certain problems. Firstly, the progress group reachability operator PGPre is computed with a contracting algorithm whose candidate set could be added to the overall candidate set. Below this potential addition is disregarded in the interest of keeping the notation relatively simple; the best way to implement candidate set computation algorithmically is to follow the recursive rules above. Secondly, it may be possible to exclude parts of the candidate set of a contracting algorithm in case there are states that will for sure be excluded by the algorithm even after refinement.

The rules above are now applied to the computation of the winning set (2.10) to obtain a candidate set of a specification of type $\Box a \wedge \Diamond\Box b \wedge \left(\bigwedge_{i\in I} \Box\Diamond c^i\right)$. The algorithm is expanding and produces an increasing set sequence $\{V_k\}_{k\geq 1}$. The first step is therefore the following:

1A. Addition: add $\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}(V_\infty) \setminus V_\infty$ to the candidate set,

1R. Recursion: add candidate set of $V_1 = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\Box\left(b \wedge \left(\bigwedge_{i\in I} \Diamond c^i\right)\right)\right)$ to the candidate set.

Then recursively consider V_1, which is computed as the stable point W_∞ of a contracting set sequence $\{W_k\}_{k\geq 1}$.

2A. Addition: add $W_1 \setminus W_\infty = W_1 \setminus V_1$ to the candidate set,

2R. Recursion: add candidate set of $W_1 = \bigcap_{i\in I} W_{1,i}$ for $W_{1,i} = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(b \,\mathbf{U}\, (b \wedge c^i)\right)$ to the candidate set.

The computation proceeds with the different $W_{1,i}$'s; each computed via an expanding set sequence $\{X_{k,i}\}_{k\geq 1}$. Since $X_{1,i} = [\![b]\!] \cap [\![c^i]\!]$ is not itself a fixed point, the recursion stops here and the final step is:

3A. Addition: add $[\![b]\!] \cap \left(\bigcup_{i\in I} \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}(W_{1,i}) \setminus W_{1,i}\right)$ to the candidate set. Here the result is intersected with $[\![b]\!]$ since states in $[\![b]\!]^C$ can not be candidates to $b \,\mathbf{U}\, (b \wedge c^i)$.

However, there is one final part of the overall candidate set, since the winning set computation is restricted to $V_{inv} = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}(\Box a)$. This set is computed via a contracting set sequence $\{\tilde{W}_k\}_{k\geq 1}$ with $\tilde{W}_1 = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(a \,\mathbf{U}\, a\right) = [\![a]\!]$. Therefore one final addition to the candidate set is made:

4A. Addition: add $[\![a]\!] \setminus V_{inv}$ to the candidate set.

Collecting the different pieces from above results in the following candidate set:

$$\begin{aligned} C_{\sharp,\forall}\left(\Box a \wedge \Diamond\Box b \wedge \left(\bigwedge_{i\in I} \Box\Diamond c^i\right)\right) &= \left(\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}(V_\infty) \setminus V_\infty\right) \cup (W_1 \setminus V_1) \\ &\cup \left([\![b]\!] \cap \left(\bigcup_{i\in I} \mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}(W_{1,i}) \setminus W_{1,i}\right)\right) \cup ([\![a]\!] \setminus V_{inv})\,. \end{aligned} \tag{A.3}$$

A candidate set for the dual algorithm (2.11) can be derived in a similar way. Let $\{V_k\}_{k\geq 1}$ be the contracting set sequence generated by (2.11).

1A. Addition: add $V_1 \setminus V_\infty$ to the candidate set,

1R. Recursion: add candidate set of $V_1 = \mathrm{Win}^{\mathcal{T}^+}_{\sharp,\forall}\left(\Diamond\left([\![a]\!] \cup [\![c]\!]\right) \vee \left(\bigvee_{i\in I} \Diamond\Box b^i\right)\right)$ to the candidate set.

Proceeding with V_1, it is equal to the fixed point value W_∞ of an expanding sequence $\{W_k\}_{k\geq 1}$.

2A. Addition: add $\mathrm{Pre}_{\sharp,\exists}^{\mathcal{T}^+,\mathbb{U}}(W_\infty) \setminus W_\infty = \mathrm{Pre}_{\sharp,\exists}^{\mathcal{T}^+,\mathbb{U}}(V_1) \setminus V_1$ to the candidate set,

2R. Recursion: add candidate set of $W_1 = \mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}\left(\bigvee_{i\in I}\left[\Box b_i \vee (b_i \,\mathbf{U}\, (a \vee c))\right]\right)$ to the candidate set.

Finally, W_1 is equal to the union over the set of fixed points X_∞^J. At the first iteration, $X_1^J = [\![a]\!] \cup [\![c]\!] \cup \left(\bigcap_{i\in J}[\![b^i]\!]\right)$. The final addition to the candidate set is thus:

2A. Addition: add $\bigcup_{J\in 2^I}[\![a]\!] \cup [\![c]\!] \cup \left(\bigcap_{i\in J}[\![b^i]\!]\right) \setminus W_1$ to the candidate set. However, it holds that $\bigcup_{J\in 2^I}\bigcap_{i\in J}[\![b^i]\!] = \bigcup_{i\in I}[\![b^i]\!]$ and furthermore that $[\![a]\!] \cup [\![c]\!] \in W_1$. Therefore the additional set simplifies to $\bigcup_{i\in I}[\![b^i]\!] \setminus W_1$.

Combined, the result is the following candidate set:

$$C_{\sharp,\forall}\left(\Diamond a \bigvee_{i\in I} \Diamond\Box b^i \vee \Box\Diamond c\right) = (V_1 \setminus V_\infty) \cup \left(\mathrm{Pre}_{\sharp,\exists}^{\mathcal{T}^+,\mathbb{U}}(V_1) \setminus V_1\right) \cup \left(\bigcup_{i\in I}[\![b^i]\!] \setminus W_1\right). \quad \text{(A.4)}$$

Next, two notable special cases of (A.3) are considered for illustration purposes. First, for a specification of the form $\Diamond\Box b$, the expression simplifies to

$$C_{\sharp,\forall}(\Diamond\Box b) = \left(\mathrm{Pre}_{\sharp,\exists}^{\mathcal{T}^+,\mathbb{U}}(V_\infty) \setminus V_\infty\right) \cup ([\![b]\!] \setminus V_1),$$

for $V_\infty = \mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}(\Diamond\Box b)$ and $V_1 = \mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}(\Box b)$. The same expression can also be obtained from (A.4).

Secondly, for a specification of the form $\bigwedge_{i\in I}\Box\Diamond c^i$, (A.3) gives that

$$C_{\sharp,\forall}\left(\bigwedge_{i\in I}\Box\Diamond c^i\right) = \left(\mathrm{Pre}_{\sharp,\exists}^{\mathcal{T}^+,\mathbb{U}}(V_\infty) \setminus V_\infty\right) \cup \left(\bigcap_{i\in I} W_{1,i} \setminus V_\infty\right) \cup \left(\bigcup_{i\in I}\mathrm{Pre}_{\sharp,\exists}^{\mathcal{T}^+,\mathbb{U}}(W_{1,i}) \setminus W_{1,i}\right)$$

where $V_\infty = \mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}\left(\bigwedge_{i\in I}\Box\Diamond c^i\right)$ and $W_{1,i} = \mathrm{Win}_{\sharp,\forall}^{\mathcal{T}^+}(\Diamond c^i)$. However, the expression can be

simplified further since the first term is contained in the union of the last two[25]:

$$C_{\sharp,\forall}\left(\bigwedge_{i\in I}\Box\Diamond c^i\right)=\left(\bigcap_{i\in I}W_{1,i}\setminus V_\infty\right)\cup\left(\bigcup_{i\in I}\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}\left(W_{1,i}\right)\setminus W_{1,i}\right).$$

Again, the same expression can be obtained from (A.4) for the special case $I=\{1\}$.

[25] Assume that $\xi\in\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}\left(V_\infty\right)\setminus V_\infty$. Since $V_\infty\subset W_{1,i}$ for all i, it follows that $\xi\in\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}\left(V_\infty\right)\subset\bigcap_{i\in I}\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}\left(W_{1,i}\right)$. Assume $\xi\notin\mathrm{Pre}^{\mathcal{T}^+,\mathbb{U}}_{\sharp,\exists}\left(W_{1,i}\right)\setminus W_{1,i}$ for any $i\in I$, then evidently $\xi\in W_{1,i}$.

Appendix B.

Supplements to Chapter 3

B.1. Proof of Theorem 3.1

The results below depend on manipulation of positive definite matrices, in particular congruency transforms and Schur complements. The following can be found in e.g [22].

Lemma B.1 (Congruency transform)**.** *Let C be a non-singular matrix. Then $X \succ 0$ if and only if $CXC^T \succ 0$.*

Lemma B.2 (Schur complement)**.** *Let $X = \left[\begin{smallmatrix} A & B \\ B^T & D \end{smallmatrix}\right]$. Then*

$$X \succ 0 \iff D \succ 0,\ A - BD^{-1}B^T \succ 0,$$
$$X \succ 0 \iff A \succ 0,\ C - B^TA^{-1}B \succ 0.$$

First three lemmas are presented. First, recall the following result from [125] which is stated without proof.

Lemma B.3. *Let R, Z be symmetric matrices, and let A and B be full matrices. Then the following conditions are equivalent*

$$\begin{bmatrix} R & AB \\ * & Z \end{bmatrix} \succ 0, \tag{B.1a}$$
$$\exists\, X = X^T \ : \begin{bmatrix} R & A \\ * & X^{-1} \end{bmatrix} \succ 0,\ \begin{bmatrix} X & B \\ * & Z \end{bmatrix} \succ 0. \tag{B.1b}$$

The utility of this lemma is to separate the bilinear matrix product AB in the upper right entry of (B.1a). However, the introduced matrix variable X appears both as itself

and its inverse. The next two lemmas are inspired by the same paper, but the formulation has been slightly altered.

Lemma B.4. *The following two statements are equivalent*

$$\begin{bmatrix} C^T X C & Y \\ * & Z \end{bmatrix} \succ 0, \tag{B.2a}$$

$$C \text{ is non-singular}, \; X \succ 0, \; \exists \Psi : \begin{bmatrix} C\Psi + \Psi^T C^T - X^{-1} & \Psi^T Y \\ * & Z \end{bmatrix} \succ 0 \tag{B.2b}$$

Proof. If it can be shown that the following two statements are equivalent:

$$\begin{bmatrix} X & Y \\ * & Z \end{bmatrix} \succ 0, \tag{B.3a}$$

$$X \succ 0, \; \exists \Psi : \begin{bmatrix} \Psi + \Psi^T - X^{-1} & \Psi^T Y \\ * & Z \end{bmatrix} \succ 0, \tag{B.3b}$$

then the result follows by replacing $X \mapsto C^T X C$ in (B.3b), applying the congruency transform $\left[\begin{smallmatrix} C & 0 \\ 0 & I \end{smallmatrix}\right]$, and re-defining $C\Psi^T \to \Psi^T$.

(B.3a) $\to$ (B.3b): First note that (B.3a) implies that $X \succ 0$ by a Schur complement argument. By a congruency transform, (B.3a) is equivalent to

$$\forall \; \Psi \text{ non-singular}, \quad \begin{bmatrix} \Psi^T X \Psi & \Psi^T Y \\ * & Z \end{bmatrix} \succ 0. \tag{B.4}$$

Since X is invertible, $\Psi^T X \Psi = \Psi^T + \Psi - X^{-1} + (\Psi - X)^T X^{-1} (\Psi - X)$. Substituting this expression in (B.4) and choosing Ψ as the (non-singular) matrix X which eliminates the last term shows that there exists Ψ such that (B.3b) holds.

(B.3b) $\to$ (B.3a): : Assuming that (B.3b) holds, the positive term $(\Psi - X)^T X^{-1} (\Psi - X)$ can be "added back" to the top left entry to obtain that

$$\exists \Psi \text{ s.t. } \begin{bmatrix} \Psi^T X \Psi & \Psi^T Y \\ * & Z \end{bmatrix} \succ 0. \tag{B.5}$$

A contradiction argument shows that such a Ψ is non-singular, so Ψ can be eliminated by a congruency transform. □

Lemma B.5. *If there exist a full matrix Θ, and symmetric matrices Γ and Ξ such that*

$$\Delta = \begin{bmatrix} \Gamma & Y \\ * & \Xi \end{bmatrix} \succ 0, \quad \begin{bmatrix} Z + \Xi & \begin{bmatrix} -X & I \end{bmatrix} \Theta & V \\ * & \Theta + \Theta^T - \Delta & 0 \\ * & * & W \end{bmatrix} \succ 0, \tag{B.6}$$

then

$$\begin{bmatrix} Z + XY + Y^T X^T & V \\ * & W \end{bmatrix} \succ 0. \tag{B.7}$$

Remark B.1. *As is evident from the proof, if $X\Gamma X^T$ is added to the top left entry of the right hand matrix in* (B.6)*, the reverse implication also holds.*

Proof. Γ and Δ^{-1} are positive definite, so adding the (possibly semi-)positive definite terms $X\Gamma X^T$ to the top left block, and adding $(\Theta - \Delta)^T \Delta^{-1} (\Theta - \Delta)$ to the middle block of (B.6) preserves positive definiteness. Using the identity $\Theta^T \Delta^{-1} \Theta = \Theta + \Theta^T - \Delta + (\Theta - \Delta)^T \Delta^{-1} (\Theta - \Delta)$ then implies that

$$\begin{bmatrix} Z + \Xi + X\Gamma X^T & \begin{bmatrix} -X & I \end{bmatrix} \Theta & V \\ * & \Theta^T \Delta^{-1} \Theta & 0 \\ * & * & W \end{bmatrix} \succ 0.$$

A contradiction argument shows non-singularity of Θ. The congruency transform $\left[\begin{smallmatrix} I & 0 & 0 \\ 0 & 0 & I \\ 0 & \Theta^{-T} & 0 \end{smallmatrix}\right]$ then gives

$$\begin{bmatrix} Z + \Xi + X\Gamma X^T & V & \begin{bmatrix} -X & I \end{bmatrix} \\ * & W & 0 \\ * & * & \Delta^{-1} \end{bmatrix} \succ 0.$$

Applying a Schur complement finally implies that

$$0 \prec \begin{bmatrix} Z + \Xi + X\Gamma X^T & V \\ * & W \end{bmatrix} - \begin{bmatrix} -X & I \\ 0 & 0 \end{bmatrix} \Delta \begin{bmatrix} -X^T & 0 \\ I & 0 \end{bmatrix} = \begin{bmatrix} Z + XY + Y^T X^T & V \\ * & W \end{bmatrix}.$$

□

Proof of Theorem 3.1. The proof verifies that the satisfaction of the LMIs given in the theorem statement guarantee controlled invariance, satisfaction of state constraints, and

satisfaction of input constraints, namely that

$$\forall x \in \prod_{n\in[N]} \mathcal{C}_n, \forall d \in \mathcal{D}, \quad Ax + B\hat{K}H_x x + Ed \in \prod_{n\in[N]} \mathcal{C}_n, \tag{B.8}$$

$$\forall x_n \in \mathcal{C}_n, \quad x_n \in \mathcal{X}_n, \tag{B.9}$$

$$\forall x \in \prod_{n\in[N]} \mathcal{C}_n, \quad \hat{K}_n H_x x \in \mathcal{U}_n. \tag{B.10}$$

Start with (B.8) and let $A_K = A + B\hat{K}H_x$ and $\mathcal{C} = \prod_{n\in[N]} \mathcal{C}_n$. Because of symmetry, $x \in \mathcal{C}$ if and only if $-x \in \mathcal{C}$, and similarly for $\mathcal{D}$. Therefore it follows that (B.8) holds if and only if

$$e_j^T Z H_x (A_K x + Ed) - 1 \leq 0 \tag{B.11}$$

for all $x \in \mathcal{C}$, $d \in \mathcal{D}$ and for all $j \in [\mathcal{N}_x]$. Furthermore note that $x \in \mathcal{C}$ if and only if for all diagonal $D_x \succ 0$

$$(\mathbf{1} - ZH_x x)^T D_x (\mathbf{1} + ZH_x x) \geq 0. \tag{B.12}$$

Similarly, $d \in \mathcal{D}$ if and only if for all diagonal $D_d \succ 0$

$$(\mathbf{1} - H_d d)^T D_d (\mathbf{1} + H_d d) \geq 0. \tag{B.13}$$

The next step is to employ the S Procedure. To prepare for this, express the left hand side of (B.11) in terms of the quadratic forms in (B.12)-(B.13) and an additional quadratic term:

$$\begin{aligned} e_j^T Z H_x A_K x + e_j^T Z H_x E d - 1 = &-(\mathbf{1} - ZH_x x)^T \tilde{D}_x^j (\mathbf{1} + ZH_x x) \\ &- (\mathbf{1} - ZH_d d)^T \tilde{D}_d^j (\mathbf{1} + ZH_d d) - \begin{bmatrix} x^T & d^T & 1 \end{bmatrix} L_x^j(\tilde{D}_x^j, \tilde{D}_d^j) \begin{bmatrix} x^T & d^T & 1 \end{bmatrix}^T, \end{aligned}$$

for $\tilde{D}_x^j \succ 0$ (diagonal), $\tilde{D}_d^j \succ 0$ (diagonal) and

$$L_x^j(\tilde{D}_x^j, \tilde{D}_d^j) = \begin{bmatrix} H_x^T Z^T \tilde{D}_x^j Z H_x & 0 & -\frac{1}{2} A_K^T H_x^T Z^T e_j \\ * & H_d^T \tilde{D}_d^j H_d & -\frac{1}{2} E^T H_x^T Z^T e_j \\ * & * & 1 - \mathbf{1}^T \tilde{D}_x^j \mathbf{1} - \mathbf{1}^T \tilde{D}_d^j \mathbf{1} \end{bmatrix}.$$

An application of the S procedure [103] shows that (B.8) holds if and only if for all $j \in [\mathcal{N}_x]$ there exist diagonal $\tilde{D}_x^j \succ 0$ and $\tilde{D}_d^j \succ 0$ such that $L_x^j(\tilde{D}_x^j, \tilde{D}_d^j) \succ 0$.

Next, Lemma B.3 implies that (B.8) holds if and only if for all $j \in [\mathcal{N}_x]$ there exist diagonal $\tilde{D}_x^j \succ 0$ and $\tilde{D}_d^j \succ 0$, and a symmetric Φ_j such that

$$M_1 = \begin{bmatrix} \begin{bmatrix} H_x^T Z^T \tilde{D}_x^j Z H_x & 0 \\ * & H_d^T \tilde{D}_d^j H_d \end{bmatrix} & \begin{bmatrix} -\frac{1}{2} A_K^T & 0 \\ 0 & -\frac{1}{2} E^T \end{bmatrix} \\ * & \begin{bmatrix} \tilde{\Phi}_j^{-1} \end{bmatrix} \end{bmatrix} \succ 0, \tag{B.14}$$

$$M_2 = \begin{bmatrix} \tilde{\Phi}_j & \begin{bmatrix} H_x^T Z^T e_j \\ H_x^T Z^T e_j \end{bmatrix} \\ * & 1 - \mathbf{1}^T \tilde{D}_x^j \mathbf{1} - \mathbf{1}^T \tilde{D}_d^j \mathbf{1} \end{bmatrix} \succ 0. \tag{B.15}$$

In the remainder of the proof, these two matrix inequalities are turned into LMIs.

Treatment of M_2. Let $\bar{H}_x = \left[\begin{smallmatrix} H_x & 0 \\ 0 & H_x \end{smallmatrix}\right]$ and apply the congruency transform $\left[\begin{smallmatrix} \bar{H}_x^{-T} & 0 \\ 0 & I \end{smallmatrix}\right]$ on M_2 to obtain the equivalent condition

$$\begin{bmatrix} \bar{H}_x^{-T} \tilde{\Phi}_j \bar{H}_x^{-1} & \begin{bmatrix} Z^T e_j \\ Z^T e_j \end{bmatrix} \\ * & 1 - \mathbf{1}^T \tilde{D}_x^j \mathbf{1} - \mathbf{1}^T \tilde{D}_d^j \mathbf{1} \end{bmatrix} \succ 0. \tag{B.16}$$

Multiply the matrix in (B.16) with a scalar $\lambda_{n(j)} > 0$, where $n(j)$ is the index of the subsystem corresponding to the jth inequality in Z, and redefine $\Phi_j = \lambda_{n(j)} \tilde{\Phi}_j$, $D_x^j = \lambda_{n(j)} \tilde{D}_x^j$, $D_d^j = \lambda_{n(j)} \tilde{D}_d^j$. Note that since Z^T is block diagonal, it holds for Λ as defined in (3.12a) that $\Lambda Z^T e_j = \lambda_{n(j)} Z^T e_j$. This results in

$$\begin{bmatrix} \bar{H}_x^{-T} \Phi_j \bar{H}_x^{-1} & \begin{bmatrix} \Lambda Z^T e_j \\ \Lambda Z^T e_j \end{bmatrix} \\ * & \lambda_j - \mathbf{1}^T D_x^j \mathbf{1} - \mathbf{1}^T D_d^j \mathbf{1} \end{bmatrix} \succ 0.$$

For $\bar{\Lambda} = \left[\begin{smallmatrix} \Lambda & 0 \\ 0 & \Lambda \end{smallmatrix}\right]$ apply the congruency transform $\left[\begin{smallmatrix} \bar{\Lambda}^{-T} & 0 \\ 0 & I \end{smallmatrix}\right]$:

$$\begin{bmatrix} \bar{\Lambda}^{-T} \bar{H}_x^{-T} \Phi_j \bar{H}_x^{-1} \Lambda^{-1} & \begin{bmatrix} Z^T e_j \\ Z^T e_j \end{bmatrix} \\ * & \lambda_j - \mathbf{1}^T D_x^j \mathbf{1} - \mathbf{1}^T D_d^j \mathbf{1} \end{bmatrix} \succ 0.$$

Now apply Lemma B.4 which implies that (B.15) holds if and only if there exist a symmetric Φ_j and full matrix Ψ_j such that

$$\begin{bmatrix} \Psi_j^T \bar{\Lambda}^{-T} \bar{H}_x^{-T} + \bar{H}_x^{-1} \bar{\Lambda}^{-1} \Psi_j - \Phi_j^{-1} & \Psi_j^T \begin{bmatrix} Z^T e_j \\ Z^T e_j \end{bmatrix} \\ * & 1 - \mathbf{1}^T D_x^j \mathbf{1} - \mathbf{1}^T D_d^j \mathbf{1} \end{bmatrix} \succ 0.$$

Finally, Lemma B.5 gives the *necessary* condition

$$\Delta_j = \begin{bmatrix} \Gamma_j & \Psi_j \\ * & \Xi_j \end{bmatrix} \succ 0, \begin{bmatrix} -\tilde{\Phi}_j^{-1} + \Xi_j & \begin{bmatrix} -\bar{H}^{-1} \bar{\Lambda}^{-1} & I \end{bmatrix} \Theta_j & \Psi_j^T \begin{bmatrix} Z^T e_j \\ Z^T e_j \end{bmatrix} \\ * & \Theta_j + \Theta_j^T - \Delta_j & 0 \\ * & * & 1 - \mathbf{1}^T D_x^j \mathbf{1} - \mathbf{1}^T D_d^j \mathbf{1} \end{bmatrix} \succ 0.$$

By further restricting to $\Theta_j = \begin{bmatrix} \bar{\Lambda} & \bar{\Lambda} \\ \Omega_j^1 & \Omega_j^2 \end{bmatrix}$, the LMIs in (3.12b)-(3.12c) are obtained.

Treatment of M_1. Apply the congruency transform $\begin{bmatrix} \lambda_{n(j)} H_x^{-T} & 00 & \\ 0 & \lambda_{n(j)} H_d^{-T} & 0 \\ 0 & 0 & I \lambda_{n(j)}^{-1} \end{bmatrix}$ to obtain

$$\begin{bmatrix} \begin{bmatrix} Z^T \lambda_{n(j)} \tilde{D}_x^j Z & 0 \\ * & \lambda_{n(j)} \tilde{D}_d^j \end{bmatrix} & \begin{bmatrix} -\frac{1}{2} H_x^{-T} A_K^T & 0 \\ 0 & -\frac{1}{2} H_d^{-T} E^T \end{bmatrix} \\ * & \lambda_{n(j)}^{-1} \tilde{\Phi}_j^{-1} \end{bmatrix} \succ 0.$$

Note that $H_x^{-T} A_K^T = (A H_x^{-1} + B \hat{K})^T$. Thus the same re-definitions as above for $\tilde{D}_x^j$, $\tilde{D}_d^j$ and $\tilde{\Phi}_j$ result in the LMI (3.12d).

As a next step, find an expression that ensures (B.9). As before, write for $k \in [\mathcal{N}_s]$

$$e_k^T (H_s x - h_s) = -(\mathbf{1} - Z H_x x)^T D_s^k (\mathbf{1} + Z H_x x) - \begin{bmatrix} x^T & 1 \end{bmatrix} L_s^k(D_s^k) \begin{bmatrix} x^T & 1 \end{bmatrix}^T,$$

for

$$L_s^k(D_s^k) = \begin{bmatrix} H_x^T Z^T D_s^k Z H_x & -\frac{1}{2} H_s^T e_k \\ * & e_k^T h_s - \mathbf{1}^T D_s^k \mathbf{1} \end{bmatrix}.$$

The S procedure applies in the same way as before and the congruency transform $\begin{bmatrix} H_x^{-T} & 0 \\ 0 & I \end{bmatrix}$ gives the LMI (3.12e) which represents necessary and sufficient conditions for (B.9).

Finally, the input constraints are handled similarly; that is, condition (B.10) is satisfied if and only if for all $l \in [\mathcal{N}_u]$,

$$e_l^T \left(H_u K x - h_u\right) \leq 0$$

for all $x \in \mathcal{C}$. By the S procedure this can be translated into positive definiteness of the matrix

$$\begin{bmatrix} H_x^T Z^T D_u^l Z H_x & -\frac{1}{2} K^T H_u^T e_l \\ * & e_l^T h_u - \mathbf{1}^T D_u^l \mathbf{1} \end{bmatrix} \succ 0.$$

Applying the congruency transform $\begin{bmatrix} H_x^{-T} & 0 \\ 0 & I \end{bmatrix}$ gives (3.12f) for the same $\hat{K}$ as before. □

B.2. Proof of Theorem 3.5

The result is obtained by quantifier elimination via two techniques: substitution of a "for all" condition by its robust counterpart, and restriction to affine dependence as a way to turn a $\forall\exists$ quantification into a $\exists\forall$ quantification. The following lemma describes the first technique.

Lemma B.6. *The following equivalences hold for $a \in \mathbb{R}^{n_\zeta}$, $H \in \mathbb{R}^{\mathcal{N}_\zeta \times n_\zeta}$, and $h \in \mathbb{R}^{\mathcal{N}_\zeta}$:*

$$\forall \zeta : H\zeta \leq h,\ a^T\zeta \leq b \ \ \textit{iff}\ \ \exists y \in \mathbb{R}^{\mathcal{N}_\zeta} : h^T y \leq b,\ H^T y = a,\ y \geq 0. \tag{B.17a}$$

$$\forall \zeta : H\zeta \leq h, a^T\zeta = b \ \ \textit{iff}\ \ \exists y_1, y_2 \in \mathbb{R}^{\mathcal{N}_\zeta} : \begin{cases} h^T y_1 \leq b,\ H^T y_1 = a,\ y_1 \geq 0, \\ h^T y_2 \leq -b,\ H^T y_2 = -a,\ y_2 \geq 0. \end{cases} \tag{B.17b}$$

Proof. Start with the first equivalence. The left-hand side of (B.17a) is satisfied if and only if

$$\sup_{\zeta : H\zeta \leq h} \zeta^T a \leq b.$$

Introducing the dual variable y, an equivalent formulation of the left-hand optimization problem is

$$(P): \quad \sup_{\zeta \in \mathbb{R}^{n_\zeta}} \inf_{y \in \mathbb{R}_+^{\mathcal{N}_\zeta}} \zeta^T a + y^T \left(h - H\zeta\right).$$

Its dual is

$$(D): \quad \inf_{y \in \mathbb{R}_+^{\mathcal{N}_\zeta}} \sup_{\zeta \in \mathbb{R}^{n_\zeta}} \zeta^T \left(a - H^T y\right) + h^T y.$$

Equivalently,

$$(D): \quad \begin{array}{ll} \inf\limits_{y \in \mathbb{R}_+^{\mathcal{N}_\zeta}} & h^T y \\ \text{s.t} & a = H^T y \end{array}$$

Assuming Slater's condition, strong duality holds, meaning that the optimal values of (P) and (D) coincide. For the second equivalence, remark that the left hand side of (B.17b) holds if and only if both $a^T\zeta \leq b$ and $-a^T\zeta \leq -b$ for all $\zeta : H\zeta \leq h$. □

Corollary B.1. *The following is equivalent for $A \in \mathbb{R}^{\mathcal{N}\times n_\zeta}$, $H \in \mathbb{R}^{\mathcal{N}_\zeta \times n_\zeta}$ vectors b, h of appropriate sizes;*

$$\forall \zeta : H\zeta \leq h,\ A\zeta \leq b \iff \exists y \in \mathbb{R}^{\mathcal{N}_\zeta \times \mathcal{N}} : h^T y \leq b^T,\ H^T y = A^T,\ y \geq 0.$$

Example B.1. *This result can be used to verify invariance of a set $\{x : H_x x \leq h_x\}$ for the dynamics $\mathbf{x}(k+1) = A\mathbf{x}(k)$. Invariance holds if and only if*

$$\exists y \in \mathbb{R}^{\mathcal{N}_x \times \mathcal{N}_x} : h_x^T y \leq h_x^T,\ H_x^T y = A^T H_x^T,\ y \geq 0,$$

which is a well-known result from set-theoretic control [20, Theorem 4.41].

Corollary B.2. *The following is equivalent for $B, A_j \in \mathbb{R}^{\mathcal{N}_1 \times \mathcal{N}_2}$, $H \in \mathbb{R}^{\mathcal{N}_\zeta \times n_\zeta}$;*

$$\forall \zeta : H\zeta \leq h,\ \sum_{j=1}^{n_\zeta} \zeta_j A_j \leq B \iff \forall i = 1, \ldots, \mathcal{N}_2\ \exists y_i \in \mathbb{R}^{\mathcal{N}_\zeta \times \mathcal{N}_1} : \begin{cases} h^T y_i \leq e_i^T B^T, \\ H^T y_i = \begin{bmatrix} e_i^T A_1^T \\ \vdots \\ e_i^T A_{n_\zeta}^T \end{bmatrix}, \\ y_i \geq 0. \end{cases}$$

Proof of Theorem 3.5. The goal is to inner approximate the set

$$\left\{ \begin{array}{l} x :\ \forall d^1 \in \mathcal{D}_1\ \exists u \in \mathcal{U}\ \forall d^2 \in \mathcal{D}_2, \\ \quad H_x \left(\left(A + \sum\limits_{j \in [n_{d^1}]} d_j^1 A_j^1 + \sum\limits_{j \in [n_{d^2}]} d_j^2 A_j^2 \right) x + Bu + \sum\limits_{j \in [n_{d^1}]} d_j^1 F_j^1 + \sum\limits_{j \in [n_{d^2}]} d_j^2 F_j^2 \right) \leq h_x \end{array} \right\},$$

which is the backwards reachable set when the disturbance $d^1 \in \mathcal{D}_1$ is measurable and

$d^2 \in \mathcal{D}_2$ is non-measurable. For $i = 1, 2$

$$H_x \sum_{j \in [n_{d^i}]} d_j^i A_j^i x = \underbrace{\begin{bmatrix} H_x A_0^i & \cdots & H_x A_{n_d^i - 1}^i \end{bmatrix}}_{H_A^i} \left(I_{n_d^i} \otimes x \right) d^i,$$

$$H_x \sum_{j \in [n_{d^i}]} d_j^i F_j^i = \underbrace{\begin{bmatrix} H_x F_0^i & \cdots & H_x F_{n_{d^i} - 1}^i \end{bmatrix}}_{F_A^i} d^i.$$

An application of Corollary B.1 results in the equivalent description

$$\left\{ \begin{array}{l} x : \forall d^1 \in \mathcal{D}_1 \; \exists u \in \mathcal{U} \; \exists y \in \mathbb{R}^{\mathcal{N}_{d^2} \times \mathcal{N}_x}, \\ \quad h_{d^2}^T y \leq \left(h_x - H_x A x - H_A^1 d^1 - H_x B u - H_F^1 d^1 \right)^T, \\ \quad H_{d^2}^T y = (H_A^2 (I_{n_d^2} \otimes x) + H_F^2)^T, \; y \geq 0. \end{array} \right\}$$

Accepting some loss of generality, restrict attention to affine dependence of u and y on d^1, and obtain for $u = u_0 + K_u d^1$ and $y = y_0 + \sum_{j \in [n_{d^1}]} K_y^j d_j^1$ the inner approximation

$$\left\{ \begin{array}{l} x : \exists y_0 \in \mathbb{R}^{\mathcal{N}_{d^2} \times \mathcal{N}_x}, \{K_y^j\}_{j \in [n_{d^1}]} \in \mathbb{R}^{\mathcal{N}_{d^2} \times \mathcal{N}_x}, u_0 \in \mathbb{R}^{n_u}, K_u \in \mathbb{R}^{n_u \times n_{d^1}} \; \forall d^1 \in \mathcal{D}_1, \\ \quad h_{d^2}^T \left(y_0 + \sum_{j \in [n_{d^1}]} K_y^j d_j^1 \right) \leq \left(h_x - H_x A x - H_A^1 d^1 - H_x B u_0 - H_x B K_u d^1 - H_F^1 d^1 \right)^T, \\ \quad H_{d^2}^T \left(y_0 + \sum_{j \in [n_{d^1}]} K_y^j d_j^1 \right) = (H_A^2 (I_{n_d^2} \otimes x) + H_F^2)^T, \; y_0 + \sum_{j \in [n_{d^1}]} K_y^j d_j^1 \geq 0, \\ \quad H_u \left(u_0 + K_u d^1 \right) \leq h_u. \end{array} \right\}$$

By writing

$$h_{d^2}^T \sum_{j \in [n_{d^1}]} K_y^j d_j^1 = (d^1)^T \begin{bmatrix} h_{d^2}^T K_y^0 \\ \vdots \\ h_{d^2}^T K_y^{n_{d^1} - 1} \end{bmatrix},$$

taking transposes, and collecting the terms with d^1 on the left-hand side one arrives at

$$\left\{\begin{array}{l} x : \exists y_0 \in \mathbb{R}^{\mathcal{N}_{d^2}\times\mathcal{N}_x}, \{K_y^j\}_{j\in[n_{d^1}]} \in \mathbb{R}^{\mathcal{N}_{d^2}\times\mathcal{N}_x}, u_0 \in \mathbb{R}^{n_u}, K_u \in \mathbb{R}^{n_u\times n_{d^1}}\ \forall d^1 \in \mathcal{D}_1, \\ \quad \left(\left[(K_y^0)^T h_{d^2} \quad \cdots \quad (K_y^{n_{d^1}-1})^T h_{d^2}\right] + H_A^1(I_{n_d^1}\otimes x) + H_F^1 + H_x B K_u\right) d^1 \\ \quad\quad \leq h_x - H_x A x - H_x B u_0 - y_0^T h_{d^2}, \\ \quad \sum\limits_{j\in[n_{d^1}]} d_j^1 (K_y^j)^T H_{d^2} = H_A^2(I_{n_d^2}\otimes x) + H_F^2 - y_0^T H_{d^2}, \\ \quad \sum\limits_{j\in[n_{d^1}]} d_j^1 (-K_y^j)^T \leq y_0^T,\ H_u K_u d^1 \leq h_u - H_u u_0. \end{array}\right\}$$

Now Corollary B.1 and B.2 can again be applied to each (in)equality to get the formulation in the theorem statement. □

Appendix C.

Supplements to Chapter 4

Proof of Theorem 4.4. First consider the case $k \leq T$. It will be shown that the selection on Line 2 of Algorithm 3 is possible if

$$\mathbf{w}_\xi(k) = \sum_{n\in[N]} \mathbb{1}_{\{\xi\}}\left(\zeta_n(k)\right), \quad \forall \xi \in V, \tag{C.1}$$

and, furthermore, that the selection guarantees that (C.1) holds at time $(k+1)$.

Due to (4.25), equation (C.1) holds at $k = 0$. For induction, assume that (C.1) holds at time k. Then by (4.26e),

$$\sum_{\mu\in\mathbb{U}} \mathbf{r}_\xi^\mu(k) = \mathbf{w}_\xi(k) = \sum_{n\in[N]} \mathbb{1}_{\{\xi\}}\left(\zeta_n(k)\right). \tag{C.2}$$

The selection on line 2 amounts to for each $\xi \in V$ assigning $\mathbf{w}_\xi(k)$ objects to $|\mathbb{U}|$ "bins" such that the μ-bin has $\mathbf{r}_\xi^\mu(k)$ members; by (C.2) this is doable. Remark that if $\sigma_n(k) = \mu$, then $\zeta_n(k+1) = \xi$ if and only if $\zeta_n(k) \in \mathcal{N}_\xi^\mu$. Thus,

$$\begin{aligned}\sum_{n\in[N]} \mathbb{1}_{\{\xi\}}\left(\zeta_n(k+1)\right) &= \sum_{n\in[N]}\sum_{\mu\in\mathbb{U}} \mathbb{1}_{\{(\xi,\mu)\}}\left(\zeta_n(k+1), \sigma_n(k)\right) \\ &= \sum_{n\in[N]}\sum_{\mu\in\mathbb{U}}\sum_{\xi'\in\mathcal{N}_\xi^\mu} \mathbb{1}_{\{(\xi',\mu)\}}\left(\zeta_n(k), \sigma_n(k)\right) = \sum_{\mu\in\mathbb{U}}\sum_{\xi'\in\mathcal{N}_\xi^\mu} \mathbf{r}_{\xi'}^\mu(k) = \mathbf{w}_\xi(k+1),\end{aligned}$$

where the last step follows from (4.18). Thus (C.1) holds for all $k \in [T+1]$.

Secondly, consider the case $k \geq T$. It will be shown that the selection on line 4 is possible

if for all $\xi \in V$

$$\sum_{\mu\in\mathbb{U}} \sum_{j\in[J]} \left\langle C_j, \gamma_j^{\circlearrowleft k-T} \right\rangle^{\{(\xi,\mu)\}} = \sum_{n\in[N]} \mathbb{1}_{\{\xi\}} \left(\zeta_n(k)\right), \tag{C.3}$$

and, furthermore, that the selection guarantees that (C.3) holds at time $(k+1)$. To show that (C.3) enables a selection $\{\sigma_n(k)\}_{n\in[N]}$ satisfying line 4, it suffices to remark that the selection problem is equivalent to above with $\mathbf{r}_\xi^\mu(k)$ replaced by $\sum_{j\in[J]} \left\langle C_j, \gamma_j^{\circlearrowleft k-T} \right\rangle^{\{(\xi,\mu)\}}$.

Due to (4.26d) and (C.1), equation (C.3) holds at $k = T$. Suppose for induction that (C.3) holds at time $k \geq T$ and that a selection $\{\sigma_n(k)\}_{n\in[N]}$ satisfying line 4 has been made. Then,

$$\begin{aligned}
&\sum_{n\in[N]} \mathbb{1}_{\{\xi\}} \left(\zeta_n(k+1)\right) = \sum_{n\in[N]} \sum_{\mu\in\mathbb{U}} \mathbb{1}_{\{(\xi,\mu)\}} \left(\zeta_n(k+1), \sigma_n(k)\right) \\
&= \sum_{n\in[N]} \sum_{\mu\in\mathbb{U}} \sum_{\xi'\in\mathcal{N}_\xi^\mu} \mathbb{1}_{\{(\xi',\mu)\}} \left(\zeta_n(k), \sigma_n(k)\right) \\
&= \sum_{j\in[J]} \sum_{\mu\in\mathbb{U}} \left\langle C_j, \gamma_j^{\circlearrowleft k-T} \right\rangle^{\mathcal{N}_\xi^\mu\times\{\mu\}} = \sum_{j\in[J]} \sum_{\mu\in\mathbb{U}} \left\langle C_j, \gamma_j^{\circlearrowleft k+1-T} \right\rangle^{\{(\xi,\mu)\}}.
\end{aligned}$$

The last step follows from the observation that a node in $\mathcal{N}_\xi^\mu$ must have a cycle index $(i-1) \mod |C_j|$ in cycle C_j, where i is the cycle index of ξ. Thus the selection on line 4 is feasible for all $k \geq T$.

Finally, it is shown that each counting constraint (X_l, R_l) is satisfied. For $k < T$ it follows from line 2 and the constraint (4.26a) that

$$\begin{aligned}
&\sum_{n\in[N]} \mathbb{1}_{X_l} \left(\zeta_n(k), \sigma_n(k)\right) = \sum_{n\in[N]} \sum_{\xi\in V} \sum_{\mu\in\mathbb{U}} \mathbb{1}_{X_l}(\xi,\mu)\ \mathbb{1}_{\{(\xi,\mu)\}} \left(\zeta_n(k), \sigma_n(k)\right) \\
&= \sum_{\xi\in V} \sum_{\mu\in\mathbb{U}} \mathbb{1}_{X_l}(\xi,\mu)\ \mathbf{r}_\xi^\mu(k) \leq R_l.
\end{aligned}$$

Thus the counting constraints are satisfied in the prefix phase. For the suffix phase, from line 4 and (4.26b) it follows that

$$\begin{aligned}
&\sum_{n\in[N]} \mathbb{1}_{X_l} \left(\zeta_n(k), \sigma_n(k)\right) \sum_{n\in[N]} \sum_{\xi\in V} \sum_{\mu\in\mathbb{U}} \mathbb{1}_{X_l}(\xi,\mu)\ \mathbb{1}_{\{(\xi,\mu)\}} \left(\zeta_n(k), \sigma_n(k)\right) \\
&= \sum_{\xi\in V} \sum_{\mu\in\mathbb{U}} \mathbb{1}_{X_l}(\xi,\mu) \sum_{j\in[J]} \left\langle C_j, \gamma_j^{\circlearrowleft k-T} \right\rangle^{\{(\xi,\mu)\}} = \sum_{j\in[J]} \left\langle C_j, \gamma_j^{\circlearrowleft k-T} \right\rangle^{X_l} \leq R_l.
\end{aligned}$$

Thus the switching protocol in Algorithm 3 generates inputs $\{\sigma_n(k)\}_{n\in[N]}$ and trajectories that satisfy the constraints of the instance (4.24). □

Proof of Theorem 4.5. Let $\{\pi_n^*\}_{n\in[N]}$ be a solution to the instance (4.24) and consider the generated controls $\sigma_n(k)$, $k \in \mathbb{N}$, and aggregate states $\mathbf{w}_\xi(k) = \sum_{n\in[N]} \mathbb{1}_{\{\xi\}}(\zeta_n(k))$. The number of possible values $\mathbf{w}(k)$ can take is finite and given by $\binom{|\mathbb{X}|+N-1}{N}$—the number of ways in which N identical objects (subsystems) can be partitioned into $|\mathbb{X}|$ sets (nodes). Thus there are times T_1, T_2 with $T_1 < T_2 \leq \binom{|\mathbb{X}|+N-1}{N}$ such that $\mathbf{w}(T_1) = \mathbf{w}(T_2)$. As shown below, the graph flows induced by $\{\pi_n^*\}_{n\in[N]}$ on the time interval $[T_1, T_2]$ can be achieved with cycle assignments by: defining a flow on a graph in a higher dimension, decomposing it into flows over cycles, and projecting the cyclic flow onto the original graph.

Let $G = (\mathbb{X}, \longrightarrow)$ be the system graph, and define a new graph $H = (\mathbb{X}_H, \longrightarrow_H)$. The node set $\mathbb{X}_H = \underbrace{\mathbb{X} \times \mathbb{X} \times \ldots \times \mathbb{X}}_{T_2-T_1 \text{ times}}$ contains $T_2 - T_1$ copies of each node in $\mathbb{X}$, and copies of $\xi \in \mathbb{X}$ are labeled ξ_k for $k \in [T_1, T_2]$. The set of edges is defined as

$$\longrightarrow_H = \left\{(\xi_k, \tilde{\xi}_{k+1}) : k \in \{T_1, \ldots, T_2 - 1\},\ (\xi, \tilde{\xi}) \in \longrightarrow\right\} \bigcup \{(\xi_{T_2}, \xi_{T_1}) : \xi \in \mathbb{X}\}.$$

An edge flow is induced on H by $\{\pi_n^*\}_{n\in[N]}$, obtained by letting the flow along $(\xi_k, \tilde{\xi}_{k+1})$ be the number of subsystems that traverses the edge $(\xi, \tilde{\xi}) \in \longrightarrow$ at time k, and by letting the flow along (ξ_{T_2}, ξ_{T_1}) be equal to the number of systems at ξ at time T_1. By construction, this flow is balanced at each node (i.e. inflows equal outflows).

By the flow decomposition theorem [4, Theorem 3.5], there are cycles and assignments in H that achieve this edge flow. As becomes evident from the proof in [4], these cycles are furthermore simple in H and thus of length at most $|\mathbb{X}_H| = |\mathbb{X}|(T_2 - T_1)$. By projecting these cycles onto a single copy of $\mathbb{X}$, cycles and assignments in G are obtained that mimic the counting performance of $\{\pi_n^*\}_{n\in[N]}$ on the interval $[T_1, T_2]$ when circulated.

An equivalent prefix-suffix strategy can therefore be constructed by taking as the prefix part

$$\mathbf{r}_\xi^\mu(k) = \sum_{n\in[N]} \mathbb{1}_{\{(\xi,\mu)\}}(\zeta_n(k), \sigma_n(k)), \quad k \in [T_1],$$

followed by a suffix part consisting of the cycles and assignments constructed as above. □

Lemma C.1. *Suppose $C = C_1 \cup C_2$ is a cycle that visits a node ξ_0 twice, so that it can be decomposed into two cycles $C_1 = (\xi_0, \xi_1) \ldots (\xi_i, \xi_0)$ and $C_2 = (\xi_0, \xi_{i+1}) \ldots (\xi_{|C|-1}, \xi_0)$.*

Let γ be an assignment to C that satisfies $\Psi^X(C,\gamma) \leq R$, let P be the graph period, and let

$$N_p = \sum_{i \in [|C|/P]} \gamma(p + iP), \quad p \in [P],$$
$$N_p^1 = \frac{|C_1|}{|C|} N_p, \quad N_p^2 = \frac{|C_2|}{|C|} N_p, \quad p \in [P].$$

Then the joint X-counts for the average assignments $\bar{\gamma}_{\{N_p^1\}_{p\in[P]}}$ for C_1 and $\bar{\gamma}_{\{N_p^2\}_{p\in[P]}}$ for C_2, satisfy

$$\Psi^X\left(\{C_1, C_2\}, \{\bar{\gamma}_{\{N_p^1\}_{p\in[P]}}, \bar{\gamma}_{\{N_p^2\}_{p\in[P]}}\}\right) \leq R.$$

Proof. For all i_1,

$$\gamma_{\{N_p^1\}_{p\in[P]}}(i_1) = \frac{PN^1_{(i_1 \mod P)}}{|C_1|} = \frac{PN_{(i_1 \mod P)}}{|C|} = \frac{P}{|C|} \sum_{k \in [|C|/P]} \gamma((i_1 \mod P) + kP),$$

and similarly for C_2. Note that P must divide both $|C_1|$ and $|C_2|$, so $((i_1 \mod |C_1|) \mod P) = (i_1 \mod P)$ for all i_1 and similarly for $|C_2|$. Below the short-hand notation $\mathbb{1}_X^C(i) = \mathbb{1}_X\left(\Phi_C^V(i), \Phi_C^{\mathbb{U}}(i)\right)$ is used to denote the indicator function of whether the i'th node in a cycle C is in the counting set X. For $j = 1, 2$,

$$\left\langle C_j, \bar{\gamma}^{\circlearrowleft k}_{\{N_p^j\}_{p\in[P]}} \right\rangle^X = \sum_{i_j \in [|C_j|]} \mathbb{1}_X^{C_j}(i_j) \left(\gamma_{\{N_p^j\}_{p\in[P]}}\right)^{\circlearrowleft k}(i_j)$$
$$= \frac{P}{|C|} \sum_{i_j \in [|C_j|]} \mathbb{1}_X^{C_j}(i_j) \sum_{k \in [|C|/P]} \gamma((i_j - k) \mod P + kP).$$

It holds that $i_2 \mod P = (|C_1| + i_2) \mod P$, and $i_2 \mapsto |C_1| + i_2$ is a mapping from $i_2 \in [|C_2|]$ to the corresponding index in C. A sum over both i_1 and i_2 can therefore be converted into a sum over the index i of C.

$$\left\langle C_1, \bar{\gamma}^{\circlearrowleft k}_{\{N_p^1\}_{p\in[P]}} \right\rangle^X + \left\langle C_2, \bar{\gamma}^{\circlearrowleft k}_{\{N_p^2\}_{p\in[P]}} \right\rangle^X$$
$$= \frac{P}{|C|} \sum_{i \in [|C|]} \mathbb{1}_X^C(i) \sum_{k \in [|C|/P]} \gamma((i - k) \mod P + kP)$$
$$\leq \max_{k \in [|C|/P]} \sum_{i \in [|C|]} \mathbb{1}_X^C(i)\, \gamma((i - k) \mod P + kP)$$
$$\leq \max_{k \in [|C|]} \sum_{i \in [|C|]} \mathbb{1}_X^C(i)\, \gamma^{\circlearrowleft k}(i) = \Psi^X(C, \gamma) \leq R.$$

□

Proof of Theorem 4.6. By Theorem 4.5, it is known that a correct solution must eventually lead to periodic behavior, and Lemma C.1 shows that the suffix of such a solution can be mapped into a suffix on simple cycles consisting of P-averaged assignments, where P is the period of the graph[26]. What remains to show is that this latter suffix is reachable from the initial state while respecting the relaxed counting constraints.

Let $\mathbf{w}(k)$ for $k \in \mathbb{N}$ be the trajectory generated from an integer solution to the instance (4.24). It satisfies the counting constraints for all k. From Lemma C.1 it follows that there is a set of simple cycles I such that some P-averaged assignments to these cycles satisfy the counting bounds. In addition, these assignments have the same parity structure as $\mathbf{w}(k)$, and hence as the initial condition; therefore they are reachable from the initial condition by virtue of Corollary 4.1. Next a switching protocol is proposed to steer the aggregate state to these assignments; the protocol consists in a gradual steering of a fraction of the systems from the original solution $\mathbf{w}(k)$ to the P-averaged assignments pertaining to the cycles in I.

The following notation is used: let $\alpha(k)$ be what remains of the correct trajectory $\mathbf{w}(k)$, let $\beta(k)$ be the fraction currently being steered towards cycle assignments, and let $\kappa(k)$ be the fraction that has already reached the simple cycles. Then $\alpha(0) = \mathbf{w}(0)$, and $\beta(0) = \kappa(0) = 0$. The overall system state is $\tilde{\mathbf{w}}(k) = \alpha(k) + \beta(k) + \kappa(k)$.

The protocol at time $k+1$ is as follows.

- If $\beta(k) = 0$, set $\alpha(k+1) = \alpha(k)(1 - \epsilon/\|\alpha(k)\|_1)$, and $\beta(k+1) = \alpha(k)\epsilon/\|\alpha(k)\|_1$,
- If $\beta(k)$ has reached the average assignments to I, set $\beta(k+1) = 0$, and $\kappa(k+1) = \beta(k) + \kappa(k)$,
- Otherwise, steer $\beta(k)$ toward the average assignments to cycles in I .

Remark that the transitions are all properly connected and merely illustrate the transport of subsystem "weight" from $\alpha(k)$ to the average assignments $\kappa(k)$. Transporting a mass ϵ takes at most time $(\mathrm{diam}(G)^2+1)$ [118]. It thus follows that the total transportation time is upper bounded by $(\mathrm{diam}(G)^2 + 1)N/\epsilon$, since an absolute weight ϵ is transported in each step.

Finally consider the counting bounds. By assumption, they are satisfied by $\mathbf{w}(k)$, and by Lemma C.1 they are also satisfied once the average assignments to cycles in I are reached.

[26] In the case of a non-connected graph the analysis can be done separately for each strongly connected component.

In the meantime,

$$\tilde{w}_\xi^\mu(k) = \alpha_\xi^\mu(k) + \beta_\xi^\mu(k) + \kappa_\xi^\mu(k).$$

For every k, there is an integer $z \leq 1/\epsilon$ such that

$$\alpha(k) = \left(1 - \frac{\epsilon z}{N}\right) \mathbf{w}(k), \quad \kappa(k) = \frac{\epsilon(z-1)}{N} \kappa_0(k),$$

where κ_0 is the average assignment to the cycles. Furthermore $\|\beta_\xi^\mu(k)\|_1 \leq \epsilon$ which shows that the counts are bounded as follows:

$$\begin{aligned}
&\sum_{\xi \in V} \sum_{\mu \in \mathbb{U}} \mathbb{1}_{X_l}(\xi, m)\ \tilde{w}_\xi^\mu(k) = \sum_{\xi \in V} \sum_{\mu \in \mathbb{U}} \mathbb{1}_{X_l}(\xi, m) \begin{pmatrix} (1 - \epsilon z/N) w_\xi^\mu(k) \\ + \epsilon(z-1)\kappa_0(k)/N + \beta_\xi^\mu(k) \end{pmatrix} \\
&\leq (1 - \epsilon z/N) R_l + \epsilon(z-1) R_l/N + \epsilon \leq R_l + \epsilon.
\end{aligned}$$

The protocol thus results in a prefix-suffix solution, where the suffix part consists of simple cycles, such that the relaxed counting bounds $(X_l, R_l + \epsilon)$ are satisfied. □

Bibliography

[1] A. A. Ahmadi and A. Majumdar. "DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization". In: *Proceedings of the IEEE Conference on Decision and Control*. 2014, pp. 394–401. DOI: 10.1109/CISS.2014.6814141.

[2] A. A. Ahmadi and P. A. Parrilo. "Towards Scalable Algorithms with Formal Guarantees for Lyapunov Analysis of Control Systems via Algebraic Optimization". In: *Proceedings of the IEEE Conference on Decision and Control*. 2014, pp. 2272–2281. DOI: 10.1109/CDC.2014.7039734.

[3] H. Ahn and D. D. Vecchio. "Safety Verification and Control for Collision Avoidance at Road Intersections". In: *IEEE Transactions on Automatic Control* (2017). DOI: 10.1109/TAC.2017.2729661.

[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Prentice Hall, 1993. ISBN: 978-0-13-617549-0.

[5] J. I. Alora, A. A. Gorodetsky, S. Karaman, Y. M. Marzouk, and N. Lowry. "Automated Synthesis of Low-rank Control Systems from sc-LTL Specifications using Tensor-Train Decompositions". In: *Proceedings of the IEEE Conference on Decision and Control*. 2016, pp. 1131–1138. DOI: 10.1109/CDC.2016.7798419.

[6] R. Alur, V. Forejt, S. Moarref, and A. Trivedi. "Safe schedulability of bounded-rate multi-mode systems". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2013, pp. 243–252. DOI: 10.1145/2461328.2461366.

[7] R. Alur, S. Moarref, and U. Topcu. "Pattern-Based Refinement of Assume-Guarantee Specifications in Reactive Synthesis". In: *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 2015, pp. 501–516. DOI: 10.1007/978-3-662-46681-0_49.

[8] R. Alur, A. Trivedi, and D. Wojtczak. "Optimal scheduling for constant-rate multi-mode systems". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2012, pp. 75–84. DOI: 10.1145/2185632.2185647.

[9] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems". In: *IEEE Transactions on Automatic Control* 62.8 (2016), pp. 3861–3876. DOI: 10.1109/TAC.2016.2638961.

[10] D. Angeli. "A Lyapunov approach to the incremental stability properties". In: *IEEE Transactions on Automatic Control* 47.3 (2002), pp. 410–421. DOI: 10.1109/9.989067.

[11] N. Athanasopoulos and R. M. Jungers. "Computing the domain of attraction of switching systems subject to non-convex constraints". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2016, pp. 41–50. DOI: 10.1145/2883817.2883823.

[12] C. Baier and J. Katoen. *Principles of Model Checking*. MIT Press, 2008. ISBN: 978-0-262-02649-9.

[13] A. Balkan, M. Vardi, and P. Tabuada. "Mode-Target Games: Reactive Synthesis for Control Applications". In: *IEEE Transactions on Automatic Control (to appear)* (2017). DOI: 10.1109/TAC.2017.2722960.

[14] G. Batt, C. Belta, and R. Weiss. "Temporal Logic Analysis of Gene Networks Under Parameter Uncertainty". In: *IEEE Transactions on Automatic Control* 53.Special Issue (2008), pp. 215–229. DOI: 10.1109/TAC.2007.911330.

[15] P. C. Bell, J.-C. Delvenne, R. M. Jungers, and V. D. Blondel. "The continuous Skolem-Pisot problem". In: *Theoretical Computer Science* 411.40-42 (2010), pp. 3625–3634. DOI: 10.1016/j.tcs.2010.06.005.

[16] C. Belta and L. Habets. "Controlling a class of nonlinear systems on rectangles". In: *IEEE Transactions on Automatic Control* 51.11 (2006), pp. 1749–1759. DOI: 10.1109/TAC.2006.884957.

[17] C. Belta, B. Yordanov, and E. A. Gol. *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017. ISBN: 978-3-319-50762-0.

[18] L. Benvenuti, A. Ferrari, E. Mazzi, and A. L. S. Vincentelli. "Contract-Based Design for Computation and Verification of a Closed-Loop Hybrid System". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. Ed. by M. Egerstedt and B. Mishra. 2008, pp. 58–71. DOI: 10.1007/978-3-540-78929-1_5.

[19] D. Bertsekas. "Infinite time reachability of state-space regions by using feedback control". In: *IEEE Transactions on Automatic Control* 17.5 (1972), pp. 604–613. DOI: 10.1109/TAC.1972.1100085.

[20] F. Blanchini and S. Miani. *Set-Theoretic Methods in Control*. Birkhäuser, 2008. ISBN: 978-0-8176-3255-7.

[21] R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. "Synthesis of Reactive(1) designs". In: *Journal of Computer and System Sciences* 78.3 (2012), pp. 911–938. DOI: 10.1016/j.jcss.2011.08.007.

[22] S. Boyd. *Linear Matrix Inequalities in System and Control Theory*. SIAM, 1994. ISBN: 978-0-89871-334-3.

[23] R. D. Carr and G. Konjevod. "Polyhedral Combinatorics". In: *Tutorials on Emerging Methodologies and Applications in Operations Research*. Springer New York, 2005, (2–1)–(2–46). DOI: 10.1007/0-387-22827-6_2.

[24] K. Chatterjee and T. A. Henzinger. "Assume-Guarantee Synthesis". In: *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. 2007, pp. 261–275. DOI: 10.1007/978-3-540-71209-1_21.

[25] F. H. Clarke, Y. S. Ledyaev, R. J. Stern, and P. R. Wolenski. *Nonsmooth Analysis and Control Theory*. Springer-Verlag, 1998. ISBN: 978-0-387-98336-3.

[26] K. L. Clarkson. "More output-sensitive geometric algorithms". In: *Proceedings of the Annual Symposium on Foundations of Computer Science*. 1994, pp. 695–702. DOI: 10.1109/SFCS.1994.365723.

[27] A. Colombo and D. Del Vecchio. "Least Restrictive Supervisors for Intersection Collision Avoidance: A Scheduling Approach". In: *IEEE Transactions on Automatic Control* 60.6 (2015), pp. 1515–1527. DOI: 10.1109/TAC.2012.2219131.

[28] S. Coogan and M. Arcak. "Efficient finite abstraction of mixed monotone systems". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2015, pp. 58–67. DOI: 10.1145/2728606.2728607.

[29] M. G. Crandall and P.-L. Lions. "Viscosity solutions of Hamilton-Jacobi equations". In: *Transactions of the American Mathematical Society* 277.1 (1983), pp. 1–42. DOI: 10.2307/1999343.

[30] S. J. Crocker and J. L. Mathieu. "Adaptive state estimation and control of thermostatic loads for real-time energy balancing". In: *Proc. ACC*. 2016, pp. 3557–3563. DOI: 10.1109/ACC.2016.7525465.

[31] E. Dallal, A. Colombo, D. Del Vecchio, and S. Lafortune. "Supervisory control for collision avoidance in vehicular networks using discrete event abstractions". In: *Discrete Event Dynamic Systems* 27.1 (2016), pp. 1–44. DOI: 10.1007/s10626-016-0228-3.

[32] E. Dallal and P. Tabuada. "Decomposing Controller Synthesis for Safety Specifications". In: *Proceedings of the IEEE Conference on Decision and Control*. 2016, pp. 5720–5725. DOI: 10.1109/CDC.2016.7799148.

[33] E. Dallal and P. Tabuada. "On compositional symbolic controller synthesis inspired by small-gain theorems". In: *Proceedings of the IEEE Conference on Decision and Control*. 2015, pp. 6133–6138. DOI: 10.1109/CDC.2015.7403184.

[34] E. De Santis, M. D. Di Benedetto, and L. Berardi. "Computation of Maximal Safe Sets for Switching Systems". In: *IEEE Transactions on Automatic Control* 49.2 (2004), pp. 184–195. DOI: 10.1109/TAC.2003.822860.

[35] E. W. Dijkstra. "Why numbering should start at zero". 1982. URL: http://www.cs.utexas.edu/users/EWD/ewd08xx/EWD831.PDF.

[36] R. Ehlers, S. Lafortune, S. Tripakis, and M. Y. Vardi. "Supervisory control and reactive synthesis: a comparative introduction". In: *Discrete Event Dynamic Systems* 27.2 (2016), pp. 209–260. DOI: 10.1007/s10626-015-0223-0.

[37] S. Esmaeil Zadeh Soudjani and A. Abate. "Aggregation and control of populations of thermostatically controlled loads by formal abstractions". In: *IEEE Transactions on Control Systems Technology* 23.3 (2015), pp. 975–990. DOI: 10.1109/TCST.2014.2358844.

[38] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, 2011. ISBN: 978-0-8218-4974-3.

[39] A. Feuer and M. Heymann. "Ω-invariance in control systems with bounded controls". In: *Journal of Mathematical Analysis and Applications* 53.2 (1976), pp. 266–276. DOI: 10.1016/0022-247X(76)90110-4.

[40] I. Filippidis, S. Dathathri, S. C. Livingston, N. Ozay, and R. M. Murray. "Control design for hybrid systems with TuLiP: The temporal logic planning toolbox". In: *Proceedings of the IEEE Conference on Control Applications*. 2016, pp. 1030–1041. DOI: 10.1109/CCA.2016.7587949.

[41] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry. "Reach-Avoid Problems with Time-Varying Dynamics, Targets and Constraints". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2015, pp. 11–20. DOI: 10.1145/2728606.2728612.

[42] E. Garone, S. Di Cairano, and I. V. Kolmanovsky. "Reference and command governors for systems with constraints: A survey on theory and applications". In: *Automatica* 75 (2017), pp. 306–328. DOI: 10.1016/j.automatica.2016.08.013.

[43] A. Ghaffari, S. Moura, and M. Krstic. "Modeling, Control, and Stability Analysis of Heterogeneous Thermostatically Controlled Load Populations Using Partial Differential Equations". In: *Journal of Dynamic Systems, Measurement, and Control* 137.10 (2015), p. 101009. DOI: 10.1115/1.4030817.

[44] E. G. Gilbert and K. T. Tan. "Linear systems with state and control constraints: the theory and application of maximal output admissible sets". In: *IEEE Transactions on Automatic Control* 36.9 (1991), pp. 1008–1020. DOI: 10.1109/9.83532.

[45] A. Girard and S. Martin. "Control Synthesis for Constrained Nonlinear Systems using Hybridization and Robust Controllers on Simplices". In: *IEEE Transactions on Automatic Control* 57.4 (2012), pp. 1046–1051. DOI: 10.1109/TAC.2011.2168874.

[46] A. Girard, G. Pola, and P. Tabuada. "Approximately Bisimilar Symbolic Models for Incrementally Stable Switched Systems". In: *IEEE Transactions on Automatic Control* 55.1 (2010), pp. 116–126. DOI: 10.1109/TAC.2009.2034922.

[47] A. Girard. "Reachability of Uncertain Linear Systems Using Zonotopes". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2005, pp. 291–305. DOI: 10.1007/978-3-540-31954-2_19.

[48] A. Girard and G. J. Pappas. "Approximation Metrics for Discrete and Continuous Systems". In: *IEEE Transactions on Automatic Control* 52.5 (2007), pp. 782–798. DOI: 10.1109/TAC.2007.895849.

[49] A. A. Gorodetsky, S. Karaman, and Y. M. Marzouk. "High-Dimensional Stochastic Optimal Control using Continuous Tensor Decompositions". In: (2016). arXiv: 1611.04706 [cs.RO].

[50] Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*. 2015. URL: http://www.gurobi.com.

[51] M. Gwerder, B. Lehmann, J. Tödtli, V. Dorer, and F. Renggli. "Control of thermally-activated building systems (TABS)". In: *Applied Energy* 85.7 (2008), pp. 565–581. DOI: 10.1016/j.apenergy.2007.08.001.

[52] L. Habets, P. Collins, and J. van Schuppen. "Reachability and control synthesis for piecewise-affine hybrid systems on simplices". In: *IEEE Transactions on Automatic Control* 51.6 (2006), pp. 938–948. DOI: 10.1109/TAC.2006.876952.

[53] H. Hao, B. M. Sanandaji, K. Poolla, and T. L. Vincent. "Aggregate Flexibility of Thermostatically Controlled Loads". In: *IEEE Transactions on Power Systems* 30.1 (2015), pp. 189–198. DOI: 10.1109/TPWRS.2014.2328865.

[54] D. Henrion and M. Korda. "Convex Computation of the Region of Attraction of Polynomial Control Systems". In: *IEEE Transactions on Automatic Control* 59.2 (2014), pp. 297–312. DOI: 10.1109/TAC.2013.2283095.

[55] T. A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. "Lazy Abstraction". In: *ACM SIGPLAN Notices* 37.1 (2002), pp. 58–70. DOI: 10.1145/565816.503279.

[56] T. A. Henzinger, P. W. Kopke, A. Puri, and P. P. Varaiya. "What's Decidable about Hybrid Automata?" In: *Journal of Computer and System Sciences* 57.1 (1998), pp. 94–124. DOI: 10.1006/jcss.1998.1581.

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

[58] M. W. Hirsch, S. Smale, and R. L. Devaney. *Differential Equations, Dynamical Systems, and an Introduction to Chaos*. 3rd ed. Academic Press, 2012. ISBN: 978-0-12-382010-5.

[59] S. Jacobs, R. Bloem, R. Brenguier, A. Khalimov, F. Klein, R. Könighofer, J. Kreber, A. Legg, N. Narodytska, G. A. Pérez, J.-F. Raskin, L. Ryzhyk, O. Sankur, M. Seidl, L. Tentrup, and A. Walker. "The 3rd Reactive Synthesis Competition (SYNTCOMP 2016): Benchmarks, Participants & Results". In: *Proceedings of the Workshop on Synthesis*. 2016, pp. 149–177. arXiv: 1609.00507v2 [cs.LO].

[60] Z. P. Jiang and Y. Wang. "A converse Lyapunov theorem for discrete-time systems with disturbances". In: *Systems & Control Letters* 45.1 (2002), pp. 49–58. DOI: 10.1016/S0167-6911(01)00164-5.

[61] C. Josz. "Counterexample to global convergence of DSOS and SDSOS hierarchies". In: (2017). arXiv: 1707.02964v1 [math.OC].

[62] Y. Kesten and A. Pnueli. "Verification by Augmented Finitary Abstraction". In: *Information and Computation* 163.1 (2000), pp. 203–243. ISSN: 0890-5401. DOI: 10.1006/inco.2000.3000.

[63] E. S. Kim, M. Arcak, and S. A. Seshia. "Symbolic control design for monotone systems with directed specifications". In: *Automatica* 83 (2017), pp. 10–19. DOI: 10.1016/j.automatica.2017.04.060.

[64] M. Kloetzer and C. Belta. "A Fully Automated Framework for Control of Linear Systems From Temporal Logic Specifications". In: *IEEE Transactions on Automatic Control* 53.1 (2008), pp. 287–297. DOI: 10.1109/TAC.2007.914952.

[65] M. Korda, D. Henrion, and C. N. Jones. "Controller design and value function approximation for nonlinear dynamical systems". In: *Automatica* 67 (2016), pp. 54–66. DOI: 10.1016/j.automatica.2016.01.022.

[66] M. Korda, D. Henrion, and C. N. Jones. "Convex Computation of the Maximum Controlled Invariant Set For Polynomial Control Systems". In: *SIAM Journal on Control and Optimization* 52.5 (2014), pp. 2944–2969. DOI: 10.1137/130914565.

[67] R. Koymans. "Specifying real-time properties with metric temporal logic". In: *Real-time systems* 2.4 (1990), pp. 255–299. DOI: 10.1007/BF01995674.

[68] D. Kozen. "Results on the propositional μ-calculus". In: *Theoretical Computer Science* 27.3 (1983), pp. 333–354. DOI: 10.1016/0304-3975(82)90125-6.

[69] G. Lafferriere, G. J. Pappas, and S. Yovine. "Reachability computation for linear hybrid systems". In: *Proceedings of the IFAC World Congress*. 1998, pp. 7–12. DOI: 10.1016/S1474-6670(17)56362-4.

[70] J. B. Lasserre. *Moments, Positive Polynomials and Their Applications*. Imperial College Press, 2010. ISBN: 978-1-84816-445-1.

[71] J. B. Lasserre. "Tractable approximations of sets defined with quantifiers". In: *Mathematical Programming* 151.2, Ser. B (2015), pp. 507–527. DOI: 10.1007/s10107-014-0838-1.

[72] Y. Lin, E. D. Sontag, and Y. Wang. "A smooth converse Lyapunov theorem for robust stability". In: *SIAM Journal on Control and Optimization* 34.1 (1996), pp. 124–160. DOI: 10.1137/S0363012993259981.

[73] J. Liu, N. Ozay, U. Topcu, and R. Murray. "Switching Protocol Synthesis for Temporal Logic Specifications". In: *Proceedings of the American Control Conference*. 2012, pp. 727–734. DOI: 10.1109/ACC.2012.6315040.

[74] J. Liu and N. Ozay. "Abstraction, Discretization, and Robustness in Temporal Logic Control of Dynamical Systems". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2014, pp. 293–302. DOI: 10.1145/2562059.2562137.

[75] J. Liu and N. Ozay. "Finite abstractions with robustness margins for temporal logic-based control synthesis". In: *Nonlinear Analysis: Hybrid Systems* 22 (2016), pp. 1–15. DOI: 10.1016/j.nahs.2016.02.002.

[76] J. Liu, N. Ozay, U. Topcu, and R. M. Murray. "Synthesis of Reactive Switching Protocols From Temporal Logic Specifications". In: *IEEE Transactions on Automatic Control* 58.7 (2013), pp. 1771–1785. DOI: 10.1109/TAC.2013.2246095.

[77] J. Löfberg. "YALMIP : A toolbox for modeling and optimization in MATLAB". In: *Proceedings of IEEE International Symposium on Computer Aided Control System Design*. 2004, pp. 284–289. DOI: 10.1109/CACSD.2004.1393890.

[78] S. Lovett and R. Meka. "Constructive Discrepancy Minimization by Walking on the Edges". In: *SIAM Journal on Computing* 44.5 (2015), pp. 1573–1582. DOI: 10.1137/130929400.

[79] J. Lygeros, C. Tomlin, and S. Sastry. "Controllers for reachability specifications for hybrid systems". In: *Automatica* 35.3 (1999), pp. 349–370. DOI: 10.1016/S0005-1098(98)00193-9.

[80] R. Mattila, Y. Mo, and R. M. Murray. "An iterative abstraction algorithm for reactive correct-by-construction controller synthesis". In: *Proceedings of the IEEE Conference on Decision and Control*. 2015, pp. 6147–+6152. DOI: 10.1109/CDC.2015.7403186.

[81] M. Mazo Jr, A. Davitian, and P. Tabuada. "PESSOA: A Tool for Embedded Controller Synthesis". In: *Proceedings of the International Conference on Computer Aided Verification*. 2010, pp. 566–569. DOI: 10.1007/978-3-642-14295-6_49.

[82] P.-J. Meyer, A. Girard, and E. Witrant. "Compositional abstraction and safety synthesis using overlapping symbolic models". In: (2017). arXiv: 1704.07124v2 [cs.SY].

[83] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games". In: *IEEE Transactions on Automatic Control* 50.7 (2005), pp. 947–957. DOI: 10.1109/TAC.2005.851439.

[84] S. Mohan and R. Vasudevan. "Convex Computation of the Reachable Set for Hybrid Systems with Parametric Uncertainty". In: *Proceedings of the American Control Conference*. 2016, pp. 5141–5147. DOI: 10.1109/ACC.2016.7526475.

[85] J. Mycielski. "Chapter 3 Games with perfect information". In: *Handbook of Game Theory with Economic Applications, Volume 1*. Ed. by R. J. Aumann and S. Hart. Elsevier, 1992, pp. 41–70. DOI: 10.1016/S1574-0005(05)80006-2.

[86] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. Ames, J. Grizzle, N. Ozay, H. Peng, and P. Tabuada. "Correct-By-Construction Adaptive Cruise Control: Two Approaches". In: *IEEE Transactions on Control Systems Technology* 24.4 (2016), pp. 1294–1307. DOI: 10.1109/TCST.2015.2501351.

[87] P. Nilsson, O. Hussien, Y. Chen, A. Balkan, M. Rungger, A. Ames, J. Grizzle, N. Ozay, H. Peng, and P. Tabuada. "Preliminary Results on Correct-by-Construction Control Software Synthesis for Adaptive Cruise Control". In: *Proceedings of the IEEE Conference on Decision and Control*. 2014, pp. 816–823. DOI: 10.1109/CDC.2014.7039482.

[88] P. Nilsson and N. Ozay. "Control Synthesis for High-Dimensional Systems With Counting Constraints". In: (2017). arXiv: 1706.07863v1 [cs.SY].

[89] P. Nilsson and N. Ozay. "Control Synthesis for Large Collections of Systems with Mode-Counting Constraints". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2016, pp. 205–214. DOI: 10.1145/2883817.2883831.

[90] P. Nilsson and N. Ozay. "Incremental Synthesis of Switching Protocols via Abstraction Refinement". In: *Proceedings of the IEEE Conference on Decision and Control*. 2014, pp. 6246–6253. DOI: 10.1109/CDC.2014.7040368.

[91] P. Nilsson and N. Ozay. "Maximizing the Time of Invariance for Large Collections of Switched Systems". In: *Proceedings of the IEEE Conference on Decision and Control (to appear)*. 2017.

[92] P. Nilsson and N. Ozay. "On a class of maximal invariance inducing control strategies for large collections of switched systems". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2017, pp. 187–196. DOI: 10.1145/3049797.3049810.

[93] P. Nilsson and N. Ozay. "Synthesis of separable controlled invariant sets for modular local control design". In: *Proceedings of the American Control Conference*. 2016, pp. 5656–5663. DOI: 10.1109/ACC.2016.7526557.

[94] P. Nilsson, N. Ozay, and J. Liu. "Augmented Finite Transition Systems as Abstractions for Control Synthesis". In: *Discrete Event Dynamic Systems* 27.2 (2017), pp. 301–340. DOI: 10.1007/s10626-017-0243-z.

[95] N. Ozay, J. Liu, P. Prabhakar, and R. Murray. "Computing Augmented Finite Transition Systems to Synthesize Switching Protocols for Polynomial Switched Systems". In: *Proceedings of the American Control Conference*. 2013, pp. 6237–6244. DOI: 10.1109/ACC.2013.6580816.

[96] P. Parrilo. "Semidefinite programming relaxations for semialgebraic problems". In: *Mathematical Programming* 96.2 (2003), pp. 293–320. DOI: 10.1007/s10107-003-0387-5.

[97] N. Piterman and A. Pnueli. "Faster Solutions of Rabin and Streett Games". In: *Proceedings of the IEEE Symposium on Logic in Computer Science*. 2006, pp. 275–284. DOI: 10.1109/LICS.2006.23.

[98] A. Pnueli and R. Rosner. "On the synthesis of a reactive module". In: *Proceedings of the ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. 1989, pp. 179–190. DOI: 10.1145/75277.75293.

[99] A. Pnueli and R. Rosner. "On the synthesis of an asynchronous reactive module". In: *Proceedings of the International Colloquium on Automata, Languages, and Programming*. 1989, pp. 652–671. DOI: 10.1007/BFb0035790.

[100] G. Pola, A. Girard, and P. Tabuada. "Approximately bisimilar symbolic models for nonlinear control systems". In: *Automatica* 44.10 (2008), pp. 2508–2516. DOI: 10.1016/j.automatica.2008.02.021.

[101] G. Pola, P. Pepe, and M. D. Di Benedetto. "Symbolic Models for Networks of Control Systems". In: *IEEE Transactions on Automatic Control* 61.11 (2016), pp. 3663–3668. DOI: 10.1109/TAC.2016.2528046.

[102] G. Pola and P. Tabuada. "Symbolic Models for Nonlinear Control Systems: Alternating Approximate Bisimulations". In: *SIAM Journal on Control and Optimization* 48.2 (2009), pp. 719–733. DOI: 10.1137/070698580.

[103] I. Pólik and T. Terlaky. "A Survey of the S-Lemma". In: *SIAM review* 49.3 (2007), pp. 371–418. DOI: 10.1137/S003614450444614X.

[104] M. Putinar. "Positive Polynomials on Compact Semi-Algebraic Sets". In: *Indiana University Mathematics Journal* 43.2 (1993), pp. 969–984.

[105] S. V. Rakovic, B. Kern, and R. Findeisen. "Practical Robust Positive Invariance for Large-Scale Discrete Time Systems". In: *Proceedings of the IFAC World Congress*. 2011, pp. 6425–6430. DOI: 10.3182/20110828-6-IT-1002.03146.

[106] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. "Model predictive control with signal temporal logic specifications". In: *Proceedings of the IEEE Conference on Decision and Control*. 2014, pp. 81–87. DOI: 10.1109/CDC.2014.7039363.

[107] B. C. Rawlings and B. E. Ydstie. "Falsification of combined invariance and reachability specifications in hybrid control systems". In: *Discrete Event Dynamic Systems* 27.2 (2017), pp. 463–479. DOI: 10.1007/s10626-017-0241-1.

[108] G. Reissig, A. Weber, and M. Rungger. "Feedback Refinement Relations for the Synthesis of Symbolic Controllers". In: *IEEE Transactions on Automatic Control* 62.4 (2017), pp. 1781–1796. DOI: 10.1109/TAC.2016.2593947.

[109] *Road Design Manual, Chapter 3 Geometrics*. Tech. rep. Michigan Department of Transportation, 2016.

[110] J. Romaní, A. de Gracia, and L. F. Cabeza. "Simulation and control of thermally activated building systems (TABS)". In: *Energy & Buildings* 127 (2016), pp. 22–42. DOI: 10.1016/j.enbuild.2016.05.057.

[111] E. J. Rossetter and J. C. Gerdes. "Lyapunov based performance guarantees for the potential field lane-keeping assistance system". In: *Journal of dynamic systems, measurement, and control* 128.3 (2006), pp. 510–522. DOI: 10.1115/1.2192835.

[112] M. Rungger and P. Tabuada. "Computing Robust Controlled Invariant Sets of Linear Systems". In: *IEEE Transactions on Automatic Control* 62.7 (2017), pp. 3665–3670. DOI: 10.1109/TAC.2017.2672859.

[113] M. Rungger and M. Zamani. "SCOTS: A Tool for the Synthesis of Symbolic Controllers". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2016, pp. 99–104. DOI: 10.1145/2883817.2883834.

[114] S. Sadraddini and C. Belta. "Feasibility Envelopes for Metric Temporal Logic Specifications". In: *Proceedings of the IEEE Conference on Decision and Control*. 2016, pp. 5732–5737. DOI: 10.1109/CDC.2016.7799150.

[115] Y. E. Sahin, P. Nilsson, and N. Ozay. "Provably-correct coordination of large collections of agents with counting temporal logic constraints". In: *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems*. 2017, pp. 249–258. DOI: 10.1145/3055004.3055021.

[116] Y. E. Sahin, P. Nilsson, and N. Ozay. "Synchronous and Asynchronous Multi-agent Coordination with cLTL+ Constraints". In: *Proceedings of the IEEE Conference on Decision and Control (to appear)*. 2017.

[117] E. Seneta. *Non-negative matrices and Markov chains*. Springer, 2006. ISBN: 978-0-387-29765-1.

[118] J. Shen. "A bound on the exponent of primitivity in terms of diameter". In: *Linear Algebra and its Applications* 244 (1996), pp. 21–33. DOI: 10.1016/0024-3795(94)00205-3.

[119] S. Smith, P. Nilsson, and N. Ozay. "Interdependence quantification for compositional control synthesis: An application in vehicle safety systems". In: *Proceedings of the IEEE Conference on Decision and Control*. 2016, pp. 5700–5707. DOI: 10.1109/CDC.2016.7799145.

[120] M. Sourbron, C. Verhelst, and L. Helsen. "Building models for model predictive control of office buildings with concrete core activation". In: *Journal of Building Performance Simulation* 6.3 (2013), pp. 175–198. DOI: 10.1080/19401493.2012.680497.

[121] F. Sun, N. Ozay, E. M. Wolff, J. Liu, and R. M. Murray. "Efficient control synthesis for augmented finite transition systems with an application to switching protocols". In: *Proceedings of the American Control Conference*. 2014, pp. 3273–3280. DOI: 10.1109/ACC.2014.6859428.

[122] M. Svorenova, J. Kretinsky, M. Chmelik, K. Chatterjee, I. Cerna, and C. Belta. "Temporal logic control for stochastic linear systems using abstraction refinement of probabilistic games". In: *Proceedings of the International Conference on Hybrid Systems: Computation and Control*. 2015, pp. 259–268. DOI: 10.1016/j.nahs.2016.04.006.

[123] P. Tabuada. *Verification and Control of Hybrid Systems*. Springer, 2009. ISBN: 978-1-4419-5498-5.

[124] P. Tabuada and G. J. Pappas. "Linear time logic control of discrete-time linear systems". In: *IEEE Transactions on Automatic Control* 51.12 (2006), pp. 1862–1877. DOI: 10.1109/TAC.2006.886494.

[125] F. Tahir and I. M. Jaimoukha. "Low-Complexity Polytopic Invariant Sets for Linear Systems Subject to Norm-Bounded Uncertainty". In: *IEEE Transactions on Automatic Control* 60.5 (2015), pp. 1416–1421. DOI: 10.1109/TAC.2014.2352692.

[126] C. J. Tomlin, J. Lygeros, and S. S. Sastry. "A game theoretic approach to controller design for hybrid systems". In: *Proceedings of the IEEE* 88.7 (2000), pp. 949–970. DOI: 10.1109/5.871303.

[127] R. Vasudevan, R. Bajcsy, and R. Tedrake. "Convex Computation of the Reachable Set for Controlled Polynomial Hybrid Systems". In: *Proceedings of the IEEE Conference on Decision and Control*. 2014, pp. 1499–1506. DOI: 10.1109/CDC.2014.7039612.

[128] W. Walter and R. Thompson. *Ordinary differential equations*. 1st ed. Springer, 1998. ISBN: 978-0-387-98459-9.

[129] E. M. Wolff, U. Topcu, and R. M. Murray. "Efficient reactive controller synthesis for a fragment of linear temporal logic". In: *Proceedings of the IEEE International Conference on Control and Automation*. 2013, pp. 5033–5040. DOI: 10.1109/ICRA.2013.6631296.

[130] E. M. Wolff, U. Topcu, and R. M. Murray. "Optimization-based trajectory generation with linear temporal logic specifications". In: *Proceedings of the IEEE Conference on Robotics and Automation*. 2014, pp. 5319–5325. DOI: 10.1109/ICRA.2014.6907641.

[131] L. Yang, N. Ozay, and A. Karnik. "Synthesis of fault tolerant switching protocols for vehicle engine thermal management". In: *Proceedings of the American Control Conference*. 2016, pp. 4213–4220. DOI: 10.1109/ACC.2016.7525584.

[132] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta. "Temporal Logic Control of Discrete-Time Piecewise Affine Systems". In: *IEEE Transactions on Automatic Control* 57.6 (2012), pp. 1491–1504. DOI: 10.1109/TAC.2011.2178328.

[133] M. Zamani, G. Pola, M. Mazo Jr, and P. Tabuada. "Symbolic Models for Nonlinear Control Systems Without Stability Assumptions". In: *IEEE Transactions on Automatic Control* 57.7 (2012), pp. 1804–1809. DOI: 10.1109/TAC.2011.2176409.