A Modular and Hierarchical Framework for Motion Planning with Feedback-Based Motion Primitives

by

Marijan Vukosavljev

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Electrical & Computer Engineering
University of Toronto

Abstract

A Modular and Hierarchical Framework for Motion Planning with Feedback-Based Motion Primitives

Marijan Vukosavljev

Doctor of Philosophy

Graduate Department of Electrical & Computer Engineering

University of Toronto

2019

As robots become more integrated into everyday society, an increasing emphasis is being placed on their ability to execute complex tasks while maintaining safety. One of the most fundamental tasks in motion planning and control is the coordination of multiple robots to safely and efficiently reach target destinations. In recent years, a hybrid systems approach, that which combines both continuous and discrete system descriptions, has proven to be an attractive methodology for control with complex specifications. Although there is extensive literature on both the design of continuous time feedback controllers and discrete motion planning algorithms, relatively few works address the rigorous integration of these two components, especially in the context of multi-vehicle coordination. In this dissertation, we leverage the hybrid systems paradigm to formulate a novel framework for motion planning and control of multi-vehicle systems that is modular, robust, and provably safe.

This dissertation contains three distinct contributions. The first contribution considers the synthesis of low level continuous time feedback controllers for guiding system trajectories along a desired direction; to this end, an open problem in classical linear quadratic control was solved. The second contribution broadens the scope to develop a modular motion planning framework that combines low level feedback-based motion primitives with high level planning algorithms. Finally, the third contribution extends this modular framework towards a multi-hierarchy of motion primitives in order to improve scalability with respect to the number of vehicles. Both the second and third contributions include experimental validation on a collection of quadrocopters.

Acknowledgements

I wish to thank my supervisors, Prof. Mireille Broucke and Prof. Angela Schoellig, for giving me the opportunity pursue graduate studies as their student. This has been an extremely valuable experience towards my professional and personal development. Your mentorship and guidance has provided me with new perspectives on how to do research and, perhaps more importantly, how to communicate it. I wish you both continued success in your future endeavors.

I would like to thank the faculty, staff, and students at both the Systems Control Group at the University of Toronto and the University of Toronto Institute for Aerospace studies (UTIAS). I am grateful to Prof. Luca Scardovi and Prof. Lacra Pavel for serving on my thesis progress committee and qualifying examination. I would like to thank Prof. Raymond Kwong and Prof. Ashish Khisti for serving on my qualifying examination, and the latter for also serving as the chair for my departmental oral defence. Finally, I would like to thank Prof. Calin Belta for serving as my external examiner and Prof. Mark Fox for serving as the chair for my final oral defence.

I have been fortunate to have been surrounded by hard-working and talented individuals. First, I would like to thank Ivo Jansen for joining me in the first experimental work on quadrocopters. Next, I would like to acknowledge the many useful and entertaining discussions with Adam Sniderman and Melkior Ornik. The framework on motion planning was developed as joint work with Zach Kroeze and I thank him for this wonderful collaborative experience. I also thank Robert Adragna for his contribution towards this project. The Dynamic Systems Lab (DSL) at UTIAS has grown so much since its beginning and has provided great support. I am particularly indebted to Chris McKinnon, Tracy Du, and SiQi Zhou.

Finally, I would like to thank my parents for their support and encouragement.

Declaration of Previous Publication

With minor exceptions, the entirety of the results presented in Chapters 3, 4, and 5 of this thesis, and the majority of the accompanying text, is contained in the previous manuscripts [110, 109, 111, 112] coauthored by the author of this thesis and his collaborators during Ph.D studies. These results have been published in peer-reviewed journal and conference venues, or are currently in the submission process.

The author of this thesis is the primary author on all these works. Chapter 3 is adapted from [110]. Chapter 4 is adapted from [111], which extends the results in [109], and was done in collaboration with Zach Kroeze. Finally, Chapter 5 is adapted from [112], which is aimed for a journal publication.

Contents

List of Figures

Index of Basic Symbols and Acronyms

Some standard symbols used in the thesis:

$x := y$	x is defined as y
$\forall$	for all
$\exists$	there exists
$\Rightarrow$	implies
$\Leftrightarrow$	equivalent
$\emptyset$	empty set
$\mathcal{X} \subset \mathcal{Y}$	set $\mathcal{X}$ is a subset of set $\mathcal{Y}$
$\mathcal{X} \supset \mathcal{Y}$	set $\mathcal{X}$ is a superset of set $\mathcal{Y}$
$\mathcal{X} \cap \mathcal{Y}$	intersection of sets $\mathcal{X}$ and $\mathcal{Y}$
$\mathcal{X} \cup \mathcal{Y}$	union of sets $\mathcal{X}$ and $\mathcal{Y}$
$x \in \mathcal{K}$	x is an element of the set $\mathcal{K}$
$\mathbb{Z}$	set of all integers
$\mathbb{R}$	set of all real numbers
$\mathbb{C}$	set of all complex numbers
$\mathbb{R}^n$	set of all real n-tuples
$\mathbb{R}^{m \times n}$	set of all real $m \times n$ matrices
$A^\top$	transpose of matrix A
A^{-1}	inverse of a square matrix A
$\sigma(A)$	spectrum of a square matrix A
$\Re(s)$	the real part of a complex number s
$\|x\|$	standard norm of vector x

$\text{co}\{v_1, \ldots, v_n\}$	convex hull of points $v_1, \ldots, v_n$

Acronyms used throughout the thesis:

RCP	Reach Control Problem
LTL	Linear Temporal Logic
LQ	Linear Quadratic
LQCP	Linear Quadratic Control Problem
ARE	Algebraic Riccati Equation
ARI	Algebraic Riccati Inequality
OTS	Output Transition System
MA	Maneuver Automaton
HMA	Hierarchical Maneuver Automaton
PA	Product Automaton
w.r.t	with respect to

Chapter 1

Introduction

With advancements in electronic and computer technology, robotic systems are becoming more embedded into our society due to their increasing capabilities and affordability. Robotic systems have the potential to transform many aspects of both the industrial and commercial domains. Among the many platforms that are currently available, some of the most exciting are flying vehicles such as the quadrocopter. Forthcoming applications are expected in a wide variety of domains such as agriculture, cinematography, construction, entertainment, environmental monitoring, health care, military, mining, search and rescue, sports, transportation, and so on. Developing a sufficient level of autonomy in these systems will be central in enabling them to perform these various functions in an efficient and safe manner.

Control theory has provided the foundations for the control of robotic systems. One can observe that the predominant method of control has been based on some form of reference trajectory tracking. However, real systems often involve complex specifications such as multiple modes of operation, safety requirements, other performance requirements like liveness, or temporal logics as specification languages. While reference trajectory tracking can be adapted to address such problems on a case-by-case basis, more systematic methods are highly desirable.

A fairly recent methodology aimed towards addressing complex models and specifications for control systems involves the notion of a hybrid system. Informally, a hybrid system consists of a collection of discrete modes, where each mode contains a continuous time model of the dynamics and a specified region of operation. Moreover, transitions between modes are enabled only when the continuous behavior achieves a certain condition, such as reaching the boundary of the region of operation. In this way, complex specifications can be formally accommodated during the design process of the hybrid system.

In this thesis, we explore these ideas in the context of developing a novel framework for the au-

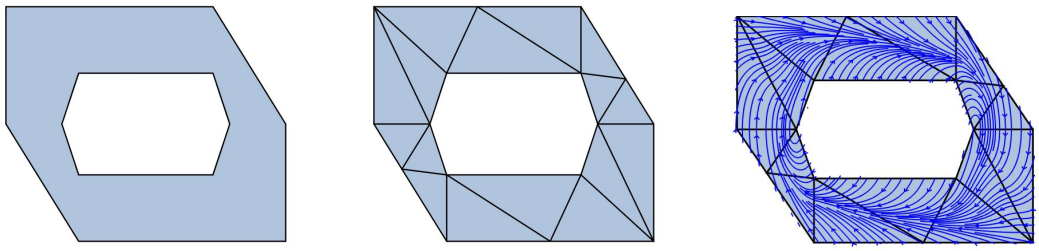

Figure 1.1: Typical workflow for the design of hybrid controllers. First, the safety constraints are modelled on the state space (left). Next, the safety region is partitioned into polytopes, typically simplices (middle). Finally, feedback controllers are designed on each polytope, driving trajectories along a specified sequence of polytopes (right).

tonomous motion planning and control of multiple robotic agents. There is a wealth of literature on this fundamental problem in robotics, which involves many aspects such as perception, mapping, state estimation, planning, and controller design. However, relatively few works have focused on ensuring the compatibility between all these components, especially on real systems. The hybrid system formalism provides a suitable medium in which these compatibilities can be described and analyzed. In this work, we focus primarily on the tight integration of the planning and control components, and make basic assumptions on the other components. We also aim to provide a framework of sufficient modularity so that existing tools for each of the components can be integrated in an effective and consistent way. Inspired by the techniques in the literature, we have also implemented our own versions of the controller design and planning modules to highlight the customizability of our approach as well as its effectiveness on a collection of quadrocopters.

1.1 Hybrid Systems Paradigm

Hybrid systems fall into the broader class of partition-based methods, which are based on modelling the system dynamics and safety constraints, followed by subdividing the problem into smaller components [47]. First, the safety constraints are modelled on the state space of the system. Second, the safety region is partitioned into polytopes, inducing a sequence of polytopes for the required temporal sequencing of the control specification. Finally, a feedback controller is designed on each polytope, driving state trajectories along the required sequence of polytopes. Figure 1.1 illustrates the main steps. The methodology is fully general and is not restricted to just the robotics domain, although it typically assumes a priori knowledge of the system dynamics and safety constraints as well as complete knowledge of the state during execution.

A number of works have targeted the various components of this methodology. Starting from the

lowest level, the reachability problem on polytopes has been studied extensively [46, 90, 16, 18]. The Reach Control Problem (RCP) for affine systems has proved to be a powerful method for synthesizing feedback controllers within a hybrid system framework. It assumes a given polytope within the temporal sequence, so that facets are designated either as restricted or exit facets. The main goal is to design a feedback controller so that all state trajectories within the polytope leave through only the exit facet in finite time. By exploiting convexity in the problem, existence of such controllers as well as their construction relies only on the polytope vertices and the system dynamics. Specialized results are available when the polytope is a simplex. Various other low level controller design methods also exist, based on potential fields [25] and reachability analysis [39].

The partitioning and sequencing of polytopes is closely intertwined and has typically been performed manually or through the guide of the control specifications. Several works have devised automated partition and sequencing strategies based on Linear Temporal Logic (LTL) specifications [58, 32, 42], relying on the existing reachability-based controller synthesis techniques, and have also been implemented in the robotics domain [11]. Our first implementation of these techniques also appears in our first experimental work [108].

There are two main drawbacks to the techniques described above. First, to the author's knowledge there does not exist a complete systematic methodology for constructing a partition into polytopes and a temporal sequencing such that the reachability problem is guaranteed to be solvable on each polytope. Typically, the partition and temporal sequence are designed first, followed by verification of reachability on each polytope. If the latter fails on even one polytope, the partition and sequencing step must be revised. Second, as the state space dimension grows due to modelling complexity, the methodology suffers from the curse of dimensionality because the number of polytopes grows exponentially.

To address the first issue, necessary and sufficient conditions for the solvability of the RCP have been investigated. Recently, work on affine and topological obstructions have attempted to fully characterize the issue [77, 82]. However, these considerations still do not provide a constructive procedure for obtaining a sequence of polytopes in the state space. The second issue has received even less consideration.

1.2 Proposed Approaches

Our work has been motivated by the two issues described above of systematic construction and scalability. We initially began with an investigation of using optimal control to devise an alternative way of obtaining reach controllers on higher dimensional systems. A second direction was later pursued to address both issues simultaneously, leading to our modular framework for multi-agent systems.

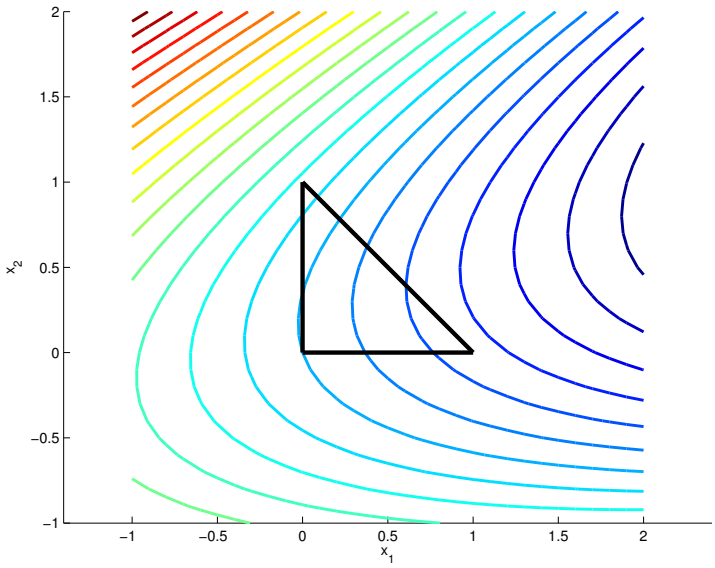

Figure 1.2: Cost level sets generated by a parabolic cost with decreasing level sets (red to blue) aligned towards the exit facet (diagonal).

1.2.1 An Optimal Control Approach to RCP

Suppose that a sequence of contiguous simplices has been specified in the state space corresponding to some complex control specification. As discussed earlier, a typical hybrid control scheme involves solving the RCP on each simplex, which generates feedback controllers driving trajectories from one simplex to the next. In higher dimensional state spaces, the hard safety constraints imposed by the RCP may result in the lack of existence of a controller over some simplex in the sequence. This led to the idea of using optimal control as an alternative method for generating feedback controllers over simplices.

Since the RCP is concerned with affine systems on simplices, we restricted ourselves to considering cost functionals similar to the standard linear quadratic optimal control formulation, but with features that would simulate the effects of solving the RCP. Namely, if we were to view the cost level sets in the state space over a given simplex, we would expect decreasing level sets in the direction of the exit facet, and increasing level sets as we approach the restricted facets, see Figure 1.2. In analogy to standard optimal control where the designer can tradeoff between the relative importance of regulation to the equilibrium and control effort, ideally we would obtain a form for the cost that can easily allow the designer to tradeoff between the relative importance of reaching the exit facet, not exiting through restricted facets, and also the control effort. This idea is somewhat reminiscent of barrier functions often encountered in optimization [14]. However, barrier functions are nonlinear and we would like for the resulting optimal control to be an affine feedback, as obtained when solving the RCP.

In formulating an appropriate cost, it was soon discovered that there was a relationship to existing

literature on indefinite linear quadratic optimal control [114, 105], which was less understood but more general than the standard linear quadratic problem. For a given cost, it is necessary to be able to characterize when the optimal control problem is solvable, and if so, to calculate an optimal control. Surprisingly, none of the existing results provided a solution for the costs we were interested in. Figure 1.2 shows an example with parabolic cost level sets that exhibits the desired features with respect to a simplex. Viewing this as an opportunity, we diverted our attention from the original motivation of hybrid control onto this long-standing unresolved theoretical problem, and solved it. Details are given in Chapter 3.

1.2.2 Modular Framework for Motion Planning

The hybrid system paradigm described above is very general and has potentially a wide range of application domains. Perhaps one source of the drawbacks on systematic construction and scalability is that the general hybrid system framework aims to be too general. Our idea was that by specializing to a particular domain and by introducing suitable assumptions, we could exploit additional structure in the problem to overcome these drawbacks.

To this end, we restricted our attention mainly to control of robotic agents, or more specifically, general nonlinear systems with symmetries in the output space (typically the positions of the agents). We introduced several assumptions to help streamline the design of the underlying hybrid system:

(i) The control specifications are described only on the lower-dimensional output space of the system.

(ii) The partition of the safety region, described in the output space, consists of a uniform grid of boxes.

(iii) A finite number of pre-computed feedback controllers is available to be implemented on any box.

These assumptions are fairly reasonable for the robotics domain, as we now justify. First, (i) implies that the designer only needs to specify the higher level task of where vehicles must go; additional constraints such as velocity and actuation limits are addressed within the implementation of low-level controllers. Second, (ii) enables us to avoid the computational complexity associated with storing the details of an arbitrary partition; moreover, it becomes very easy to determine in real-time which element of the partition the system is in. Finally, (iii) enables us to systematize the implementation of feedback controllers on any element of the partition, as well as the transition between controllers.

Together, these assumptions imply that we can design a finite number of feedback controllers, called *motion primitives*, over a single box of the partition, and then apply them to any box. The allowable

transitions between motion primitives is modelled by a hybrid system called the *maneuver automaton*, similarly to [33] from which we adopted this term. Moreover, motion primitives can often be designed for independent subsystems and then composed, as in the case of a multi-robot system. These notions enable us to simultaneously address both the issues of systematic construction and scalability.

Our approach is modular, theoretically rigorous, and experimentally validated. Modularity refers to the feature that the design of motion primitives (controller design) is decoupled from the selection of motion primitives over the grid to achieve the temporal specification (planning). As such, the designer may employ different synthesis techniques on these components. For example, in our applications we use the RCP to design motion primitives and standard graph search algorithms for planning. Theoretical rigor refers to the compatibility that we establish between the planning and control design components in order to enforce safety and satisfaction of the overall specification. Finally, these ideas have been demonstrated on a variety of experiments involving up to eight quadrocopters, showing that these ideas apply in practice.

While there are several existing works that have explored the idea of motion primitives [33, 93], to the author's knowledge this work is the first rigorous treatment of feedback-based motion primitives on a gridded output space. Chapter 4 provides details, including a comparison to related literature.

Given that there is a wealth of literature for synthesizing feedback controllers at the low-level and discrete planning algorithms at the high-level, it was observed that the "middle-layer" gluing the two levels had the most potential for additional research and impact. Recognizing the uniqueness of our approach in combining motion primitives within the hybrid systems framework, we extended these ideas in the direction of obtaining a multi-hierarchy of motion primitives. Hierarchy has been explored in a variety of forms and contexts [74, 53]. Our work is the first to consider a multi-hierarchy of motion primitives on a gridded output space. Chapter 5 provides details on this direction.

1.3 Thesis Organization

This thesis is mainly organized according to how the ideas developed. We began with tools such as the Reach Control Problem and optimal control, and first focused on low-level control synthesis (Chapter 3). Realizing some of the limitations for real world problems, we broadened our scope by using the general framework of hybrid systems and by exploiting symmetries, modularity, and eventually hierarchy, to address simultaneously low-level control design and higher-level planning (Chapters 4 and 5). We now highlight the contents of each chapter in more detail.

Chapter 2 covers basic mathematical notions used in the remainder of the thesis. We note that it

is not comprehensive, and instead serves to establish basic results and notation needed for the later chapters.

Chapter 3, based on [110], is devoted to addressing a gap we identified in the existing literature on indefinite linear quadratic optimal control. We begin by reviewing the related literature and showing that although the problem was solved under the assumption of controllability [105], the case when the dynamics are merely stabilizable was unresolved and fundamentally a non-trivial extension. We then motivate the problem in terms of the hybrid control scheme and the Reach Control Problem. Following a formal problem statement and some preliminaries, we solve the main problem, which involves characterizing the existence of optimal controls and calculating the optimal feedback controller. Afterwards, we show that our result recovers known classical results in the optimal control literature. Although not presented in [110], we provide two numerical examples that highlight the utility of our main results. Finally, we conclude the chapter and make a brief remark on its applicability in the context of hybrid control.

Chapter 4, based on [111], presents the modular framework for motion planning with feedback-based motion primitives. First the work is motivated and compared within the existing robotics literature in order to highlight the areas of novelty. A formal problem statement is given on the motion planning problem of reach-avoid, where the system must safely reach a target region. Then we present the components of the modular framework: the partition of the output space; motion primitives and the maneuver automaton; the product automaton of the partitioned output space and the maneuver automaton; and the high-level plan. Using the modular framework, we provide a set of design constraints on the individual components, a set of initial conditions, and a feedback controller that solves the reach-avoid problem. Following this main result, we concentrate on motion primitives and maneuver automata, first describing the parallel composition of maneuver automata, and then providing a specific design of motion primitives for integrator systems. We then describe how these motion primitives can be parallel composed to control a multi-agent system, and we provide three illustrative implementations of a high-level plan. Finally, the methodology is demonstrated experimentally on quadrocopters in various scenarios.

Chapter 5, based on [112], is a natural extension of Chapter 4 towards a multi-hierarchy of motion primitives. The main objective of this work was to provide the designer of motion primitives with a concrete set of rules in which to explore new designs that can lead to better organization and significant computational savings. Although Chapter 5 is organized similarly and generalizes the results of Chapter 4, it is presented as independently as possible and contains many notable differences, which we now highlight. First, the work is again motivated and compared with literature, emphasizing new works in the context of hierarchy and abstractions. Next, we present a novel variation to the reach-avoid objective,

which includes the notion of a behavioral constraint. In contrast to Chapter 4, the framework is now structured as a collection of maneuver automata at various hierarchical levels. This framework is used to solve the main problem, which again involves providing a set of conditions on the design of motion primitives at each hierarchical level, a set of initial conditions, and a hierarchical feedback controller. Following the main result, we sketch out some composition procedures at higher levels and provide a specific design of higher level motion primitives. We then describe two different strategies of hierarchy applied to multi-agent control, resulting in very efficient methods for formation control and formation morphing. These strategies are demonstrated experimentally on quadrocopters in various scenarios, which are much more complex than those considered in Chapter 4.

Chapter 6 supplies some analysis of the motion primitives introduced in Chapter 5, motivated from an observed phenomenon of convergence to a limit cycle behavior. The problem is introduced, followed by some preliminaries on the underlying motion primitives. A specialized result is proven for a collection of single integrators using the contraction principle from real analysis, showing the existence of a limit cycle in certain settings. The result is conjectured to be true for a collection for double integrators. Finally, Chapter 7 concludes the thesis with a summary and some remarks on possible future research directions.

1.4 Main Contributions

1.4.1 Chapter 3

The principal contributions of this chapter are the identification of an open gap in the existing literature on linear quadratic optimal control with indefinite cost functionals, and the solution to this open problem. More specifically:

- Theorem 3.5.9 gives necessary and sufficient conditions for the existence of optimal controls, and the form of the optimal feedback controller.
- Section 3.6 discusses how our general theory recovers known results in linear quadratic optimal control as special cases.

1.4.2 Chapter 4

The principal contributions of this chapter are the formulation of a modular framework for motion planning, the rigorous proof that the framework can solve the reach-avoid objective, and experimental work demonstrating its effectiveness. Importantly, we note that this chapter shares a significant portion

of the content presented earlier in a chapter of the thesis of Zach Kroeze [61], who was a coauthor for [109, 111]. While it is difficult to split the contributions exactly as the effort was highly collaborative, we highlight the main contributions of the author of this thesis:

- Theorem 4.5.1 ties together the various modules and design assumptions in order to solve the reach-avoid problem.
- Theorem 4.6.2 proves that our proposed construction for parallel composition of maneuver automata preserves the design assumptions.
- Section 4.7 includes the design of various motion primitives along with a formal verification of the design assumptions, and features a novel extension to multi-speed motion primitives.
- Section 4.8 discusses various implementation details for high-level planning and illustrates the results experimentally on up to eight Crazyflie quadrocopters. The associated videos are:
  - http://tiny.cc/modular-3alg,
  - http://tiny.cc/quad5scenes, and
  - http://tiny.cc/quadrocopterPlanar.

1.4.3 Chapter 5

The principal contributions of this chapter are the extension of the modular framework into a multi-level hierarchy, the rigorous proof on its correctness, and experimental work demonstrating its additional benefits over the plain modular framework. More specifically:

- Theorem 5.5.2 ties together all the levels in the hierarchy and design assumptions in order to solve the reach-avoid problem with behavioral constraints.
- Section 5.7 provides the design of higher level motion primitives, which are used for multiple agents to maintain a formation.
- Section 5.8 discusses various implementation details for using the proposed higher level motion primitives for formation flight and formation morphing, while Section 5.9 illustrates the results experimentally on up to eight Crazyflie quadrocopters. An associated video can be found at http://tiny.cc/hier-moprim.

1.4.4 Chapter 6

The principal contributions of this chapter are the motivation for the observed hybrid limit cycle, the precise mathematical formulation, and a proof of its existence. More specifically:

- Theorem 6.3.1 proves the existence of a hybrid limit cycle for a collection of single integrators executing a constant sequence of formation motion primitives.

Chapter 2

Mathematical Preliminaries

In this chapter we establish notation and recount some notions that are used throughout the thesis. We also discuss modelling of quadrotors. For the sake of brevity and understanding, we only identify those concepts that are not already well-known in the control-theoretic and basic mathematical literature. Standard notions of topology are assumed. Some specific results are given that will be needed later; results with proofs are contributed by the author (Lemmas 2.2.1 and 2.4.2). A list of notation is provided at the beginning of the thesis.

2.1 Basic Notation

Let $\mathbb{Z}$ denote the integers and $\mathbb{R}$ denote the real numbers. Let $|\cdot|$ denote the cardinality of a set. If A is a set, let $\mathcal{P}(A)$ denote its power set. The set difference of A and B is denoted $A \setminus B$. For a collection of sets $\{A_i\}_{i=1}^n$, the cartesian product is denoted $\prod_{i=1}^n A_i$; when $n = 2$, we may write $A_1 \times A_2$, and when $A_i = A$ for all $i = 1, \ldots, n$, we may write A^n (which in general should not be confused with superscripts for indexing in other contexts). Given a function $f : A \to B$, the image of $A_1 \subset A$ under f and the preimage of $B_1 \subset B$ under f are defined in the usual way, and are denoted as $f(A_1) \subset B$ and $f^{-1}(B_1) \subset A$, respectively. Let $\mathrm{co}\{v_1, \ldots, v_m\}$ denote the convex hull of the vectors $v_1, \ldots, v_m \in \mathbb{R}^n$. Given two vectors $v, w \in \mathbb{R}^n$, we denote the component-wise multiplication (or Hadamard product) as $v \circ w$. Let $\mathcal{X}(\mathbb{R}^n)$ denote the set of globally Lipschitz vector fields on $\mathbb{R}^n$. Let $\Re(s)$ denote the real part of a complex number s. Let I_n be the $n \times n$ identity matrix (the subscript is omitted if the dimension is clear from the context). Let $P^\dagger$ denote the (unique) pseudo-inverse of $P \in \mathbb{R}^{n \times m}$. The set of eigenvalues of $A \in \mathbb{R}^{n \times n}$ is denoted by $\sigma(A)$.

2.2 Linear Geometric Theory

The following notions are needed for Chapter 3.

A subspace $\mathcal{V} \subset \mathbb{R}^n$ is $A-$invariant if $A\mathcal{V} \subset \mathcal{V}$. We use the following subsets of the complex plane: $\mathbb{C}^- := \{s \in \mathbb{C} \mid \Re(s) < 0\}$, $\mathbb{C}^0 := \{s \in \mathbb{C} \mid \Re(s) = 0\}$, and $\mathbb{C}^+ := \{s \in \mathbb{C} \mid \Re(s) > 0\}$. Given a real monic polynomial p there is a unique factorization $p = p_- \cdot p_0 \cdot p_+$ into real monic polynomials with p_-, p_0, and p_+ having all roots in $\mathbb{C}^-$, $\mathbb{C}^0$, and $\mathbb{C}^+$, respectively. Then if $A \in \mathbb{R}^{n\times n}$ and if p is its characteristic polynomial, then we define the spectral subspaces $\mathcal{X}^-(A) := \mathrm{Ker}(p_-(A))$, $\mathcal{X}^0(A) := \mathrm{Ker}(p_0(A))$, and $\mathcal{X}^+(A) := \mathrm{Ker}(p_+(A))$. Each of these subspaces are $A-$invariant and the restriction of A to $\mathcal{X}^-(A)(\mathcal{X}^0(A), \mathcal{X}^+(A))$ has characteristic polynomial $p_-(p_0, p_+)$. For two subspaces $\mathcal{V}$ and $\mathcal{W}$, let $\mathcal{V} \oplus \mathcal{W}$ denote their direct sum and let $\mathcal{V} \sim \mathcal{W}$ denote that they are isomorphic. For an arbitrary matrix $A \in \mathbb{R}^{n\times n}$ and subspace $\mathcal{V} \subset \mathbb{R}^n$ we define the subspace $\langle A \mid \mathcal{V}\rangle := \mathcal{V} + A\mathcal{V} + \ldots A^{n-1}\mathcal{V}$, and by further writing $\mathcal{V} = \mathrm{Ker}(W)$ for some $W \in \mathbb{R}^{p\times n}$ we also define $\langle \mathcal{V} \mid A\rangle := \mathrm{Ker}(W) \cap \mathrm{Ker}(WA) \ldots \cap \mathrm{Ker}(WA^{n-1})$. For a linear time-invariant system, $\dot{x} = Ax + Bu$, the controllable subspace will be denoted in the usual way $\langle A \mid \mathrm{Im}(B)\rangle$. If there is an output $y = Cx$, then $\langle \mathrm{Ker}(C) \mid A\rangle$ denotes the unobservable subspace of (C, A). If M is a real $n \times n$ matrix and $\mathcal{V}$ is a subspace of $\mathbb{R}^n$, then $M^{-1}\mathcal{V} := \{x \in \mathbb{R}^n \mid Mx \in \mathcal{V}\}$. If $\mathcal{V}$ is a subspace of $\mathbb{R}^n$ then $\mathcal{V}^\perp$ denotes its orthogonal complement with respect to the standard Euclidean inner product.

The following results on observability can be established.

Lemma 2.2.1. *Let* $A = \begin{bmatrix} A_1 & A_{12} \\ 0 & A_2 \end{bmatrix}$ *with* $\sigma(A_2) \subset \mathbb{C}^-$ *and* $C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$. *Then*

(i) All of the eigenvalues of A on the imaginary axis are (C, A) observable if and only if all of the eigenvalues of A_1 on the imaginary axis are (C_1, A_1) observable.

(ii) An eigenvalue of A is (C, A) observable if and only if it is $(C^\top C, A)$ observable.

Proof. Recall that an eigenvalue λ of A is said to be (C, A) observable if $\mathrm{rank} \begin{bmatrix} A - \lambda I \\ C \end{bmatrix} = n$ (equivalently, $\mathrm{Ker}(A - \lambda I) \cap \mathrm{Ker}(C) = 0$), see page 46 of [106].

(i) ($\Rightarrow$) Assume that $\mathrm{Ker}(A-\lambda I)\cap\mathrm{Ker}(C) = 0$ for all $\lambda \in \sigma(A)$ such that $\Re(\lambda) = 0$. Let $\lambda \in \sigma(A_1)$ and $\Re(\lambda) = 0$, and show $\mathrm{Ker}(A_1-\lambda I)\cap\mathrm{Ker}(C_1) = 0$. Equivalently, let p_1 satisfy $A_1 p_1 = \lambda p_1$ and $C_1 p_1 = 0$, and show that $p_1 = 0$. Now we define $p = \begin{bmatrix} p_1 \\ 0 \end{bmatrix}$, and first show that $p \in \mathrm{Ker}(A - \lambda I) \cap \mathrm{Ker}(C)$.

We have

$$(A-\lambda I)p = \begin{bmatrix} A_1-\lambda I & A_{12} \\ 0 & A_2-\lambda I \end{bmatrix} \begin{bmatrix} p_1 \\ 0 \end{bmatrix} = \begin{bmatrix} (A_1-\lambda)p_1 \\ 0 \end{bmatrix} = 0.$$

since $A_1 p_1 = \lambda p_1$. Also $Cp = C_1 p_1 = 0$. Thus $p \in \mathrm{Ker}(A-\lambda I) \cap \mathrm{Ker}(C)$. Since $\mathrm{Ker}(A-\lambda I) \cap \mathrm{Ker}(C) = 0$, $\lambda \in \sigma(A_1) \subset \sigma(A)$, and $\Re(\lambda) = 0$, we can use the assumption to conclude that $p = 0$ and hence $p_1 = 0$.

($\Leftarrow$) Assume that $\mathrm{Ker}(A_1 - \lambda I) \cap \mathrm{Ker}(C_1) = 0$ for all $\lambda \in \sigma(A_1)$ such that $\Re(\lambda) = 0$. Let $\lambda \in \sigma(A)$ and $\Re(\lambda) = 0$, and show $\mathrm{Ker}(A-\lambda I) \cap \mathrm{Ker}(C) = 0$. Equivalently, let p satisfy $Ap = \lambda p$ and $Cp = 0$, and show that $p = 0$. Write $p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$. Then

$$Ap = \lambda p \Rightarrow \begin{bmatrix} A_1-\lambda I & A_{12} \\ 0 & A_2-\lambda I \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = 0$$

Since $\sigma(A_2) \subset \mathbb{C}^-$, $\sigma(A) = \sigma(A_1) \uplus \sigma(A_2)$, and $\Re(\lambda) = 0$, we have $\lambda \in \sigma(A_1)$ and $\lambda \notin \sigma(A_2)$. Thus the second equation $A_2 p_2 = \lambda p_2$ implies $p_2 = 0$. Then the first equation reduces to $A_1 p_1 = \lambda p_1$. Also $Cp = 0$ reduces to $C_1 p_1 = 0$. We have established that $p_1 \in \mathrm{Ker}(A_1 - \lambda I) \cap \mathrm{Ker}(C_1)$, which by assumption is the trivial subspace and hence $p_1 = 0$. Together $p_1 = 0$ and $p_2 = 0$ imply $p = 0$, as desired.

(ii) Using the equivalent characterization, it is enough to show that $\mathrm{Ker}C = \mathrm{Ker}(C^\top C)$. If $Cx = 0$, then $C^\top Cx = 0$. Conversely, let $C^\top Cx = 0$. We have that $(Cx)^2 = x^\top C^\top Cx = 0$. Thus $Cx = 0$.

□

2.3 Real Analysis

The material below is needed for Chapters 3 and 6.

Let $\mathbb{R}^+ := \{t \in \mathbb{R} \mid t \geq 0\}$ and $\mathbb{R}^e := \mathbb{R} \cup \{-\infty, +\infty\}$. Additionally, given a function $f : \mathbb{R} \to \mathbb{R}$, the statement that $\lim_{t\to\infty} f(t)$ exists in $\mathbb{R}^e$ means that $\lim_{t\to\infty} f(t)$ is either equal to a real number, ∞, or $-\infty$ in the usual sense.

We denote the space of all measurable vector-valued functions on $\mathbb{R}^+$ that are locally square integrable as $L^m_{2,loc}(\mathbb{R}^+) = \left\{ u : \mathbb{R}^+ \to \mathbb{R}^m \mid (\forall T \geq 0) \int_0^T u(t)^\top u(t)\, dt < \infty \right\}$. Let $d_{\mathcal{L}} : \mathbb{R}^n \to [0, \infty)$ denote the function giving the minimum Euclidean distance from a point to a set $\mathcal{L} \subset \mathbb{R}^n$.

We also recall the Mean Value Theorem (see Theorem 3 in Chapter 3 of [86]). Let $f : [a, b] \to \mathbb{R}$ be a continuous function that is differentiable on (a, b). There exists a point $c \in (a, b)$ such that $f(b) - f(a) = f'(c)(b - a)$.

Finally, we make use of the well known contraction principle. The following is taken from Exercise 27c in Chapter 4 of [86], which is a more loose statement than the standard contraction principle (compare with Theorem 24 in Chapter 4).

Definition 2.3.1. *Let M be a metric space with metric $d : M \times M \to \mathbb{R}$. A* weak contraction *of M is a mapping $f : M \to M$ such that for all $x, y \in M$, if $x \neq y$, then $d(f(x), f(y)) < d(x, y)$. If there exists some $x^* \in M$ such that $f(x^*) = x^*$, then x^* is a* fixed-point *of f.*

Theorem 2.3.2 ((Weak) Contraction Principle)**.** *Suppose that $f : M \to M$ is a weak contraction and the metric space M is compact. Then f has a unique fixed-point, x^*, and for all $x \in M$, the iterate $f^n(x) = f \circ \cdots \circ f(x)$ converges to x^* as $n \to \infty$.*

2.4 Symmetric Matrices

The material in this section is needed for Chapter 3.

Given a quadratic form on $\mathbb{R}^n$, $\omega : \mathbb{R}^n \to \mathbb{R}$, it is said to be *positive definite* if for all $x \in \mathbb{R}^n$, $\omega(x) \geq 0$, and $\omega(x) = 0$ if and only if $x = 0$; *positive semidefinite* if for all $x \in \mathbb{R}^n$, $\omega(x) \geq 0$; *negative definite* if $-\omega$ is positive definite; *negative semidefinite* if $-\omega$ is positive semidefinite; and *indefinite* if ω is neither positive semidefinite nor negative semidefinite. Writing $\omega(x) := x^\top P x$ for some symmetric matrix $P \in \mathbb{R}^{n \times n}$, we say that the matrix P is positive definite if the quadratic form ω is positive definite and so on. We write $P > 0$, $P \geq 0$, $P < 0$, and $P \leq 0$ if the matrix is positive definite, positive semidefinite, negative definite, and negative semidefinite, respectively. Given symmetric matrices $P, Q \in \mathbb{R}^{n \times n}$, we write $P < Q$ if $Q - P > 0$, and likewise for the other inequalities.

Let Λ denote a subset of the set of all symmetric matrices in $\mathbb{R}^{n \times n}$. We say that M^+ *(M^-) is the maximal (minimal) element on* Λ if $M^+ \in \Lambda$ ($M^- \in \Lambda$) and for all $M \in \Lambda$, $M \leq M^+$ ($M \geq M^-$). The maximal and minimal elements, which are called the *extremal elements on* Λ, are unique if they exist since Λ forms a partially ordered set.

The following standard result relates the positive semidefiniteness of a matrix in terms of its block components.

Theorem 2.4.1 (Theorem 1, [2])**.** *Given a real symmetric matrix* $P = \begin{bmatrix} P_1 & P_{12} \\ P_{12}^\top & P_2 \end{bmatrix}$*, the following conditions are equivalent:*

1. $P \geq 0$.

2. $P_1 \geq 0, \;\; (I - P_1 P_1^\dagger) P_{12} = 0, \;\; P_2 - P_{12}^\top P_1^\dagger P_{12} \geq 0$.

3. $P_2 \geq 0, \;\; (I - P_2 P_2^\dagger) P_{12}^\top = 0, \;\; P_1 - P_{12} P_2^\dagger P_{12}^\top \geq 0$.

The following result is easily established.

Lemma 2.4.2. *Let M be a symmetric positive semidefinite matrix with the block form*

$$M = \begin{bmatrix} M_1 & M_{12} \\ M_{12}^\top & M_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & M_{12,1} \\ 0 & M_{1,22} & M_{12,2} \\ M_{12,1}^\top & M_{12,2}^\top & M_2 \end{bmatrix}.$$

Then $M_{12,1} = 0$.

Proof. Since $M \geq 0$, Theorem 2.4.1 in particular implies that $(I - M_1 M_1^\dagger) M_{12} = 0$. Using the properties of the pseudo-inverse, it can be shown that $M_1^\dagger = \begin{bmatrix} 0 & 0 \\ 0 & M_{1,22}^\dagger \end{bmatrix}$. Then the result follows from

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \left(I - \begin{bmatrix} 0 & 0 \\ 0 & M_{1,22} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & M_{1,22}^\dagger \end{bmatrix} \right) \begin{bmatrix} M_{12,1} \\ M_{12,2} \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I - M_{1,22} M_{1,22}^\dagger \end{bmatrix} \begin{bmatrix} M_{12,1} \\ M_{12,2} \end{bmatrix}.$$

□

2.5 Discrete and Hybrid Systems

In our work on motion planning, we describe the relevant structures typically as a discrete or hybrid system. The following notions are used in Chapters 4 and 5.

Informally a discrete system contains nodes, edges, and occasionally additional structure such as labels, relations, or initial conditions. Alternative names include transition system, graph, or automaton; as there are many variations on the definitions, we use the formulation from [3] as a basic reference.

Definition 2.5.1 (Transition System)**.** *A transition system $T = (Q, \Pi, \to, \models, Q_0)$ consists of*

- *a (possibly infinite) set Q of states;*
- *a finite alphabet Π of propositions;*
- *a transition relation $\to \subset Q \times Q$;*

- *a satisfaction relation* $\models \subset Q \times \Pi$*;*
- *a set* $Q_0 \subset Q$ *of initial states.*

Next, a hybrid system contains the basic elements of a transition system, but associates continuous-time data to each discrete state. Again, many variations exist, so we refer to [3] here.

Definition 2.5.2 (Hybrid System)**.** *A hybrid system is a tuple* $H = (V, n, X_0, F, Inv, R)$ *where*

- V *is a finite set of locations, and* $n \geq 0$ *is the dimension of* H*. The state space is* $X = V \times \mathbb{R}^n$*. Each state has the form* (l, x)*, where* $l \in V$ *is the discrete part and* $x \in \mathbb{R}^n$ *is the continuous part.*
- $X_0 \subset X$ *is the set of initial states.*
- $F : X \to \mathcal{P}(\mathbb{R}^n)$ *assigns to each state* $(l, x) \in X$ *a set* $F(l, x) \subset \mathbb{R}^n$*, which constrains the time derivative of the continuous part of the state; in the discrete location* l*, the continuous part of the state satisfies the differential inclusion* $\dot{x} \in F(l, x)$*.*
- $Inv : V \to \mathcal{P}(\mathbb{R}^n)$ *assigns to each location* $l \in V$ *an invariant set* $Inv(l) \subset \mathbb{R}^n$*, which constrains the value of the continuous part of the state while the discrete part is* l*.*
- $R \subset X \times X$ *is a relation capturing the discontinuous state changes.*

Semantics of a hybrid system involve the notion of trajectories, often called executions. Executions begin at an initial state in X_0. The continuous component evolves according to the differential inclusion F within the invariant Inv while the discrete component makes discrete jumps according to R. More formal details on our specialized hybrid system (called a Maneuver Automaton) and its executions will be presented in Chapters 4 and 5.

Typically a linear temporal logic (LTL) formula is specified over a transition system [3, 58, 116]. Although we refer to LTL many times throughout the thesis, we omit these details since we do not explicitly study a problem with LTL specifications.

2.6 Reach Control

Reach Control is used in the design of motion primitives in Chapter 4. We review only the aspects that are used directly, which are adapted from [51, 82].

Let $\mathcal{S} := \text{co}\{v_0, v_1, \ldots, v_n\} \subset \mathbb{R}^n$ be an n-dimensional simplex with vertices $v_0, \ldots, v_n$. Its facets shall be denoted by $\mathcal{F}_0, \ldots, \mathcal{F}_n$, where each facet is indexed by the vertex it does not contain. Furthermore,

for each $i \in \{0, \dots, n\}$ let h_i be the unit normal vector to the facet $\mathcal{F}_i$ pointing outside the simplex. Let $I_i := \{1, \dots, n\} \setminus \{i\}$.

Consider the system

$$\dot{x} = Ax + Bu + a, \quad x \in \mathcal{S}, \quad u \in \mathbb{R}^m, \tag{2.1}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $a \in \mathbb{R}^n$. Let $\phi_u(\cdot, x_0)$ be the trajectory generated by system (2.1), with control law u and initial condition x_0.

The Reach Control Problem aims to find a closed-loop control feedback which results in every trajectory of (2.1) leaving $\mathcal{S}$ through $\mathcal{F}_0$ in finite time.

Problem 2.6.1 (Reach Control Problem (RCP))**.** *Consider system* (2.1) *defined on a simplex* $\mathcal{S}$*. Find a state feedback* $u(x)$ *such that for every* $x_0 \in \mathcal{S}$*, there exist* $T \geq 0$ *and* $\varepsilon > 0$ *such that*

(i) $\phi_u(t, x_0) \in \mathcal{S}$ *for all* $t \in [0, T]$*.*

(ii) $\phi_u(T, x_0) \in \mathcal{F}_0$*.*

(iii) $\phi_u(T + \varepsilon, x_0) \notin \mathcal{S}$ *for all* $t \in (T, T + \varepsilon)$*.*

The main result is the following:

Theorem 2.6.2. *Given the system* (2.1) *on a simplex* $\mathcal{S}$ *and an affine feedback* $u(x) = Kx + g$*, then* $u(x)$ *solves the RCP if and only if*

- *The invariance conditions hold:*

$$h_j \cdot (Ax + Bu(x) + a) \leq 0, \;\; j \in I_i, \;\; i \in \{0, \dots, n\}, \;\; x \in \mathcal{S}. \tag{2.2}$$

- *There is no equilibrium in* $\mathcal{S}$*.*

Using the affine feedback $u(x) = Kx + g$, let $u_i := u(v_i)$ for $i \in \{0, \dots, n\}$. The following relationship is useful for designing the gains K and g:

$$\begin{bmatrix} K^\top \\ g^\top \end{bmatrix} = \begin{bmatrix} v_0^\top & 1 \\ \vdots & \vdots \\ v_n^\top & 1 \end{bmatrix}^{-1} \begin{bmatrix} u_0^\top \\ \vdots \\ u_n^\top \end{bmatrix}. \tag{2.3}$$

2.7 Quadrotor Modelling and Control

In this section we describe a standard model for a quadrotor and some basic techniques for control.

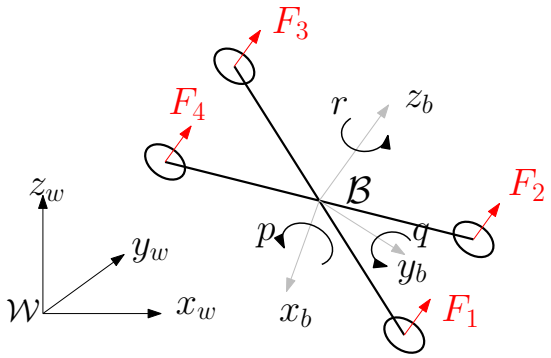

Figure 2.1: The inertial world frame $\mathcal{W}$ and quadrocopter body-fixed frame $\mathcal{B}$ are shown. The quadrocopter is actuated by varying the thrusts F_i, $i \in \{1, 2, 3, 4\}$ produced by each motor. This results in changes to its body rotation rates, (p, q, r) and vertical acceleration, which then causes a change to the quadrocopter's position and attitude.

2.7.1 Model

The standard model is discussed in many works [73, 91, 69, 122]; this section is adapted from [73]. The quadrotor has three positional and three rotational degrees of freedom. Define a world frame $\mathcal{W}$ and vehicle body frame $\mathcal{B}$, as shown in Figure 2.1. The position in the world frame is $r = (x, y, z)$, and time derivatives are denoted with overhead dots. We use the ZXY Euler angles to define the roll, pitch, and yaw angles (ϕ, θ, ψ), although other conventions may be used. The rotation matrix from $\mathcal{B}$ to $\mathcal{W}$ is given by R_{wb}. The angular velocity of $\mathcal{B}$ with respect to $\mathcal{W}$ is $\omega = (p, q, r)$, with components expressed in $\mathcal{B}$; it can be related to the time derivatives of the Euler angles. Each vehicle rotor, $i \in \{1, 2, 3, 4\}$, has an angular speed ω_i and produces a force $F_i = k_F \omega_i^2$ and moment $M_i = k_M \omega_i^2$, for constants k_F and k_M. Motor dynamics are relatively fast and are ignored, and aerodynamic effects are neglected. The control input to the system can be written as $u = (u_1, u_2, u_3, u_4)$, where u_1 is the net body force (thrust) and $u_M = (u_2, u_3, u_4)$ are the net body moments, and they can be mapped to the rotor speeds.

Using Newton's equations of motion, the translational equation is

$$m\ddot{r} = -mge_3 + u_1 R_{wb} e_3$$

where m is the vehicle mass, g is the acceleration due to gravity and $e_3 = (0, 0, 1)$. The forces are gravity and the sum of the forces of the rotors in the vertical direction of the body frame. The rotational equation is

$$\dot{\omega} = I^{-1} \left(-\omega \times I\omega + u_M \right),$$

where I is the moment of inertia matrix at the center of mass along the $\mathcal{B}$ axes. The full state is $(r, \dot{r}, R_{wb}, \omega)$.

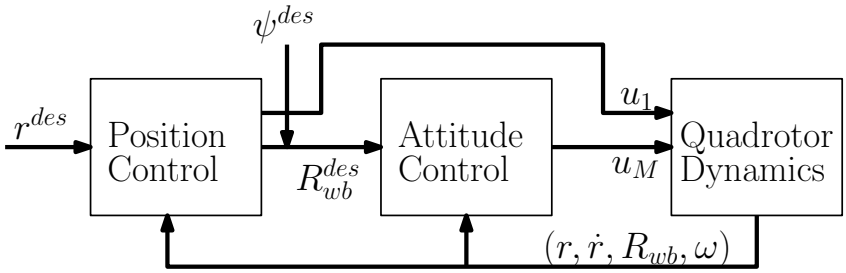

Figure 2.2: The typical cascaded control architecture for quadrotors.

2.7.2 Basic Control Techniques

The quadrotor has complex nonlinear dynamics and many control techniques have been devised. We do not give a comprehensive survey here, but highlight some key developments. Feedback linearization on the full state space to achieve a path tracking objective is studied in [91, 1]. Given that most control objectives are given in terms of the desired position, a more common approach employs a cascaded controller structure, in which the attitude can be stabilized independently of the position [66, 92, 69].

Figure 2.2 shows the cascaded controller structure, which consists of an outer position control module followed by an attitude control module, and assumes that the full state of the vehicle can be measured. Typically, the attitude control module runs at a high frequency onboard the vehicle, while the position control module may run either offboard or onboard and at a lower frequency. The position control module receives a desired reference trajectory, or can generate these internally as a feedback on the states. Intermediate reference trajectories for the attitude control module are generated by the position control module, typically the desired rotation matrix (and sometimes also the desired angular velocity). The attitude control module computes the input u_M, which is mapped along with the thrust u_1 from the position control module to the rotor speeds for the quadrotor.

An important feature is that the quadrotor is differentially flat [73], in which the full state and control can be mapped bijectively to a carefully chosen set of flat outputs and their derivatives [89]. In particular, suitable flat outputs for a quadrotor are the position and yaw, $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4) = (x, y, z, \psi)$. In this way, arbitrary signals (of sufficient smoothness) for the flat outputs can be prescribed and achieved, despite the underactuation of the quadrotor.

This feature is related to the cascaded control architecture described above, namely that the desired position is prescribed for the outer position control module, and then transformed to intermediate references for the attitude control module. Interestingly, in [73] it is shown that the rotation matrix R_{wb} is a function of the linear accelerations and yaw. Specifically, write the columns $R_{wb} = \begin{bmatrix} x_b & y_b & z_b \end{bmatrix}$,

and define $t = (\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3 + g)$ and $x_c = (\cos\sigma_4, \sin\sigma_4, 0)$. Then

$$z_b = \frac{t}{\|t\|}, \quad y_b = \frac{z_b \times x_c}{\|z_b \times x_c\|}, \quad x_b = y_b \times z_b, \tag{2.4}$$

provided that the singularity $z_b \times x_c = 0$ is avoided. Moreover, $u_1 = m\|t\|$.

Here we do not present specific implementation details of the position and attitude control modules, of which many have been proposed [66, 73, 92, 69]. The strength of this approach is that one can focus on the difficult aspects of motion planning through suitable design of the position control module, and rely on the attitude control module for low level implementation. In this dissertation we provide a novel design for the position control module through the use of feedback-based motion primitives on double integrator systems and the observation (2.4) above, and rely on existing attitude control modules. Details are given in Section 4.8.

In summary, one first designs the flat outputs σ as a reference trajectory directly or shapes them by designing a closed-loop vector field according to some higher level objective. Then the flat outputs and their derivatives determine desired angular quantities of the quadrotor, which can finally be used to compute the low-level input rotor speeds.

Chapter 3

Feedback controller synthesis via indefinite linear quadratic optimal control

3.1 Introduction

A key step in the hybrid systems methodology is the synthesis of feedback controllers on a given subset of the safe set. The Reach Control Problem (RCP), as described in Section 2.6, has become the standard method for the case of affine dynamics defined over a polytopic subset. The RCP provides necessary and sufficient conditions for the existence of such feedback controllers as well as a synthesis procedure. In this chapter we focus on an alternative approach for feedback controller synthesis based on optimal control. Our motivation is to soften the strict invariance conditions associated with the RCP and the underlying partition in order to broaden the situations in which a controller can be found. The main idea is to design a cost function that encourages trajectories of the closed-loop system to flow towards and along a desired direction in the given subset.

In our development, we discovered that the form of such cost functions, combined with the consideration of affine dynamics over the given subset in the state space, could be generally expressed in a similar form to the usual linear quadratic optimal control with linear dynamics. We aimed to leverage existing results to provide answers as to when the optimal control problem was solvable and how to synthesize the associated optimal controller.

To this end, in this chapter we consider the regular, infinite-horizon linear quadratic optimal control problem in which the cost functional is the integral of an *indefinite* quadratic form. The *regular* linear quadratic (LQ) problem, when the quadratic form in the cost functional is *positive definite* in the control variables, has been studied extensively in the literature [15, 118, 7]. It has been especially well studied under the standard assumption, the so-called *positive semidefinite* case, when the quadratic form in the cost functional is positive semidefinite in the control and state variables simultaneously. The more general indefinite case imposes no definiteness condition in the control and state variables simultaneously [114, 105]. The LQ problem is termed *infinite-horizon* if the cost functional is integrated over time from zero to infinity. Finally, the most typical treatment of the LQ problem is the *fixed-endpoint* problem where the state is required to converge to zero as time tends to infinity. The case when no such condition is imposed has also been studied and is referred to as the *free-endpoint* problem [105, 99, 38]. In fact, an entire family of LQ problems can be obtained by requiring that the state converges to a subspace. This so-called *stability-modulo-a-subspace* family of LQ problems includes the fixed- and free-endpoint problems as special cases [99, 38]. For the remainder of this chapter, we restrict our attention to the regular and infinite-horizon versions of the problem, for otherwise the optimization problem may yield optimal controllers that are not static linear state feedbacks [115, 7]. Also, we focus on stability-modulo-a-subspace, since it is the more general case.

Traditionally, a complete solution of any variant of the LQ problem requires to find necessary and sufficient conditions for the existence of a finite optimal cost and optimal controls. Existence of a finite optimal cost is called well-posedness, while existence of an optimal control is called attainability. Further, when they exist, a complete solution involves determining the optimal cost and an optimal control. Both should be expressed in terms of the given problem data; that is, the system matrices, the instantaneous cost matrices, and the desired subspace.

In the regular, infinite-horizon, fixed-endpoint, positive semidefinite case, the LQ problem was fully resolved in 1968 by Wonham [117, 118], resulting in the well known necessary and sufficient conditions involving stabilizability and detectability. The corresponding free-endpoint LQ problem was fully characterized much later [36, 106], resulting in conditions involving output stabilizability, a condition less strict than stabilizability [36, 106]. In the regular, infinite-horizon, indefinite case, the fixed-endpoint problem was solved in 1971 by Willems [114], while the free-endpoint problem and general stability-modulo-a-subspace were addressed in 1989 by Trentelman [105, 99]. Importantly, all of the indefinite cases made use of the assumption that the dynamics are controllable. Moreover the solutions are incomplete in that only sufficient conditions for the existence of a finite optimal cost were given (except for the fixed-endpoint problem). The main contribution of this chapter is to extend the above results

for the regular, infinite-horizon, stability-modulo-a-subspace, indefinite case of the LQ problem. Rather than assuming controllability, we only require stabilizability.

It is well known that in both the positive semidefinite and indefinite cases of the regular, infinite-horizon, stability-modulo-a-subspace LQ problem, the optimal cost and optimal controls are given in terms of a particular solution of the algebraic Riccati equation (ARE) [106, 105]. In the treatment of the regular, infinite-horizon, indefinite LQ problem, the controllability assumption is crucial in order to utilize the geometry of the set of all real symmetric solutions of the ARE [114, 64]. In particular, if this solution set is nonempty, there exist a maximal and minimal solution of the ARE [64]. The regular, infinite-horizon, fixed-endpoint LQ problem, both definite and indefinite cases, has always been easier in the sense that the optimal cost and feedback control law are given in terms of the maximal solution, which is the only solution that can stabilize the closed-loop system [114, 117]. For the regular, infinite-horizon, stability-modulo-a-subspace, indefinite case and under the assumption of controllability, the optimal cost and feedback control law are given by a real symmetric solution to the ARE that depends on both its maximal and minimal solutions [99]. In contrast, under the stabilizability assumption, it is unclear which solution of the ARE to select because the geometry of the set of all real symmetric ARE solutions is less well-behaved. In particular, the minimal solution may no longer exist [41, 64]. This ambiguity of the correct choice of ARE solution for the regular, infinite-horizon, stability-modulo-a-subspace, indefinite LQ problem under merely stabilizable dynamics was discussed by Geerts [37, 38], but it has remained elusive.

In this chapter we give the exact form of the optimal feedback that solves the regular, infinite-horizon, stability-modulo-a-subspace, indefinite LQ problem under stabilizable dynamics. Thus we resolve the ambiguity regarding which solution of the ARE to take. Our result requires two assumptions, which are precisely our sufficient conditions for well-posedness: existence of a negative semidefinite solution to the algebraic Riccati inequality (ARI) and stabilizability of the system dynamics. These assumptions may be compared to the sufficient conditions for well-posedness in [105]: existence of a negative semidefinite solution to the ARE and controllability of the system dynamics. The first assumption on existence of a negative semidefinite solution of the ARE or ARI provides for a lower bound on the value function, based on a result of Molinari [78]. Our generalization to the ARI is based on an observation by Geerts [37]. The generalization to the case when the dynamics are stabilizable proves to be the more difficult challenge, as discussed above. This extension constitutes the central contribution of the chapter. Finally, we give necessary and sufficient conditions for optimal controls to exist, which, as pointed out in [105], are nontrivial for regular, infinite-horizon, non-fixed-endpoint, indefinite LQ problems.

As a further validation of the correctness of our results, we recover known results for other variants of

the regular, infinite-horizon LQ problem by adding assumptions to match those problems. In the regular, infinite-horizon, stability-modulo-a-subspace, indefinite case, if we assume controllable dynamics, we obtain the same necessary and sufficient conditions for the existence of optimal controls, the same form of the optimal cost, and the same form of the optimal control as stated in [114, 105, 99]. In the regular, infinite-horizon, positive semidefinite LQ problem, for both the fixed- and free-endpoint cases, if we assume positive semidefiniteness, then we again obtain the same necessary and sufficient conditions for the existence of optimal controls, the same form of the optimal cost, and the same form of the optimal control as stated in [106].

Our resolution of the gap in the LQ literature provides more than just an answer to an academic question. First of all, it provides the foundations for synthesizing feedback controllers along a desired direction, as motivated at the beginning of this chapter. Moreover, these results may also find application to other domains. Recently, the work in [83] considered a linear term in the state of the cost functional and a free-endpoint objective, albeit over the finite-horizon; with a transformation, this cost can be converted to an indefinite problem with stabilizable but not controllable dynamics. The gap was also recently discussed in [30], which deals with the cooperative indefinite LQ problem. As such, our result has application to game theoretic formulations and economics. Although this chapter can potentially have many applications, we have aimed to present the majority of this interesting new development in LQ theory in a transparent and application-free manner.

The outline of this chapter is as follows. In the next section, we motivate the cost functional of an indefinite form with stabilizable dynamics. In Section 3.3 we present the problem statement. In Section 3.4 we summarize the key ingredients needed regarding the geometry of the ARE solutions. In Section 3.5 we state and prove our main results. In Section 3.6 we compare our main result to existing results in the literature. Section 3.7 supplies two examples highlighting the utility of the new theory. Finally, we conclude in Section 3.8, making a connection back to the original motivation from hybrid control.

3.2 Motivation

Consider an affine system

$$\dot{x} = Ax + Bu + a, \qquad x(0) = x_0,$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$. For a control function $u \in L^m_{2,loc}(\mathbb{R}^+)$, let $x(\cdot; x_0, u)$ denote the state trajectory of the system starting at $x_0 \in \mathbb{R}^n$.

Next, consider a unit vector $\xi \in \mathbb{R}^n$ that points in a desired direction in the state space. For example, if there is an underlying simplex $\mathcal{S} \subset \mathbb{R}^n$ with exit facet $\mathcal{F}_0$ characterized by the outward normal h_0, we may impose that $\xi \cdot h_0 > 0$. By using a coordinate transformation to orient the simplex to the canonical simplex with vertices given by unit vectors in the positive orthant [8], we can assume without loss of generality that the base of ξ is at the origin. The direction ξ may also be compared with the notion of a *flow condition* [51].

Now we devise a cost functional in terms of the directional vector ξ. We consider an infinite horizon, discounted cost functional of the form

$$J(x_0, u) = \int_0^\infty \mathrm{e}^{2\alpha t} \omega(x(t; x_0, u), u(t))\, dt,$$

where $\alpha \leq 0$ is a discount factor and $\omega : \mathbb{R}^{n+m} \to \mathbb{R}$ is an instantaneous cost. In the standard linear quadratic formulation, a linear system is considered ($a = 0$), there is no discounting ($\alpha = 0$), and $\omega(x, u) = x^\top Q x + u^\top R u$, with $Q \geq 0$ and $R > 0$. The symmetric cost matrices Q and R enable the designer to tradeoff the aggressiveness of regulation of the state to the origin with the control effort. The discount factor is utilized to help achieve a finite cost, which weighs future performance less. Motivated by the fact that the RCP synthesizes a feedback that causes trajectories to flow out only through the exit facet, our proposed form of the instantaneous cost is

$$\omega(x, u) = d_1(x)^2 - d_2(x) + u^\top R u, \quad R > 0.$$

This cost is a tradeoff between two terms related to the direction ξ and the control effort. To describe $d_1(x)$ and $d_2(x)$, first we decompose the state as $x = x_\parallel + x_\perp$, with $x_\parallel = \lambda \xi$ for $\lambda \in \mathbb{R}$ and $x_\perp \cdot \xi = 0$. Then $d_1(x) := \|x_\perp\|$ is the perpendicular distance of the state to the line in the direction of ξ through the origin and $d_2(x) := x \cdot \xi = \lambda$ is the signed distance of the state along the direction of ξ, which grows positively in the direction ξ. Hence this cost aims to minimize $d_1(x)$,$-d_2(x)$, and the control effort. It is easy to show that $d_1(x)^2 = x^\top Q(\xi) x$, with $Q(\xi) = (I_n - \xi\xi^\top)$. An $n = 2$ example of the level sets for $d_1(x)^2 - d_2(x)$ is shown in Figure 1.2, with ξ pointing in the positive orthant.

We would like to obtain an analytic solution to this problem and to leverage, if possible, existing results. Several complications arise, namely that the term $d_2(x)$ introduces a linear term in the state and may be negative, and the dynamics are affine. We can transform the problem to an equivalent one in which the dynamics are linear and the cost has only quadratic terms [7]. Let $\hat{x} := \mathrm{e}^{\alpha t} \begin{bmatrix} x^\top & z \end{bmatrix}^\top$ with

$\dot{z} = 0$ and $\hat{x}_0 = \hat{x}(0) = \begin{bmatrix} x_0^\top & 1 \end{bmatrix}^\top$, and $\hat{u} := \mathrm{e}^{\alpha t} u$. We obtain linear dynamics

$$\dot{\hat{x}} = \begin{bmatrix} A + \alpha I_n & a \\ 0 & \alpha \end{bmatrix} \hat{x} + \begin{bmatrix} B \\ 0 \end{bmatrix} \hat{u},$$

with cost functional

$$J(\hat{x}_0, \hat{u}) = \int_0^\infty \left(\hat{x}(t)^\top \begin{bmatrix} Q(\xi) & -\xi/2 \\ -\xi^\top/2 & 0 \end{bmatrix} \hat{x}(t) + \hat{u}(t)^\top R \hat{u}(t) \right) dt.$$

We can see that the augmented linear system is only stabilizable if $\alpha < 0$ and (A, B) is controllable, highlighting the role of the discount factor. Moreover, the instantaneous cost is not bounded below, and consequently the quadratic term on the augmented state is indefinite (alternatively Theorem 2.4.1 may be used). From this, we observed that the existing LQ literature could not resolve this case of stabilizable dynamics with an indefinite cost, leading to our general study of this problem.

3.3 Problem Statement

We consider the linear control system

$$\dot{x} = Ax + Bu, \qquad x(0) = x_0, \tag{3.1}$$

where $x \in \mathbb{R}^n$ and $u \in \mathbb{R}^m$. For a control function $u \in L^m_{2,loc}(\mathbb{R}^+)$, let $x(\cdot; x_0, u)$ denote the state trajectory of (3.1) starting at $x_0 \in \mathbb{R}^n$. Then for $T \geq 0$, the *cost function* is

$$J_T(x_0, u) = \int_0^T \omega(x(t; x_0, u), u(t))\, dt \tag{3.2}$$

with a quadratic *instantaneous cost*

$$\omega(x, u) := x^\top Q x + u^\top R u = \begin{bmatrix} x^\top & u^\top \end{bmatrix} W \begin{bmatrix} x \\ u \end{bmatrix}, \qquad W := \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix}, \qquad R = I_m. \tag{3.3}$$

We allow Q to be indefinite, whereas $R := I_m > 0$. More general quadratic cost functions can be considered, but they can be converted via a feedback transformation to the form we use here, as in Chapter 10 of [106]. This feedback transformation does not affect solvability of the problem; hence,

there is no loss of generality in our choice of W.

Because W may be indefinite, we define the set of control inputs that yield a cost that is either finite, ∞, or $-\infty$:

$$\mathcal{U}(x_0) := \left\{ u \in L^m_{2,loc}(\mathbb{R}^+) \mid \lim_{T\to\infty} J_T(x_0, u) \text{ exists in } \mathbb{R}^e \right\}. \tag{3.4}$$

Let $\mathcal{L} \subset \mathbb{R}^n$ be a subspace. Recalling that $d_{\mathcal{L}}$ gives the minimum distance from a point to the set $\mathcal{L}$, the set of permissible control inputs such that the state asymptotically converges to $\mathcal{L}$ is

$$\mathcal{U}_{\mathcal{L}}(x_0) := \left\{ u \in \mathcal{U}(x_0) \mid \lim_{t\to\infty} d_{\mathcal{L}}(x(t; x_0, u)) = 0 \right\}. \tag{3.5}$$

For $u \in \mathcal{U}_{\mathcal{L}}(x_0)$, we define

$$J(x_0, u) := \lim_{T\to\infty} J_T(x_0, u). \tag{3.6}$$

We define the *optimal cost* or *value function* to be

$$V_{\mathcal{L}}(x_0) := \inf\{J(x_0, u) \mid u \in \mathcal{U}_{\mathcal{L}}(x_0)\}. \tag{3.7}$$

Now we define the linear quadratic optimal control problem with stability-modulo-$\mathcal{L}$ $(\text{LQCP})_{\mathcal{L}}$.

Problem 3.3.1 ($(\text{LQCP})_{\mathcal{L}}$)**.** *Consider the system* (3.1) *with the quadratic cost criterion* (3.2)*. Let* $\mathcal{L} \subset \mathbb{R}^n$ *be a given subspace. For all* $x_0 \in \mathbb{R}^n$*, find the optimal cost* $V_{\mathcal{L}}(x_0)$ *and an optimal control* $u^\star \in \mathcal{U}_{\mathcal{L}}(x_0)$ *such that* $V_{\mathcal{L}}(x_0) = J(x_0, u^\star)$*.*

The $(\text{LQCP})_{\mathcal{L}}$ is called *regular* (as opposed to singular) if $R > 0$. It is called *positive semidefinite* if ω is positive semidefinite on $\mathbb{R}^{n+m}$, and *indefinite* otherwise. If $\mathcal{L} = \mathbb{R}^n$, the $(\text{LQCP})_{\mathcal{L}}$ is called a *free-endpoint problem*, and if $\mathcal{L} = 0$, it is called a *fixed-endpoint problem.* We are particularly interested in characterizing two properties of the $(\text{LQCP})_{\mathcal{L}}$.

Definition 3.3.2. *We say the* $(LQCP)_{\mathcal{L}}$ *is* well-posed *if for all* $x_0 \in \mathbb{R}^n$*,* $V_{\mathcal{L}}(x_0) \in \mathbb{R}$*. We say the* $(LQCP)_{\mathcal{L}}$ *is* attainable *if for all* $x_0 \in \mathbb{R}^n$*, there exists a control* $u^\star \in \mathcal{U}_{\mathcal{L}}(x_0)$ *such that* $V_{\mathcal{L}}(x_0) = J(x_0, u^\star)$*. Such an input is called* optimal*. We say the* $(LQCP)_{\mathcal{L}}$ *is* solvable *if it is both well-posed and attainable.*

3.4 Preliminaries

The main results on the $(\text{LQCP})_{\mathcal{L}}$ are centered on the algebraic Riccati equation (ARE):

$$\phi(K) := A^\top K + KA + Q - KBB^\top K = 0. \tag{3.8}$$

The algebraic Riccati inequality (ARI) is given by $\phi(K) \geq 0$. For convenience, we define

$$A(K) := A - BB^T K. \tag{3.9}$$

Also we define the following solution sets:

$$\begin{aligned}
\Gamma &:= \{K \in \mathbb{R}^{n\times n} \mid K = K^\top, \phi(K) \geq 0\},\\
\partial\Gamma &:= \{K \in \mathbb{R}^{n\times n} \mid K = K^\top,\ \phi(K) = 0\},\\
\Gamma_- &:= \{K \in \Gamma \mid K \leq 0\}.
\end{aligned}$$

The geometry of the solutions to the ARE can be studied in both the controllable and stabilizable cases; see, in particular, Chapters 7 and 8 of [64] and also [105]. First we consider the case when (A, B) is controllable. The next result summarizes what is known about the extremal solutions in Γ and in $\partial\Gamma$.

Theorem 3.4.1. *Suppose (A, B) is controllable.*

(i) If $\Gamma \neq \emptyset$, then the maximal and minimal solutions in Γ exist, $\partial\Gamma \neq \emptyset$, its maximal and minimal solutions exist, and they are identical to the maximal and minimal solutions in Γ.

(ii) If $\partial\Gamma \neq \emptyset$, then its maximal and minimal solutions $K^+, K^- \in \partial\Gamma$ satisfy: $\forall K \in \partial\Gamma$, $K^- \leq K \leq K^+$. Moreover, they are the unique solutions in $\partial\Gamma$ such that $\sigma(A(K^+)) \subset \mathbb{C}^- \cup \mathbb{C}^0$ and $\sigma(A(K^-)) \subset \mathbb{C}^+ \cup \mathbb{C}^0$.

Proof. The first statement is Theorem 14(b) in [94]. The second statement was proved in [114]. See also Theorem 7.5.1, p. 168, in [64]. □

If $\partial\Gamma \neq \emptyset$, define the *gap* of the ARE to be $\Delta := K^+ - K^-$. Let Ω denote the set of all $A(K^-)-$invariant subspaces contained in $\mathcal{X}^+(A(K^-))$. The following theorem was first proven by Willems [114]; see also [64].

Theorem 3.4.2 (Theorem 3.1, [105])**.** *Let (A, B) be controllable and suppose $\partial\Gamma \neq \emptyset$. If $\mathcal{V} \subset \Omega$, then $\mathbb{R}^n = \mathcal{V} \oplus \Delta^{-1}(\mathcal{V}^\perp)$. There exists a bijection $\gamma : \Omega \to \partial\Gamma$ defined by*

$$\gamma(\mathcal{V}) := K^- P_{\mathcal{V}} + K^+(I_n - P_{\mathcal{V}}), \tag{3.10}$$

where $P_{\mathcal{V}}$ is the projection onto $\mathcal{V}$ along $\Delta^{-1}(\mathcal{V}^\perp)$. If $K = \gamma(\mathcal{V})$, then $\mathcal{X}^+(A(K)) = \mathcal{V}$, $\mathcal{X}^0(A(K)) = \mathrm{Ker}(\Delta)$, and $\mathcal{X}^-(A(K)) = \mathcal{X}^-(A(K^+)) \cap \Delta^{-1}(\mathcal{V}^\perp)$.

An application of Theorem 3.4.2 is the main result of [99], which provides a solution of the $(\text{LQCP})_{\mathcal{L}}$ when (A, B) is controllable. To state the sufficient condition for well-posedness, an additional definition is needed from [99]: for a given subspace $\mathcal{L} \subset \mathbb{R}^n$ and symmetric matrix $K \in \mathbb{R}^{n\times n}$, K is said to be *negative semidefinite on* $\mathcal{L}$ if for all $x \in \mathcal{L}$, $x^\top Kx \leq 0$, and $x^\top Kx = 0$ if and only if $Kx = 0$. Notice that $K \leq 0$ implies that for all $\mathcal{L} \subset \mathbb{R}^n$, K is negative semidefinite on $\mathcal{L}$. To see this, fix $\mathcal{L} \subset \mathbb{R}^n$ and note that $K \leq 0$ implies that there exists $H \in \mathbb{R}^{p\times n}$ for some p such that $K = -H^\top H$. Then for all $x \in \mathcal{L} \subset \mathbb{R}^n$, obviously $x^\top Kx \leq 0$, $Kx = 0$ implies $x^\top Kx = 0$, and

$$x^\top Kx = -(Hx)^\top (Hx) = 0 \quad \Leftrightarrow \quad Hx = 0 \quad \Rightarrow \quad -H^\top (Hx) = Kx = 0. \tag{3.11}$$

Theorem 3.4.3 (Theorem 4.1, [99]). *Let (A, B) be controllable. Assume $\partial\Gamma \neq \emptyset$ and K^- is negative semidefinite on $\mathcal{L}$. Then we have*

(i) For all $x_0 \in \mathbb{R}^n$, $V_{\mathcal{L}}(x_0)$ is finite.

(ii) For all $x_0 \in \mathbb{R}^n$, $V_{\mathcal{L}}(x_0) = x_0^\top K^\star x_0$, where $K^\star := \gamma(\mathcal{N}(\mathcal{L}))$ and $\mathcal{N}(\mathcal{L}) := \langle \mathcal{L} \cap \operatorname{Ker}(K^-) \mid A(K^-)\rangle \cap \mathcal{X}^+(A(K^-))$.

(iii) For all $x_0 \in \mathbb{R}^n$, there exists an optimal input $u^\star$ if and only if $\operatorname{Ker}(\Delta) \subset \mathcal{L} \cap \operatorname{Ker}(K^-)$.

(iv) If $\operatorname{Ker}(\Delta) \subset \mathcal{L} \cap \operatorname{Ker}(K^-)$, then for each $x_0 \in \mathbb{R}^n$, there exists exactly one optimal input $u^\star$, and it is given by the feedback $u^\star = -B^\top K^\star x$.

The results of this chapter can be regarded as a generalization of the previous result to the stabilizable case. That is, we require weaker assumptions for the sufficient condition of well-posedness to be able to provide the form of the value function, necessary and sufficient conditions for attainability, and the form of the optimal control. Our new assumptions involve the stabilizability of (A, B) rather than controllability, and the existence of a negative semidefinite solution to the ARI rather than imposing that specifically K^-, a solution to the ARE, is negative semidefinite on $\mathcal{L}$. Because necessary and sufficient conditions for well-posedness are still an open problem, note that we have not attempted to generalize our second condition in terms of the existence of an ARI solution that is negative semidefinite on $\mathcal{L}$. Regardless, the main technical obstacle is that there is no direct generalization of Theorem 3.4.2 to the stabilizable case; indeed the minimal solution K^- may not exist in this case.

Now supposing that (A, B) is stabilizable, we can write the system (3.1) in the Kalman controllability decomposition. Let $\mathcal{C} = \langle A \mid \operatorname{Im}(B)\rangle \subset \mathbb{R}^n$ be the controllable subspace with dimension $n_1 \leq n$. Also, let $\mathcal{X}_2$ be any complement such that

$$\mathbb{R}^n = \mathcal{C} \oplus \mathcal{X}_2. \tag{3.12}$$

Then the system matrices have the block form:

$$A = \begin{bmatrix} A_1 & A_{12} \\ 0 & A_2 \end{bmatrix}, \qquad B = \begin{bmatrix} B_1 \\ 0 \end{bmatrix}. \tag{3.13}$$

It can be shown that coordinate transformations only affect the solutions $K \in \partial\Gamma$ of the $(\mathrm{LQCP})_{\mathcal{L}}$ (in any endpoint case) up to a congruent transformation, so there is no loss of generality to assume that (A, B) already has the form (3.13). If we write the symmetric matrices Q and K in block form

$$Q = \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^\top & Q_2 \end{bmatrix}, \quad K = \begin{bmatrix} K_1 & K_{12} \\ K_{12}^\top & K_2 \end{bmatrix}, \tag{3.14}$$

then $\phi(K)$ also can be decomposed in block form:

$$\phi(K) = \begin{bmatrix} \phi_1(K_1) & A_1(K_1)^\top K_{12} + K_{12}A_2 + K_1A_{12} + Q_{12} \\ * & A_2^\top K_2 + K_2A_2 + K_{12}^\top A_{12} + A_{12}^\top K_{12} + Q_2 - K_{12}^\top B_1 B_1^\top K_{12} \end{bmatrix}. \tag{3.15}$$

We note that $\phi(K)$ is symmetric, and $\phi_1(K_1)$ is defined below in (3.17). Let

$$A_1(K_1) := A_1 - B_1B_1^\top K_1. \tag{3.16}$$

Then $\phi(K) = 0$ gives rise to three equations

$$\phi_1(K_1) := A_1^T K_1 + K_1A_1 + Q_1 - K_1B_1B_1^\top K_1 = 0, \tag{3.17}$$

$$A_1(K_1)^\top K_{12} + K_{12}A_2 = -(Q_{12} + K_1A_{12}), \tag{3.18}$$

$$A_2^\top K_2 + K_2A_2 = K_{12}^\top B_1B_1^\top K_{12} - K_{12}^\top A_{12} - A_{12}^\top K_{12} - Q_2\,. \tag{3.19}$$

The first equation (3.17) is a quadratic equation with (A_1, B_1) controllable. Its solutions K_1 are decoupled from K_{12} and K_2, so this lower order $(n_1 \times n_1)$ ARE equation can be solved first. The relevant solution sets are denoted as:

$$\begin{aligned} \Gamma_1 &:= \{K_1 \in \mathbb{R}^{n_1\times n_1} \mid K_1^\top = K_1,\ \phi_1(K_1) \geq 0\}, \\ \partial\Gamma_1 &:= \{K_1 \in \mathbb{R}^{n_1\times n_1} \mid K_1^\top = K_1,\ \phi_1(K_1) = 0\}, \\ \Gamma_{1-} &:= \{K_1 \in \Gamma_1 \mid K_1 \leq 0\}, \\ \partial\Gamma_{1-} &:= \{K_1 \in \partial\Gamma_1 \mid K_1 \leq 0\}. \end{aligned}$$

Using any solution $K_1 \in \partial\Gamma_1$, if it exists, (3.18) is a linear (Sylvester) equation for K_{12} which may have no solutions, infinitely many solutions, or a unique solution. The third equation (3.19) is also a linear (Sylvester) equation. Using any solution K_{12}, if it exists, gives a unique solution to K_2. To see this, recall that if $M_1 \in \mathbb{R}^{n_1\times n_1}$, $M_2 \in \mathbb{R}^{n_2\times n_2}$, and $M_3 \in \mathbb{R}^{n_1\times n_2}$ are given matrices, then the Sylvester equation $M_1X + XM_2 = M_3$ has a unique solution $X \in \mathbb{R}^{n_1\times n_2}$ exactly when $\sigma(M_1)\cap\sigma(-M_2) = \emptyset$ [34]. Because stabilizability of (A, B) implies $\sigma(A_2) \subset \mathbb{C}^-$, then by applying the Sylvester solvability criteria to (3.19), we have that $\sigma(A_2^\top)\cap\sigma(-A_2) = \emptyset$, and so K_2 is unique for any given K_{12}.

In preparation for characterizing the existence and form of the value function analogously to Theorem 3.4.3 (i) and (ii), we consider existence of extremal solutions in $\partial\Gamma$. It is known that when (A, B) is stabilizable, then the maximal solution $K^+ \in \partial\Gamma$ exists, whereas the minimal solution K^- may not exist.

Theorem 3.4.4 (Theorem 2.1, [41]; Theorem 7.9.3, p. 195, [64])**.** *Suppose (A, B) is stabilizable and $\partial\Gamma \neq \emptyset$. Then the unique maximal solution $K^+ \in \partial\Gamma$ exists. Moreover, $\sigma(A(K^+)) \subset \mathbb{C}^- \cup \mathbb{C}^0$.*

To obtain a generalization of Theorem 3.4.3 to the stabilizable case, one of the major steps in the sequel is to apply Theorem 3.4.3 to the controllable subsystem (A_1, B_1) and its ARE (3.17). Theorem 3.4.3 requires that the minimal solution K_1^- of (3.17) exists and is negative semidefinite on $\mathcal{L}$ within the controllable subspace. The following lemma provides for the existence of this minimal, negative semidefinite solution.

Lemma 3.4.5. *Suppose (A, B) is stabilizable, $\Gamma_- \neq \emptyset$, and the state space is decomposed as in* (3.12)*. Then the minimal solution $K_1^- \in \partial\Gamma_{1-}$ exists.*

Proof. Let $K \in \Gamma_-$ so that $\phi(K) \geq 0$ and $K \leq 0$. Consider K, Q, and $\phi(K)$ in block form (3.14)-(3.15). Applying Theorem 2.4.1 to both K and $\phi(K)$, we obtain $\phi_1(K_1) \geq 0$ and $K_1 \leq 0$, which implies $K_1 \in \Gamma_{1-} \neq \emptyset$. Since also (A_1, B_1) is controllable, we can apply Theorem 3.4.1(i) to conclude $K_1^+, K_1^- \in \Gamma_1$, the maximal and minimal solutions, exist. Moreover $\partial\Gamma_1 \neq \emptyset$ and its maximal and minimal elements are precisely K_1^+ and K_1^-. Because $K_1 \leq 0$, $K_1^- \in \Gamma_1$ is minimal, and $K_1, K_1^- \in \Gamma_1$, we have that $K_1^- \leq K_1 \leq 0$. That is, $K_1^- \in \partial\Gamma_{1-}$, as desired. □

3.5 Solution of the (LQCP)$_{\mathcal{L}}$

In this section we present the solution of the (LQCP)$_{\mathcal{L}}$. That is, we give sufficient conditions for well-posedness, the form of the value function, necessary and sufficient conditions for attainability, and form

of the optimal control. We assume that $\mathcal{L} \subset \mathbb{R}^n$ is a given subspace. Well-posedness and the form of the value function are addressed through the following sufficient condition, which are also found in [37, 38].

Assumption 3.5.1. *We assume that (A, B) is stabilizable and $\Gamma_- \neq \emptyset$.*

The following theorem states that the value function is given in terms of a quadratic form of a particular solution to the ARE.

Theorem 3.5.2 (Theorem 2.1 [37], Lemma 5 [78])**.** *Consider the (LQCP)$_\mathcal{L}$ and suppose Assumption 3.5.1 holds. Then there exists a unique $K^\star \in \partial\Gamma$ such that for all $x_0 \in \mathbb{R}^n$, $V_\mathcal{L}(x_0) = x_0^\top K^\star x_0$.*

Next we turn to the form of $K^\star$. Our approach is to choose a suitable basis based on the Kalman controllability decomposition (3.12) and on Theorem 3.4.2, following the same method in [105]. Then we systematically determine each of the blocks of $K^\star$. First we determine $K_1^\star$ using results from [99]; second, we compute $K_{12}^\star$ assuming $K_1^\star$ is known; finally, we compute $K_2^\star$ assuming $K_{12}^\star$ is known. Now we give a more detailed roadmap on how the technical results are obtained.

The choice of $K_1^\star$ is resolved by applying Theorem 3.4.3 to the controllable subsystem. We construct a smaller optimal control problem on the controllable subsystem. Intuitively, the smaller optimal control problem should be equivalent to the original (LQCP)$_\mathcal{L}$ for initial conditions in the controllable subspace. After proving this equivalence, we apply Theorem 3.4.3 to obtain $K_1^\star = \overline{K}_1$, where $\overline{K}_1$ is defined in (3.22) below. Next, we fix the choice of $K_1^\star$ that solves (3.17) and turn to the solution set of (3.18). Generally, this linear Sylvester equation may have an infinite number of solutions, making the choice of $K_{12}^\star$ nontrivial to determine. However, once $K_{12}^\star$ is determined, then $K_2^\star$ is uniquely determined from the linear Sylvester equation (3.19), since (A, B) is stabilizable. Thus $K_{12}^\star$ is the main obstacle. Interestingly, under a restrictive regularity assumption introduced in [41], the solution set of (3.18) collapses to a single element. On the other hand, Theorem 3.5.2 states that $K_{12}^\star$ exists without the regularity assumption. We forego the assumption and search for a more general principle that can resolve the choice of $K_{12}^\star$.

Our approach involves exploiting the structure within the Kalman controllability decomposition, similarly as in [105]. Based on a modal decomposition of $A_1(\overline{K}_1)$, the Sylvester equation (3.18) with $K_1 = K_1^\star$ splits into three decoupled linear Sylvester equations (3.34)-(3.36). The problematic part of $K_{12}^\star$, denoted $K_{12,1}^*$ is then isolated to (3.34) only. Regarding the solution of (3.34), it is well known (see Theorem 10.13 of [106]) that for stabilizable systems with positive semidefinite cost in the free endpoint case, the solution of the ARE is given by the smallest positive semidefinite solution in $\partial\Gamma$. Also, $0 \in \Gamma$ if and only if $Q \geq 0$ (see for example equation (1.16) of [38]) and so $0 \in \Gamma_-$ and $x_0^\top 0 x_0 = 0$ gives a lower bound on the value function. Using the previous two observations, we find through repeated trials that

$K_{12}^\star = 0$ in the positive semidefinite case. At this point we make a guess that the same form of $K_{12}^\star$ would arise in the indefinite case. Finally, we unambiguously deduce that $K_{12}^\star = 0$.

Once we have fully characterized the form of $K^\star$, obtaining necessary and sufficient conditions for attainability follows analogously to the proof presented in [105, 99]. We require only a few augmentations to account for the uncontrollable (but stable) dynamics. Now we proceed to the actual development.

The first step is to fix a suitable basis so that the blocks of $K^\star$ can be computed. Consider the Kalman controllability decomposition (3.12), and suppose Assumption 3.5.1 holds. Then by Lemma 3.4.5, the unique minimal solution $K_1^- \in \partial\Gamma_1 \neq \emptyset$ exists and $K_1^- \leq 0$. Similarly, because (A_1, B_1) is controllable and $\partial\Gamma_1 \neq \emptyset$, we can apply Theorem 3.4.1 to obtain the unique maximal solution $K_1^+ \in \partial\Gamma_1$. Let $\Delta_1 := K_1^+ - K_1^-$ be the gap associated with (3.17), the ARE in the controllable subspace. Following [105, 99], we can further decompose the controllable subspace based on Theorem 3.4.2. To that end, define the following subspaces of $\mathbb{R}^{n_1}$:

$$\mathcal{L}_1 := \mathcal{L} \cap \mathcal{C} \tag{3.20}$$

$$\mathcal{N}_1(\mathcal{L}_1) := \langle \mathcal{L}_1 \cap \mathrm{Ker}(K_1^-) \mid A_1(K_1^-) \rangle \cap \mathcal{X}^+(A_1(K_1^-)) \,. \tag{3.21}$$

Here and for the remainder of this section, for simplicity we do not notationally differentiate a subspace that can belong to various vector spaces of different dimensions. For example, although technically $\mathcal{L} \cap \mathcal{C} \subset \mathbb{R}^n$, we can view $\mathcal{L}_1$ as a subspace of $\mathbb{R}^{n_1} \sim \mathcal{C}$.

Let $P_{\mathcal{N}_1(\mathcal{L}_1)} : \mathbb{R}^{n_1} \to \mathcal{N}_1(\mathcal{L}_1)$ be the projection onto $\mathcal{N}_1(\mathcal{L}_1)$ along $\Delta_1^{-1}(\mathcal{N}_1(\mathcal{L}_1)^\perp)$. Because $\mathcal{N}_1(\mathcal{L}_1)$ is an $A_1(K_1^-)$-invariant subspace contained in $\mathcal{X}^+(A_1(K_1^-))$ for any $\mathcal{L}_1$, we can apply Theorem 3.4.2 to obtain a solution $\overline{K}_1 \in \partial\Gamma_1$ of the ARE of the form

$$\overline{K}_1 := \gamma(\mathcal{N}_1(\mathcal{L}_1)) = K_1^- P_{\mathcal{N}_1(\mathcal{L}_1)} + K_1^+ (I_{n_1} - P_{\mathcal{N}_1(\mathcal{L}_1)}). \tag{3.22}$$

Following Theorem 3.4.2, define the following subspaces in $\mathcal{C} \sim \mathbb{R}^{n_1}$:

$$\mathcal{X}_{1,1} := \mathcal{X}^+(A_1(\overline{K}_1)) = \mathcal{N}_1(\mathcal{L}_1), \tag{3.23}$$

$$\mathcal{X}_{1,2} := \mathcal{X}^0(A_1(\overline{K}_1)) = \mathrm{Ker}(\Delta_1), \tag{3.24}$$

$$\mathcal{X}_{1,3} := \mathcal{X}^-(A_1(\overline{K}_1)) = \mathcal{X}^-(A_1(K_1^+)) \cap \Delta_1^{-1}(\mathcal{N}_1(\mathcal{L}_1)^\perp). \tag{3.25}$$

Then the state space decomposition (3.12) splits further into

$$\mathbb{R}^n = \mathcal{X}_{1,1} \oplus \mathcal{X}_{1,2} \oplus \mathcal{X}_{1,3} \oplus \mathcal{X}_2 \,. \tag{3.26}$$

Let $n_{1,i} := \dim(\mathcal{X}_{1,i})$ for $i = 1, 2, 3$ so that $n_1 = n_{1,1} + n_{1,2} + n_{1,3} \leq n$. Without loss of generality (after a change of coordinates), the system matrices have the block form

$$A = \begin{bmatrix} A_1 & A_{12} \\ 0 & A_2 \end{bmatrix} = \begin{bmatrix} A_{1,11} & A_{1,12} & A_{1,13} & A_{12,1} \\ A_{1,21} & A_{1,22} & A_{1,23} & A_{12,2} \\ A_{1,31} & A_{1,32} & A_{1,33} & A_{12,3} \\ 0 & 0 & 0 & A_2 \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ 0 \end{bmatrix} = \begin{bmatrix} B_{1,1} \\ B_{1,2} \\ B_{1,3} \\ 0 \end{bmatrix}. \tag{3.27}$$

The cost matrix Q and each $K \in \Gamma$ have the block form

$$Q = \begin{bmatrix} Q_1 & Q_{12} \\ Q_{12}^\top & Q_2 \end{bmatrix} = \begin{bmatrix} Q_{1,11} & Q_{1,12} & Q_{1,13} & Q_{12,1} \\ Q_{1,12}^\top & Q_{1,22} & Q_{1,23} & Q_{12,2} \\ Q_{1,13}^\top & Q_{1,23}^\top & Q_{1,33} & Q_{12,3} \\ Q_{12,1}^\top & Q_{12,2}^\top & Q_{12,3}^\top & Q_2 \end{bmatrix}, \quad K = \begin{bmatrix} K_1 & K_{12} \\ K_{12}^\top & K_2 \end{bmatrix} = \begin{bmatrix} K_{1,11} & K_{1,12} & K_{1,13} & K_{12,1} \\ K_{1,12}^\top & K_{1,22} & K_{1,23} & K_{12,2} \\ K_{1,13}^\top & K_{1,23}^\top & K_{1,33} & K_{12,3} \\ K_{12,1}^\top & K_{12,2}^\top & K_{12,3}^\top & K_2 \end{bmatrix}. \tag{3.28}$$

Our goal is to compute all of the blocks in (3.28) for $K = K^\star$. First we resolve the choice of $K_1^\star$.

Theorem 3.5.3. *Consider the (LQCP)$_{\mathcal{L}}$ and suppose Assumption 3.5.1 holds. Then in the state space decomposition* (3.12)*, $K_1^\star = \overline{K}_1$, as given in* (3.22)*.*

Proof. Since (A, B) is stabilizable, without loss of generality, (A, B) has the form (3.13), and Q and K have the block form (3.14). Defining $x := (x_1, x_2)$, the Kalman controllability decomposition is

$$\dot{x}_1 = A_1 x_1 + A_{12} x_2 + B_1 u, \; x_1(0) = x_{1,0} \tag{3.29}$$

$$\dot{x}_2 = A_2 x_2, \; x_2(0) = x_{2,0}. \tag{3.30}$$

The controllable subspace is $\mathcal{C} = \{x \in \mathbb{R}^n \mid x_2 = 0\}$. If $x_{2,0} = 0$, then for all $t \geq 0$, $x_2(t) = 0$ and $x(t) \in \mathcal{C}$. Thus, we can define a new $(\text{LQCP})_{\mathcal{L}_1}$ on $\mathcal{C}$ with dynamics $\dot{x}_1 = A_1 x_1 + B_1 u$, $x_1(0) = x_{1,0}$, and (A_1, B_1) is controllable. The cost function is $J_{1T}(x_{1,0}, u) := \int_0^T \omega_1(x_1(t; x_{1,0}, u), u(t))\, dt$ with $\omega_1(x_1, u) := x_1^\top Q_1 x_1 + u^\top u$. Let $\mathcal{L}_1 = \mathcal{L} \cap \mathcal{C}$ be the terminal subspace and let $d_{1\mathcal{L}_1} : \mathbb{R}^{n_1} \to [0, \infty)$ be

the distance function. The input spaces are

$$\mathcal{U}_1(x_{1,0}) := \left\{ u \in L^m_{2,loc}(\mathbb{R}^+) \mid \lim_{T\to\infty} J_{1T}(x_{1,0}, u) \text{ exists in } \mathbb{R}^e \right\}, \tag{3.31}$$

$$\mathcal{U}_{1\mathcal{L}_1}(x_{1,0}) := \left\{ u \in \mathcal{U}_1(x_{1,0}) \mid \lim_{t\to\infty} d_{1\mathcal{L}_1}(x_1(t; x_{1,0}, u)) = 0 \right\}. \tag{3.32}$$

The optimal cost is $V_{1\mathcal{L}_1}(x_{1,0}) := \inf\{\lim_{T\to\infty} J_{1T}(x_{1,0}, u) \mid u \in \mathcal{U}_{1\mathcal{L}_1}(x_{1,0})\}$. The ARE for the $(\text{LQCP})_{\mathcal{L}_1}$ is $\phi_1(K_1) = 0$ as in (3.17) with solution set $\partial\Gamma_1$. Consider any initial condition $x_0 = (x_{1,0}, 0) \in \mathcal{C}$ and any control $u \in L^m_{2,loc}(\mathbb{R}^+)$. Then $x(t; x_0, u) = (x_1(t; x_{1,0}, u), 0)$ and $\omega(x(t; x_0, u), u(t)) = \omega_1(x_1(t; x_{1,0}, u), u(t))$, so for all $T \geq 0$, $J_T(x_0, u) = J_{1T}(x_{1,0}, u)$. Consequently, we have $\mathcal{U}(x_0) = \mathcal{U}_1(x_{1,0})$. Also, $\lim_{t\to\infty} d_{\mathcal{L}}(x(t; x_0, u)) = 0$ is equivalent to $\lim_{t\to\infty} d_{1\mathcal{L}_1}(x_1(t; x_{1,0}, u)) = 0$. Thus $\mathcal{U}_{\mathcal{L}}(x_0) = \mathcal{U}_{1\mathcal{L}_1}(x_{1,0})$. With all the above, we conclude that $V_{\mathcal{L}}(x_0) = V_{1\mathcal{L}_1}(x_{1,0})$ for $x_0 = (x_{1,0}, 0) \in \mathcal{C}$.

Since (A_1, B_1) is controllable, we can apply the results of [99] to solve the $(\text{LQCP})_{\mathcal{L}_1}$. Since $\Gamma_- \neq \emptyset$, we can apply Lemma 3.4.5 to get that the minimal solution $K_1^- \in \partial\Gamma_{1-}$ exists. Since $K_1^- \leq 0$, from (3.11) it follows that K_1^- is negative semidefinite on $\mathcal{L}_1$. By Theorem 3.4.3(ii), $V_{1\mathcal{L}_1}(x_{1,0}) = x_{1,0}^\top \overline{K}_1 x_{1,0}$ with $\overline{K}_1$ given in (3.22). Since we have already shown that $V_{\mathcal{L}}(x_0) = V_{1\mathcal{L}_1}(x_{1,0})$ for $x_0 = (x_{1,0}, 0) \in \mathcal{C}$, it can be easily shown that $K_1^\star = \overline{K}_1$. □

To resolve the remaining blocks of $K^\star$, we recall some results from [105]. For this to apply, we continue to assume that the state space is decomposed according to (3.26). It was shown in (5.5) and (5.7) of [105] that $\overline{K}_1$ in (3.22) and the closed-loop system matrix $A_1(\overline{K}_1)$ using $\overline{K}_1$ have the form

$$\overline{K}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & K_{1,22}^- & K_{1,23}^- \\ 0 & K_{1,23}^{-\top} & K_{1,33}^+ \end{bmatrix}, \quad A_1(\overline{K}_1) = \begin{bmatrix} \overline{A}_{1,11} & 0 & 0 \\ 0 & \overline{A}_{1,22} & 0 \\ 0 & 0 & \overline{A}_{1,33} \end{bmatrix}, \tag{3.33}$$

where $\sigma(\overline{A}_{1,11}) \subset \mathbb{C}^+$, $\sigma(\overline{A}_{1,22}) \subset \mathbb{C}^0$, and $\sigma(\overline{A}_{1,33}) \subset \mathbb{C}^-$. For the choice of $K_1 = \overline{K}_1$ and substituting (3.27), (3.28), and (3.33), the second ARE equation (3.18) splits into three linear Sylvester equations:

$$\overline{A}_{1,11}^\top K_{12,1} + K_{12,1} A_2 = -Q_{12,1}, \tag{3.34}$$

$$\overline{A}_{1,22}^\top K_{12,2} + K_{12,2} A_2 = -(Q_{12,2} + K_{1,22}^- A_{12,2} + K_{1,23}^- A_{12,3}), \tag{3.35}$$

$$\overline{A}_{1,33}^\top K_{12,3} + K_{12,3} A_2 = -(Q_{12,3} + K_{1,23}^{-\top} A_{12,2} + K_{1,33}^+ A_{12,3}). \tag{3.36}$$

Using these facts, we can now resolve the remaining blocks of $K^\star$. The main difficulty is that (3.34) may have an infinite number of solutions for the $K_{12,1}$ block since $\sigma(\overline{A}_{1,11}^\top) \cap \sigma(-A_2)$ is not necessarily

empty. The key insight is that $K_{12,1}^{\star}$ can be unambiguously determined by invoking Theorem 3.5.5(ii) given below, that any negative semidefinite solution $K_N \in \Gamma_-$ to the ARI provides a lower bound to the value function. In order to utilize this property to resolve the choice of $K_{12,1}^{\star}$, the next lemma describes the block structure of any $K_N \in \Gamma_-$.

Lemma 3.5.4. *Suppose Assumption 3.5.1 holds and the state space is decomposed as in* (3.26)*. Then for all $K_N \in \Gamma_-$, K_N has the block form*

$$K_N = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & K_{1,22}^- & K_{1,23}^- & K_{12,2} \\ 0 & K_{1,23}^{-\top} & K_{1,33} & K_{12,3} \\ 0 & K_{12,2}^\top & K_{12,3}^\top & K_2 \end{bmatrix}.$$

Proof. Let $K_N \in \Gamma_-$ have the block form in (3.28). Since $\Gamma_- \neq \emptyset$ and (A, B) is stabilizable, we can apply Lemma 3.4.5 to obtain that the minimal solution $K_1^- \in \partial\Gamma_{1-}$ exists. Also $\partial\Gamma_{1-} \subset \partial\Gamma_1 \neq \emptyset$. Because $K_N \in \Gamma_- \subset \Gamma$, by Theorem 2.4.1 we establish that its upper left block satisfies $K_1 \in \Gamma_1$. Since (A_1, B_1) is controllable and $\partial\Gamma_1 \neq \emptyset$, we can apply Theorem 3.4.1(i) to get that the maximal solution $K_1^+ \in \partial\Gamma_1$ also exists. Moreover, Theorem 3.4.1(i) also implies that $K_1^-, K_1^+ \in \Gamma_1$, and consequently $K_1^- \leq K_1 \leq K_1^+$. Since $\partial\Gamma_1 \neq \emptyset$, it has been shown (see equation (5.6) in [105] and equation (5.4) in [99]) that K_1^+, K_1^-, and Δ_1 have the block form

$$K_1^+ = \begin{bmatrix} K_{1,11}^+ & 0 & 0 \\ 0 & K_{1,22}^+ & K_{1,23}^+ \\ 0 & K_{1,23}^{+\top} & K_{1,33}^+ \end{bmatrix}, \quad K_1^- = \begin{bmatrix} 0 & 0 & 0 \\ 0 & K_{1,22}^- & K_{1,23}^- \\ 0 & K_{1,23}^{-\top} & K_{1,33}^- \end{bmatrix}, \quad \Delta_1 = \begin{bmatrix} \Delta_{1,11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Delta_{1,33} \end{bmatrix}, \tag{3.37}$$

where $K_{1,22}^+ = K_{1,22}^-$, $K_{1,23}^+ = K_{1,23}^-$, and $\Delta_{1,33} = K_{1,33}^+ - K_{1,33}^-$. Now consider $K_1 \geq K_1^-$ in block form, assuming the decomposition of K_1^- in (3.37). We have

$$K_1 - K_1^- = \begin{bmatrix} K_{1,11} - 0 & K_{1,12} - 0 & K_{1,13} - 0 \\ (K_{1,12} - 0)^\top & K_{1,22} - K_{1,22}^- & K_{1,23} - K_{1,23}^- \\ (K_{1,13} - 0)^\top & (K_{1,23} - K_{1,23}^-)^\top & K_{1,33} - K_{1,33}^- \end{bmatrix} \geq 0\,.$$

Using Theorem 2.4.1, we find $K_{1,11} \geq 0$. Since $K_N \in \Gamma_-$ by assumption, $K_N \leq 0$. Applying Theorem 2.4.1 to $K_N = \begin{bmatrix} K_{1,11} & * \\ * & * \end{bmatrix}$, we get $K_{1,11} \leq 0$. Thus $K_{1,11} = 0$. Now consider again $K_1^- \leq K_1 \leq K_1^+$

with the information that $K_{1,11} = 0$:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & K_{1,22}^- & K_{1,23}^- \\ 0 & (K_{1,23}^-)^\top & K_{1,33}^- \end{bmatrix} \leq \begin{bmatrix} 0 & K_{1,12} & K_{1,13} \\ K_{1,12}^\top & K_{1,22} & K_{1,23} \\ K_{1,13}^\top & K_{1,23}^\top & K_{1,33} \end{bmatrix} \leq \begin{bmatrix} K_{1,11}^+ & 0 & 0 \\ 0 & K_{1,22}^- & K_{1,23}^- \\ 0 & (K_{1,23}^-)^\top & K_{1,33}^+ \end{bmatrix}$$

where we have $K_{1,22}^+ = K_{1,22}^-$ and $K_{1,23}^+ = K_{1,23}^-$ as in (3.37). We claim that $K_{1,12} = 0$, $K_{1,13} = 0$, $K_{1,22} = K_{1,22}^-$, and $K_{1,23} = K_{1,23}^-$. First, we have

$$K_1 - K_1^- = \begin{bmatrix} 0 & K_{1,12} & K_{1,13} \\ K_{1,12}^\top & * & * \\ K_{1,13}^\top & * & * \end{bmatrix} \geq 0.$$

Applying Theorem 2.4.1 again, we get $(I - 00^\dagger) \begin{bmatrix} K_{1,12} & K_{1,13} \end{bmatrix} = 0$, so that $K_{1,12} = 0$ and $K_{1,13} = 0$. Then $K_1 - K_1^- \geq 0$ reduces to

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & K_{1,22} - K_{1,22}^- & K_{1,23} - K_{1,23}^- \\ 0 & (K_{1,23} - K_{1,23}^-)^\top & K_{1,33} - K_{1,33}^- \end{bmatrix} \geq 0$$

which implies by Theorem 2.4.1 that

$$\begin{bmatrix} K_{1,22} - K_{1,22}^- & K_{1,23} - K_{1,23}^- \\ (K_{1,23} - K_{1,23}^-)^\top & K_{1,33} - K_{1,33}^- \end{bmatrix} \geq 0.$$

Similarly, $K_1^+ - K_1 \geq 0$ gives

$$\begin{bmatrix} K_{1,22}^- - K_{1,22} & K_{1,23}^- - K_{1,23} \\ (K_{1,23}^- - K_{1,23})^\top & K_{1,33}^+ - K_{1,33} \end{bmatrix} \geq 0. \tag{3.38}$$

Applying Theorem 2.4.1 to the previous two statements, we get $K_{1,22}^- \leq K_{1,22} \leq K_{1,22}^-$, so $K_{1,22} = K_{1,22}^-$. Then rewriting the previous inequality (3.38)

$$\begin{bmatrix} 0 & K_{1,23}^- - K_{1,23} \\ (K_{1,23}^- - K_{1,23})^\top & K_{1,33}^+ - K_{1,33} \end{bmatrix} \geq 0.$$

Applying Theorem 2.4.1, we get $(I - 00^\dagger)(K^-_{1,23} - K_{1,23}) = 0$, so $K_{1,23} = K^-_{1,23}$. So far we have for $K_N \in \Gamma_-$

$$K_N = \begin{bmatrix} 0 & 0 & 0 & K_{12,1} \\ 0 & K^-_{1,22} & K^-_{1,23} & K_{12,2} \\ 0 & (K^-_{1,23})^\top & K_{1,33} & K_{12,3} \\ K^\top_{12,1} & K^\top_{12,2} & K^\top_{12,3} & K_2 \end{bmatrix} \leq 0.$$

Then $-K_N \geq 0$ has the block form:

$$\begin{bmatrix} 0 & 0 & -K_{12,1} \\ 0 & * & * \\ -K^\top_{12,1} & * & * \end{bmatrix} \geq 0.$$

Applying Lemma 2.4.2, we get $K_{12,1} = 0$. □

In the next result we completely characterize the form of $K^\star$. Before proceeding with this result, we collect some well known results about the cost function.

Theorem 3.5.5. *Consider the system* (3.1) *with the cost function* (3.2) - (3.3). *Let* $x_0 \in \mathbb{R}^n$, $T \geq 0$, *and* $u \in L^m_{2,loc}(\mathbb{R}^+)$.

(i) Let $K \in \partial\Gamma$. *Then* $J_T(x_0, u) = \int_0^T \|u(t) + B^\top Kx(t)\|^2\, dt + x_0^\top Kx_0 - x^\top(T)Kx(T)$, *where* $x(t) := x(t; x_0, u)$.

(ii) For all $x_0 \in \mathbb{R}^n$ *and* $K_N \in \Gamma_-$, $V_{\mathcal{L}}(x_0) \geq x_0^\top K_N x_0$.

(iii) Suppose Assumption 3.5.1 holds. If $J(x_0, u) = x_0^\top K^\star x_0$, *then* $u = -B^\top K^\star x$ *and* $\lim_{T\to\infty} x^\top(T)K^\star x(T) = 0$.

Proof. Statement (i) is standard. See for instance [114] or [105]. Statement (ii) is Proposition 1.8 of [37]. See also Lemma 4.4 of [105]. Statement (iii) is Theorem 2.8(c) of [37]. See also the proof of Theorem 5.1(iii) in [105]. □

Theorem 3.5.6. *Consider the* $(LQCP)_{\mathcal{L}}$ *and suppose Assumption 3.5.1 holds. Then in the state space decomposition* (3.26), $K^\star \in \partial\Gamma$ *has the form*

$$K^\star = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & K^-_{1,22} & K^-_{1,23} & K^\star_{12,2} \\ 0 & K^{-\top}_{1,23} & K^+_{1,33} & K^\star_{12,3} \\ 0 & (K^\star_{12,2})^\top & (K^\star_{12,3})^\top & K^\star_2 \end{bmatrix}, \tag{3.39}$$

where $K_{12,2}^\star$ is the unique solution to (3.35)*, $K_{12,3}^\star$ is the unique solution to* (3.36)*, and $K_2^\star$ is the unique solution to* (3.19) *with $K_{12} = K_{12}^\star$.*

Proof. By Theorem 3.5.3, $K_1^\star = \overline{K}_1$ with the form of $\overline{K}_1$ given in (3.22). By Theorem 3.5.2, $K^\star \in \partial\Gamma$. Next we consider (3.18). Using the decompositions above and with the choice $K_1 = \overline{K}_1$, the second ARE equation (3.18) splits into (3.34), (3.35), and (3.36). Since $\sigma(\overline{A}_{1,22}) \subset \mathbb{C}^0$, $\sigma(\overline{A}_{1,33}) \subset \mathbb{C}^-$, and $\sigma(-A_2) \subset \mathbb{C}^+$, (3.35) and (3.36) have unique solutions $K_{12,2}^\star$ and $K_{12,3}^\star$, respectively [34]. Similarly, (3.19) has a unique solution $K_2^\star$, assuming $K_{12} = K_{12}^\star$. At this point we know that $K^\star$ has the block form:

$$K^\star = \begin{bmatrix} 0 & 0 & 0 & K_{12,1}^\star \\ 0 & K_{1,22}^- & K_{1,23}^- & K_{12,2}^\star \\ 0 & K_{1,23}^{-\top} & K_{1,33}^+ & K_{12,3}^\star \\ (K_{12,1}^\star)^\top & (K_{12,2}^\star)^\top & (K_{12,3}^\star)^\top & K_2^\star \end{bmatrix}. \tag{3.40}$$

Comparing to (3.39), it remains only to show that $K_{12,1}^\star = 0$. By Theorem 3.5.2, $V_{\mathcal{L}}(x_0) = x_0^\top K^\star x_0$. Let $K_N \in \Gamma_-$. By Theorem 3.5.5(ii), for all $x_0 \in \mathbb{R}^n$, $V_{\mathcal{L}}(x_0) = x_0^\top K^\star x_0 \geq x_0^\top K_N x_0$; that is, $K^\star \geq K_N$. Using the block form of $K^\star$ in (3.40) and the block form of K_N in Lemma 3.5.4, we have

$$K^\star - K_N = \begin{bmatrix} 0 & 0 & 0 & K_{12,1}^\star \\ 0 & 0 & 0 & K_{12,2}^\star - K_{12,2N} \\ 0 & 0 & K_{1,33}^+ - K_{1,33N} & K_{12,3}^\star - K_{12,3N} \\ (K_{12,1}^\star)^\top & (K_{12,2}^\star - K_{12,2N})^\top & (K_{12,3}^\star - K_{12,3N})^\top & K_2^\star - K_{2N} \end{bmatrix} \geq 0\,.$$

Applying Lemma 2.4.2 yields that $K_{12,1}^\star = 0$, as desired. □

Remark 3.5.7. *We observe from the form of $K^\star$ that $K_{12,1}^\star = 0$. If we substitute $K_{12,1}^\star = 0$ into* (3.34)*, we get that $Q_{12,1} = 0$. One can derive the fact that $Q_{12,1} = 0$ via a separate argument, and this provides an independent validation of our result that $K_{12,1}^\star = 0$. Suppose Assumption 3.5.1 holds. Take any symmetric K with the special form:*

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & K_{1,22} & K_{1,23} & K_{12,2} \\ 0 & K_{1,23}^\top & K_{1,33} & K_{12,3} \\ 0 & K_{12,2}^\top & K_{12,3}^\top & K_2 \end{bmatrix}.$$

We decompose A and B as in (3.27)*. Using a result analogous to equation (5.2) in [105], it can be shown*

that $\mathcal{N}_1(\mathcal{L}_1)$ is A_1-invariant, and this implies $A_{1,21} = A_{1,31} = 0$. Then by direct computation $\phi(K)$ has the form:

$$\phi(K) = \begin{bmatrix} Q_{1,11} & Q_{1,12} & Q_{1,13} & Q_{12,1} \\ Q_{1,12}^\top & * & * & * \\ Q_{1,13}^\top & * & * & * \\ Q_{12,1}^\top & * & * & * \end{bmatrix}.$$

Now choose the upper left block of the above K to be $K_1^- \in \partial\Gamma$. By (3.37) this choice is consistent with the form of K above. Since the upper left block of $\phi(K)$ is written as $\phi_1(K_1)$ and we know that $\phi_1(K_1^-) = 0$, it immediately follows that $Q_{1,11} = 0$, $Q_{1,12} = 0$, and $Q_{1,13} = 0$. Next, since $\Gamma_- \neq \emptyset$, let $K_N \in \Gamma_-$. By Lemma 3.5.4, K_N has the special form above. Then we have

$$\phi(K_N) = \begin{bmatrix} 0 & 0 & 0 & Q_{12,1} \\ 0 & * & * & * \\ 0 & * & * & * \\ Q_{12,1}^\top & * & * & * \end{bmatrix} \geq 0.$$

By applying Lemma 2.4.2, we conclude that $Q_{12,1} = 0$.

We conclude this section by applying Theorem 3.5.6 to obtain necessary and sufficient conditions for attainability of the $(\text{LQCP})_{\mathcal{L}}$. Remarkably, the attainability result depends only on the controllable subspace.

Theorem 3.5.8. *Suppose Assumption 3.5.1 holds and the state space is decomposed as in* (3.12)*. Then the $(LQCP)_{\mathcal{L}}$ is attainable if and only if* $\operatorname{Ker}(\Delta_1) \subset \mathcal{L}_1 \cap \operatorname{Ker}(K_1^-)$.

Proof. Due to Assumption 3.5.1, we may further assume that the state space is decomposed according to (3.26). Let $W_1 \in \mathbb{R}^{n_1 \times n_1}$ be a matrix such that $\operatorname{Ker}(W_1) = \mathcal{L}_1$ and let $d_{1\mathcal{L}_1} : \mathbb{R}^{n_1} \to [0, \infty)$ be the distance function in $\mathbb{R}^{n_1}$ to $\mathcal{L}_1$. Since $\mathcal{X}_{1,1} = \langle \mathcal{L}_1 \cap \operatorname{Ker}(K_1^-) \mid A_1(K_1^-) \rangle \cap \mathcal{X}^+(A_1(K_1^-))$, we have $\mathcal{X}_{1,1} \subset \langle \operatorname{Ker}(W_1) \cap \operatorname{Ker}(K_1^-) \mid A_1(K_1^-) \rangle \subset \operatorname{Ker}(W_1) \cap \operatorname{Ker}(K_1^-) = \operatorname{Ker}\left(\begin{bmatrix} K_1^- \\ W_1 \end{bmatrix} \right)$. We claim

$$\begin{bmatrix} K_1^- \\ W_1 \end{bmatrix} = \begin{bmatrix} 0 & D_2 & D_3 \end{bmatrix}. \tag{3.41}$$

Proof of Claim: Let $x_1 \in \mathcal{X}_{1,1}$. Then $x_1 \in \operatorname{Ker}\left(\begin{bmatrix} K_1^- \\ W_1 \end{bmatrix} \right) =: \operatorname{Ker}\left(\begin{bmatrix} D_1 & D_2 & D_3 \end{bmatrix} \right)$. Also since

$x_1 \in \mathcal{X}_{1,1}$, in coordinates it has the form $x_1 = (x_{1,1}, 0, 0)$. Then $\begin{bmatrix} D_1 & D_2 & D_3 \end{bmatrix} x_1 = D_1 x_{1,1} = 0$. Since $x_{1,1}$ is arbitrary, we get $D_1 = 0$, as desired.

($\Rightarrow$) Suppose the $(\mathrm{LQCP})_{\mathcal{L}}$ is attainable. Let $x_0 \in \mathbb{R}^n$. By definition there exists $u^\star \in \mathcal{U}_{\mathcal{L}}(x_0)$ such that $V_{\mathcal{L}}(x_0) = J(x_0, u^\star)$. By Theorem 3.5.2 we know $V_{\mathcal{L}}(x_0) = x_0^\top K^\star x_0$ where $K^\star \in \partial\Gamma$, and by Theorem 3.5.6, $K^\star$ is given in (3.39). Now we can apply Theorem 3.5.5(iii) to get $u^\star = -B^\top K^\star x$. The closed-loop dynamics are $\dot{x} = A(K^\star)x$. Let $x := (x_1, x_2) := (x_{1,1}, x_{1,2}, x_{1,3}, x_2)$ according to the decomposition (3.26). Then using the block form of $A_1(K_1^\star) = A(\overline{K}_1)$ in (3.33), we have

$$\dot{x} = \begin{bmatrix} A_1(K_1^\star) & * \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \overline{A}_{1,11} & 0 & 0 & \overline{A}_{12,1} \\ 0 & \overline{A}_{1,22} & 0 & \overline{A}_{12,2} \\ 0 & 0 & \overline{A}_{1,33} & \overline{A}_{12,3} \\ 0 & 0 & 0 & A_2 \end{bmatrix} \begin{bmatrix} x_{1,1} \\ x_{1,2} \\ x_{1,3} \\ x_2 \end{bmatrix}, \tag{3.42}$$

where $\sigma(\overline{A}_{1,11}) \subset \mathbb{C}^+$, $\sigma(\overline{A}_{1,22}) \subset \mathbb{C}^0$, $\sigma(\overline{A}_{1,33}) \subset \mathbb{C}^-$, and by stabilizability, $\sigma(A_2) \subset \mathbb{C}^-$. Using the variation of constants formula we get that at $t = T$

$$x_{1,i}(T) = \mathrm{e}^{\overline{A}_{1,ii}T} x_{1,i}(0) + \int_0^T \mathrm{e}^{\overline{A}_{1,ii}(T-\tau)} \overline{A}_{12,i} \mathrm{e}^{A_2 \tau} x_2(0)\, d\tau, \quad i = 1, 2, 3\,. \tag{3.43}$$

Since $\sigma(A_2) \subset \mathbb{C}^-$, $\lim_{T\to\infty} x_2(T) = 0$. Using (3.43) for $i = 3$, $\sigma(\overline{A}_{1,33}) \subset \mathbb{C}^-$, and the fact that $\lim_{T\to\infty} x_2(T) = 0$, we also get $\lim_{T\to\infty} x_{1,3}(T) = 0$. Now using (3.39), the block form of K_1^- given in (3.37), and the fact that $K_{1,33}^+ = \Delta_{1,33} + K_{1,33}^-$, we have

$$\begin{aligned} x^\top K^\star x &= x_1^\top K_1^\star x_1 + 2x_1^\top K_{12}^\star x_2 + x_2^\top K_2^\star x_2 \\ &= x_1^\top K_1^- x_1 + x_{1,3}^\top \Delta_{1,33} x_{1,3} + 2(x_{1,2}^\top K_{12,2}^\star + x_{1,3}^\top K_{12,3}^\star) x_2 + x_2^\top K_2^\star x_2\,. \end{aligned} \tag{3.44}$$

Using this expression combined with the fact that $\lim_{T\to\infty} x_{1,3}(T) = 0$, $\lim_{T\to\infty} x_2(T) = 0$, and $\lim_{T\to\infty} x^\top(T) K^\star x(T) = 0$ from Theorem 3.5.5(iii), we get

$$\lim_{T\to\infty} x^\top(T) K^\star x(T) = \lim_{T\to\infty} \left(x_1^\top(T) K_1^- x_1(T) + 2x_{1,2}^\top(T) K_{12,2}^\star x_2(T) \right) = 0\,. \tag{3.45}$$

Now we observe that $\lim_{T\to\infty} 2x_{1,2}^\top(T) K_{12,2}^\star x_2(T) = 0$ because $\sigma(\overline{A}_{1,22}) \subset \mathbb{C}^0$ and $\sigma(A_2) \subset \mathbb{C}^-$. Returning to (3.45), this implies that also $\lim_{T\to\infty} x_1^\top(T) K_1^- x_1(T) = 0$.

We have assumed that $u^\star \in \mathcal{U}_{\mathcal{L}}(x_0)$ and (A, B) is stabilizable. Therefore, $\lim_{T\to\infty} d_{\mathcal{L}\cap\mathcal{C}}(x(T)) = 0$, and thus within the controllable subspace $\lim_{T\to\infty} d_{1\mathcal{L}_1}(x_1(T)) = 0$. Since $\mathcal{L}_1 = \mathrm{Ker}(W_1)$, we have

$\lim_{T\to\infty} W_1 x_1(T) = 0$. Meanwhile by Lemma 3.4.5, $K_1^- \leq 0$. Since $\lim_{T\to\infty} x_1^\top(T) K_1^- x_1(T) = 0$, by taking the limit in (3.11) we have that $\lim_{T\to\infty} K_1^- x_1(T) = 0$. Overall, we have $\lim_{T\to\infty} \begin{bmatrix} K_1^- \\ W_1 \end{bmatrix} x_1(T) = 0$. Using (3.41), this gives $\lim_{T\to\infty} (D_2 x_{1,2}(T) + D_3 x_{1,3}(T)) = 0$. We already know that $\lim_{T\to\infty} x_{1,3}(T) = 0$, so we get $\lim_{T\to\infty} D_2 x_{1,2}(T) = 0$. However, $\sigma(\overline{A}_{1,22}) \subset \mathbb{C}^0$ and $x_{1,2}(0)$ is arbitrary, so $D_2 = 0$. Finally, we observe that if $x_1 \in \mathcal{X}_{1,2}$, then $\begin{bmatrix} 0 & 0 & D_3 \end{bmatrix} x_1 = 0$ since $x_1 = (0, x_{1,2}, 0)$. That is, $\mathcal{X}_{1,2} \subset \operatorname{Ker}\left(\begin{bmatrix} 0 & 0 & D_3 \end{bmatrix}\right)$. In sum, we have

$$\mathcal{X}_{1,2} = \operatorname{Ker}(\Delta_1) \subset \operatorname{Ker}\left(\begin{bmatrix} 0 & 0 & D_3 \end{bmatrix}\right) = \operatorname{Ker}\left(\begin{bmatrix} K_1^- \\ W_1 \end{bmatrix}\right) = \mathcal{L}_1 \cap \operatorname{Ker}(K_1^-)\,. \tag{3.46}$$

($\Leftarrow$) Suppose that $\operatorname{Ker}(\Delta_1) \subset \mathcal{L}_1 \cap \operatorname{Ker}(K_1^-)$. Let $x_0 \in \mathbb{R}^n$. To show attainability, we must find an optimal control. Consider the candidate $u^c := -B^\top K^\star x$, where $K^\star$ is given in (3.39). We must show $V_{\mathcal{L}}(x_0) = J(x_0, u^c)$ and $u^c \in \mathcal{U}_{\mathcal{L}}(x_0)$. The closed-loop dynamics using u^c are given in (3.42). Following the same arguments as above we have that $\lim_{T\to\infty} x_2(T) = 0$ and $\lim_{T\to\infty} x_{1,3}(T) = 0$. By assumption, $\operatorname{Ker}(\Delta_1) \subset \mathcal{L}_1 \cap \operatorname{Ker}(K_1^-)$. From above, $\mathcal{L}_1 \cap \operatorname{Ker}(K_1^-) = \operatorname{Ker}\left(\begin{bmatrix} K_1^- \\ W_1 \end{bmatrix}\right) = \operatorname{Ker}\left(\begin{bmatrix} 0 & D_2 & D_3 \end{bmatrix}\right)$. We claim that $D_2 = 0$. To see this, let $x_1 \in \operatorname{Ker}(\Delta_1) = \mathcal{X}_{1,2}$. Then $x_1 = (0, x_{1,2}, 0)$. Since $\operatorname{Ker}(\Delta_1) \subset \operatorname{Ker}\left(\begin{bmatrix} 0 & D_2 & D_3 \end{bmatrix}\right)$ we have $\begin{bmatrix} 0 & D_2 & D_3 \end{bmatrix} x_1 = D_2 x_{1,2} = 0$. Since $x_{1,2}$ is arbitrary, $D_2 = 0$. Using the block form of K_1^- in (3.37), we have

$$\begin{bmatrix} K_1^- \\ W_1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & K_{1,22}^- & K_{1,23}^- \\ 0 & K_{1,23}^{-\top} & K_{1,33}^- \\ W_{11} & W_{12} & W_{13} \end{bmatrix} = \begin{bmatrix} 0 & 0 & D_3 \end{bmatrix}.$$

This implies $K_{1,22}^- = K_{1,23}^- = 0$. Now we observe $K^\star \in \partial\Gamma$ by Theorem 3.5.2 and $u^c \in L_{2,loc}^m(\mathbb{R}^+)$ for any fixed $T \geq 0$. Therefore, we can apply Theorem 3.5.5(i) with $K = K^\star$ and $u = u^c$ to get

$$J_T(x_0, u^c) = x_0^\top K^\star x_0 - x(T)^\top K^\star x(T)\,. \tag{3.47}$$

We claim that $\lim_{T\to\infty} x^\top(T) K^\star x(T) = 0$. Using the expansion of $x(T)^\top K^\star x(T)$ given in (3.44), and the fact that $\lim_{T\to\infty} x_2(T) = 0$ and $\lim_{T\to\infty} x_{1,3}(T) = 0$, we get $\lim_{T\to\infty} x^\top(T) K^\star x(T) = \lim_{T\to\infty} x_1^\top(T) K_1^- x_1(T)$. Using the available information about the block form of K_1^- and that the

limit $\lim_{T\to\infty} x_{1,3}(T) = 0$, we find

$$\lim_{T\to\infty} x^\top(T)K^\star x(T) = \lim_{T\to\infty} x_1^\top(T)\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & K_{1,33}^- \end{bmatrix} x_1(T) = \lim_{T\to\infty} x_{1,3}^\top(T)K_{1,33}^- x_{1,3}(T) = 0\,.$$

Returning to (3.47), we have $\lim_{T\to\infty} J_T(x_0, u^c) = J(x_0, u^c) = x_0^\top K^\star x_0$, as desired.

Finally, we must show $u^c \in \mathcal{U}_\mathcal{L}(x_0)$, and particularly $\lim_{T\to\infty} d_\mathcal{L}(x(T)) = 0$. Since $\lim_{T\to\infty} x_{1,3}(T) = 0$, we have that

$$\lim_{T\to\infty} \begin{bmatrix} K_1^- \\ W_1 \end{bmatrix} x_1(T) = \lim_{T\to\infty} \begin{bmatrix} 0 & 0 & D_3 \end{bmatrix} x_1(T) = D_3 x_{1,3}(T) = 0.$$

Thus, $\lim_{T\to\infty} W_1 x_1(T) = 0$, which implies $\lim_{T\to\infty} d_{1\mathcal{L}_1}(x_1(T)) = 0$. Since $\mathcal{L}_1 = \mathcal{L} \cap \mathcal{C}$ and $\lim_{T\to\infty} x_2(T) = 0$, we have $\lim_{T\to\infty} d_\mathcal{L}(x(T)) = 0$. Thus, $u^c \in \mathcal{U}_\mathcal{L}(x_0)$, as desired. □

We collect all of the previous results to obtain the culminating result on the solution of the $(\text{LQCP})_\mathcal{L}$. It is a generalization of Theorem 3.4.3 for the case of (A, B) controllable to the case when (A, B) is stabilizable.

Theorem 3.5.9. *Consider the $(LQCP)_\mathcal{L}$. Suppose Assumption 3.5.1 holds and the state space is decomposed as in* (3.12)*. Then we have*

(i) The problem is well-posed.

(ii) For all $x_0 \in \mathbb{R}^n$, $V_\mathcal{L}(x_0) = x_0^\top K^\star x_0$.

(iii) For all $x_0 \in \mathbb{R}^n$, the problem is attainable if and only if $\operatorname{Ker}(\Delta_1) \subset \mathcal{L}_1 \cap \operatorname{Ker}(K_1^-)$.

(iv) If the problem is attainable, then for each $x_0 \in \mathbb{R}^n$, there exists exactly one optimal input $u^\star$, and it is given by $u^\star = -B^\top K^\star x$.

Proof. Statements (i) and (ii) follow from Theorem 3.5.2. The form of $K^\star$ follows from Theorem 3.5.6. Statement (iii) is an immediate consequence of Theorem 3.5.8. Statement (iv) follows from Theorem 3.5.5 (iii). □

3.6 Discussion

In this section we discuss several special cases of our main result. This includes a comparison with classical results in the positive semidefinite case. First, we consider the special case when $\mathcal{N}_1(\mathcal{L}_1) = 0$

which was also treated in Theorem 6.1 of [105]. From our experience it is only in exceptional cases that $\mathcal{N}_1(\mathcal{L}_1) \neq 0$. The following result shows that when $\mathcal{N}_1(\mathcal{L}_1) = 0$, then $K^\star = K^+$, the maximal solution in $\partial\Gamma$. This result has practical significance because there are many powerful algorithms for numerically finding the maximal solution of the ARE.

Theorem 3.6.1. *Consider the $(LQCP)_{\mathcal{L}}$, suppose that Assumption 3.5.1 holds, and that the state space decomposed as in* (3.12)*. Then $\mathcal{N}_1(\mathcal{L}_1) = 0$ if and only if $K^* = K^+$, where $K^+ \in \partial\Gamma$ is the maximal solution.*

Proof. (Only if) Suppose $\mathcal{N}_1(\mathcal{L}_1) = 0$. By Theorem 3.5.6, $K^\star := \begin{bmatrix} K_1^\star & K_{12}^\star \\ K_{12}^{\star\top} & K_2^\star \end{bmatrix} \in \partial\Gamma$, where $K_1^\star = \overline{K}_1 = \gamma(\mathcal{N}_1(\mathcal{L}_1))$. By assumption, $P_{\mathcal{N}_1(\mathcal{L}_1)} = 0$, and then (3.22) gives $K_1^\star = K_1^+$, where K_1^+ is the maximal solution in $\partial\Gamma_1$. By Theorem 3.4.1(ii), we also know $K_1^+ \in \partial\Gamma_1$ is the unique maximal solution such that $\sigma(A_1(K_1^+)) \subset \mathbb{C}^- \cup \mathbb{C}^0$. Furthermore, by stabilizability, $\sigma(A_2) \subset \mathbb{C}^-$. Therefore, $\sigma(A_1(K_1^+)) \cap \sigma(-A_2) = \emptyset$, so $K_{12}^\star$ is the unique solution of the Sylvester equation (3.18). Similarly, since $\sigma(A_2^\top) \cap \sigma(-A_2) = \emptyset$, $K_2^\star$ is the unique solution of the Sylvester equation (3.19).

Meanwhile, since $\partial\Gamma \neq \emptyset$, by Theorem 3.4.4, the maximal solution $K^+ \in \partial\Gamma$ exists and satisfies $\sigma(A(K^+)) \subset \mathbb{C}^- \cup \mathbb{C}^0$. We claim $K^\star = K^+$. Let $K^+ = \begin{bmatrix} K_1 & K_{12} \\ K_{12}^\top & K_2 \end{bmatrix}$ in block form. Since $K^+ \in \partial\Gamma$, we have that $K_1 \in \partial\Gamma_1$. Using (3.13), $\sigma(A(K^+)) = \sigma(A_1(K_1)) \uplus \sigma(A_2) \subset \mathbb{C}^- \cup \mathbb{C}^0$. Then since $\sigma(A_2) \subset \mathbb{C}^-$, we have $\sigma(A_1(K_1)) \subset \mathbb{C}^- \cup \mathbb{C}^0$. However, by Theorem 3.4.1(ii), $K_1 \in \partial\Gamma_1$ and $\sigma(A_1(K_1)) \subset \mathbb{C}^- \cup \mathbb{C}^0$ together imply $K_1 = K_1^+ = K_1^\star$, the unique maximal solution in $\partial\Gamma_1$. It immediately follows that $K_{12} = K_{12}^\star$ and $K_2 = K_2^\star$, as desired.

(If) Suppose $K^* = K^+$, the maximal solution in $\partial\Gamma$. By writing K^+ in block form, $K^+ = \begin{bmatrix} K_1 & K_{12} \\ K_{12}^\top & K_2 \end{bmatrix}$, we have $K_1 = K_1^\star$. We also have that $K_1 = K_1^+$ is the maximal solution in $\partial\Gamma_1$ using an argument analogous to the one above. That is, using (3.13), $\sigma(A(K^+)) = \sigma(A_1(K_1)) \uplus \sigma(A_2)$. By Theorem 3.4.4, $\sigma(A(K^+)) \subset \mathbb{C}^- \cup \mathbb{C}^0$. Since $\sigma(A_2) \subset \mathbb{C}^-$, we get $\sigma(A_1(K_1)) \subset \mathbb{C}^- \cup \mathbb{C}^0$. Then by Theorem 3.4.1(ii), $K_1 = K_1^+ \in \partial\Gamma_1$. Meanwhile by Theorem 3.5.3, $\overline{K}_1 = K_1^\star$. Putting this altogether, we have that $\overline{K}_1 = K_1^\star = K_1^+$. Finally, using $\overline{K}_1 = K_1^+$ in (3.22) gives that $P_{\mathcal{N}_1(\mathcal{L}_1)} = 0$, so $\mathcal{N}_1(\mathcal{L}_1) = 0$. □

Next we discuss how Theorem 3.5.9 recovers well known results for the free-endpoint and fixed-endpoint problems when Q is positive semidefinite and (A, B) is stabilizable. First, we observe that when $Q \geq 0$, then $\phi(0) \geq 0$ so $0 \in \Gamma_- \neq \emptyset$. Therefore, Assumption 3.5.1 holds. We also assume that the state space is decomposed as in (3.26) wherever needed.

The main results on the free endpoint problem are summarized in Theorem 10.13 in [106]. In

particular, when $\mathcal{L} = \mathbb{R}^n$, $V_{\mathcal{L}}(x_0) = x_0^\top P^- x_0$, where $P^- \geq 0$ is the smallest positive semidefinite solution to the ARE, and the optimal control is $u^\star(t) = -B^\top P^- x(t)$. We would like to verify that our Theorem 3.5.9 recovers these results. We will show that when $Q \geq 0$, $K^\star$ given in (3.39) satisfies $K^\star = P^-$. To aid in this endeavor, we invoke a result from [105]. Let $\partial\Gamma_{1+} := \{K_1 \in \partial\Gamma_1 \mid K_1 \geq 0\}$.

Theorem 3.6.2 (Theorem 6.3 [105])**.** *Assume (A_1, B_1) is controllable and $\partial\Gamma_{1-} \neq \emptyset$. Then the following hold: if $\partial\Gamma_{1+} \neq \emptyset$, then (i) $\overline{K}_1 \in \partial\Gamma_{1+}$ and (ii) $K_1 \in \partial\Gamma_{1+}$ implies $\overline{K}_1 \leq K_1$.*

Lemma 3.6.3. *Consider the $(LQCP)_{\mathcal{L}}$. Suppose (A, B) is stabilizable, $\mathcal{L} = \mathbb{R}^n$, and $Q \geq 0$. Then $K^\star = P^-$.*

Proof. We begin by applying Theorem 3.6.2 to show that $\overline{K}_1$ is the smallest solution in $\partial\Gamma_{1+}$. To that end, we must show that $\partial\Gamma_{1-} \neq \emptyset$ and $\partial\Gamma_{1+} \neq \emptyset$. First, since Assumption 3.5.1 holds, we can apply Lemma 3.4.5 to get $K_1^- \in \partial\Gamma_{1-}$ exists, so $\partial\Gamma_{1-} \neq \emptyset$. Second, because $Q \geq 0$, we know $\phi(0) \geq 0$, so $0 \in \Gamma_-$. By Theorem 3.5.2, $V_{\mathcal{L}}(x) = x^\top K^\star x$. Applying Theorem 3.5.5(ii) with $K_N = 0$, we get $x^\top K^\star x \geq x^\top 0 x = 0$, for all $x \in \mathbb{R}^n$, so $K^\star \geq 0$. That is, $K^\star \in \partial\Gamma_+$. By Theorem 2.4.1, this implies $K_1^\star \geq 0$, so $K_1^\star = \overline{K}_1 \in \partial\Gamma_{1+} \neq 0$. Now we can apply Theorem 3.6.2 to get $K_1^\star = \overline{K}_1$ is the smallest solution in $\partial\Gamma_{1+}$.

It remains to show that $K^\star = P^-$ is the smallest solution in $\partial\Gamma_+$. To arrive at a contradiction, suppose there exists $K \in \partial\Gamma_+$ such that $K \neq K^\star$ and $K \leq K^\star$. There are two cases. First, suppose $K \in \partial\Gamma_+$ with $K \leq K^\star$ such that $K_1 \neq K_1^\star$, where K_1 is the upper left block of K. Since $K \in \partial\Gamma$, $\phi(K) = 0$, so $\phi_1(K_1) = 0$, implying $K_1 \in \partial\Gamma_1$. By Theorem 2.4.1, $K \geq 0$ implies $K_1 \geq 0$, so $K_1 \in \partial\Gamma_{1+}$. Again by Theorem 2.4.1, $K \leq K^\star$ implies $K_1 \leq K_1^\star$. Thus, we have $K_1 \in \partial\Gamma_{1+}$ such that $K_1 \leq K_1^\star$, which contradicts that $K_1^\star$ is the smallest solution in $\partial\Gamma_{1+}$.

For the second case, suppose $K \in \partial\Gamma_+$ with $K \leq K^\star$ such that $K_1 = K_1^\star$. By (3.33), K has the form

$$K = \begin{bmatrix} 0 & 0 & 0 & K_{12,1} \\ 0 & K_{1,22}^- & K_{1,23}^- & K_{12,2} \\ 0 & K_{1,23}^{-\top} & K_{1,33}^+ & K_{12,3} \\ K_{12,1}^\top & K_{12,2}^\top & K_{12,3}^\top & K_2 \end{bmatrix}.$$

Since $K \geq 0$, we can apply Lemma 2.4.2 to find that $K_{12,1} = 0$. Then since $K_1 = K_1^\star$, $K_{12,1} = K_{12,1}^\star = 0$, and $\phi(K) = 0$, the solutions for $K_{12,2}$ and $K_{12,3}$ are unique and match $K_{12,2}^\star$ and $K_{12,3}^\star$, respectively. Thus $K = K^\star$, a contradiction. We conclude that $K^\star$ is the smallest solution in $\partial\Gamma_+$. This proves that for the free endpoint case when $Q \geq 0$ that $V_{\mathcal{L}}(x_0) = x_0^\top P^- x_0$. Also, Theorem 3.5.9 (iv) gives the optimal control $u(t) = -B^\top P^- x(t)$ since $P^- = K^\star$. □

Next we consider attainability in the free endpoint case. Since Assumption 3.5.1 holds, we can apply Theorem 3.5.9(iii). In the free endpoint problem, $\mathcal{L}_1 = \mathcal{C}$, so by Theorem 3.5.9(iii), the problem is attainable if and only if $\mathrm{Ker}(\Delta_1) \subset \mathrm{Ker}(K_1^-)$. By Proposition 6.4 of [105], the latter condition always holds. Thus, we recover the well-known fact that for the free endpoint case in the positive semidefinite case, the problem is always attainable.

Now we discuss the fixed endpoint problem. The main results are summarized in Theorem 10.18 in [106]. In particular, when $\mathcal{L} = 0$, $V_{\mathcal{L}}(x_0) = x_0^\top P^+ x_0$, where $P^+ \geq 0$ is the largest positive semidefinite solution to the ARE, and the optimal control is $u^\star(t) = -B^\top P^+ x(t)$. We would like to verify that our Theorem 3.5.9 recovers these results. We must show that when $Q \geq 0$, then $K^\star = P^+$. For the fixed endpoint problem, $\mathcal{L}_1 = 0$, so $\mathcal{N}_1(\mathcal{L}_1) = 0$. The desired result is then immediately obtained from Theorem 3.6.1.

Now we consider attainability in the fixed endpoint case. The well-known necessary and sufficient conditions for attainability in the positive semidefinite case, stated in Theorem 10.18(iii) of [106], is that every eigenvalue of A on the imaginary axis is (Q, A) observable. We must show that this statement is equivalent to our attainability result in Theorem 3.5.9(iii), which for the fixed-endpoint case requires that $\mathrm{Ker}(\Delta_1) \subset 0 \cap \mathrm{Ker}(K_1^-)$, or equivalently, $\Delta_1 > 0$. This connection is resolved in Theorem 3.6.6, requiring the notion of the Hamiltonian and several preliminary results.

Consider the Kalman controllable decomposition (3.13). First, we define the *Hamiltonian matrix* on the controllable subspace:

$$H_1 := \begin{bmatrix} A_1 & -B_1 B_1^\top \\ -Q_1 & -A_1^\top \end{bmatrix}. \tag{3.48}$$

Theorem 3.6.4 ([78])**.** *Let (A_1, B_1) be controllable. The following conditions are equivalent.*

(i) The maximal and minimal solutions of the ARE (and ARI), K_1^+ and K_1^- respectively, exist and $\Delta_1 > 0$.

(ii) The Hamiltonian matrix has no pure imaginary eigenvalues, i.e., if $\lambda \in \sigma(H_1)$ then $\Re(\lambda) \neq 0$.

Lemma 3.6.5 (Lemma 8, [62])**.** *Suppose that $Q_1 = C_1^\top C_1$. Then there is an eigenvalue λ of H_1 such that $\Re(\lambda) = 0$ if and only if there is an uncontrollable eigenvalue λ of (A_1, B_1) and/or unobservable eigenvalue λ of (C_1, A_1) such that $\Re(\lambda) = 0$.*

Theorem 3.6.6. *Suppose (A, B) is stabilizable and $Q \geq 0$. Then every eigenvalue of A on the imaginary axis is (Q, A) observable if and only if $\Delta_1 > 0$.*

Proof. First we check that $\Delta_1 = K_1^+ - K_1^-$ is well-defined. Since (A, B) is stabilizable and $Q \geq 0$

implies $0 \in \Gamma_-$, we can apply Lemma 3.4.5 to get that $K_1^- \subset \partial\Gamma_{1-} \subset \partial\Gamma_1 \neq \emptyset$, and so Theorem 3.4.1(i) establishes the existence of K_1^+ as well.

Since $Q \geq 0$, write $Q = C^\top C$. In the Kalman controllability decomposition, this means $C = \begin{bmatrix} C_1 & C_2 \end{bmatrix}$, so that $Q_1 = C_1^\top C_1$.

First we can use Theorem 3.6.4 to establish $\Delta_1 > 0$ is equivalent to H_1 having no pure imaginary eigenvalues. The contrapositive of the Lemma 3.6.5 says that H_1 has no pure imaginary eigenvalues if and only if there is no uncontrollable eigenvalue of (A_1, B_1) *and* no unobservable eigenvalue of (C_1, A_1) on the imaginary axis. Since in our scenario (A_1, B_1) is controllable, this statement is equivalent to H_1 has no pure imaginary eigenvalues if and only if there are no unobservable eigenvalues of (C_1, A_1) on the imaginary axis. Of course, there are no unobservable eigenvalues of (C_1, A_1) on the imaginary axis if and only if all the eigenvalues of A_1 on the imaginary axis are (C_1, A_1) observable. Applying Lemma 2.2.1(i), we get that all the eigenvalues of A_1 on the imaginary axis are (C_1, A_1) observable if and only if all the eigenvalues of A on the imaginary axis are (C, A) observable. Then applying Lemma 2.2.1(ii), all the eigenvalues of A on the imaginary axis are (C, A) observable if and only if all the eigenvalues of A on the imaginary axis are $(C^\top C, A)$ observable. Since $Q = C^\top C$, we have proven every eigenvalue of A on the imaginary axis is (Q, A) observable if and only if $\Delta_1 > 0$ by a long chain of equivalent statements. $\square$

The final verification of our result in the fixed endpoint case is to show that the closed-loop system, $\dot{x}(t) = (A - BB^\top K^+)x(t) = A(K^+)x(t)$, is asymptotically stable, thereby recovering Theorem 10.18(v) in [106]. Note that $A(K) = A - BB^\top K = \begin{bmatrix} A_1(K_1) & * \\ 0 & A_2 \end{bmatrix}$ so that $\sigma(A(K)) = \sigma(A_1(K_1)) \uplus \sigma(A_2)$. By Theorem 5 in [114], we have that $\Delta_1 > 0$ if and only if $\sigma(A_1(K_1^+)) \subset \mathbb{C}^-$. Since $\sigma(A_2) \subset \mathbb{C}^-$ by stabilizability and $\Delta_1 > 0$ by attainability, we have $\sigma(A(K^+)) \subset \mathbb{C}^-$, as desired.

3.7 Examples

In this section, we supply two examples that illustrate the utility of the new theory. The first example is "pathological" in that it exhibits many of the difficulties encountered prior to the new theory, namely when there is no minimal solution in $\partial\Gamma$, $K_{12}^\star$ has an infinite solution set, and $K^\star \neq K^+$. The second example illustrates the solution to a parabolic cost, as motivated by Figure 1.2. These examples constitute additional material not found in source paper [110] for this chapter.

Example 3.7.1. Suppose we have a system composed of two uncoupled integrators with a drift in the

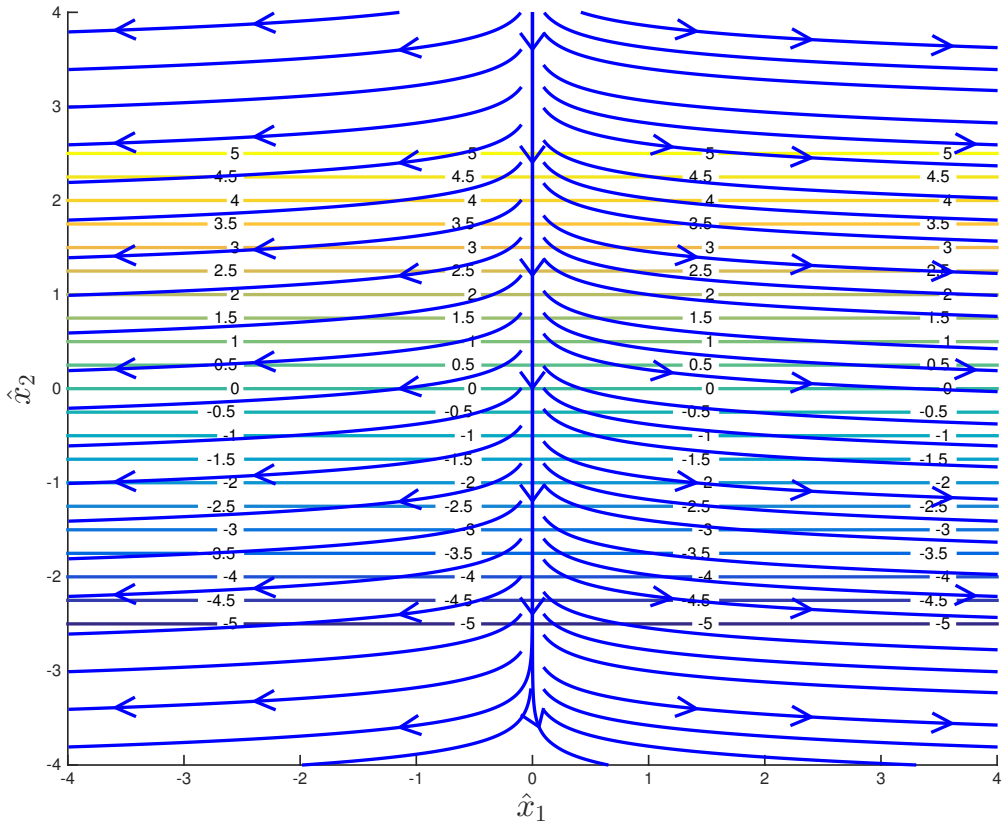

Figure 3.1: The closed-loop system under the optimal control for Example 3.7.1. Here $a_0 = 2$, $a_0' = 0$, and $\alpha = -1.5$.

state

$$\dot{\hat{x}} = \begin{bmatrix} a_0 & 0 \\ 0 & a_0' \end{bmatrix} \hat{x} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \hat{u}, \quad \hat{x}(0) = \hat{x}_0,$$

where $a_0 \geq 0$ and $a_0' \leq 0$. Consider a discounted cost functional with a linear cost in the state

$$\int_0^\infty \mathrm{e}^{2\alpha t}(2q^\top \hat{x} + q_0 + \hat{u}^\top \hat{u})\, dt,$$

where $\alpha < 0$ is the discount factor, $q = (0, 1)$ gives the direction of decreasing cost level sets, and q_0 is an offset to the cost value (to be set later to help with ensuring well-posedness since it has no effect on the optimal control). Figure 3.1 shows the level sets of the cost. Since the cost is not in the standard form of an ellipsoid centered about a desired equilibrium point, we consider the free-endpoint problem.

Following the standard transformation [7], let $x := \mathrm{e}^{\alpha t} \begin{bmatrix} \hat{x}^\top & z \end{bmatrix}^\top$ with $\dot{z} = 0$ and $x_0 = x(0) =$

$\begin{bmatrix}\hat{x}_0^\top & 1\end{bmatrix}^\top$, and $u := \mathrm{e}^{\alpha t}\hat{u}$. In the standard form (3.1) (3.3), the system and cost matrices are

$$A = \begin{bmatrix} a_0+\alpha & 0 & 0 \\ 0 & a_0'+\alpha & 0 \\ 0 & 0 & \alpha \end{bmatrix},\ B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix},\ Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & q_0 \end{bmatrix}.$$

This is already in controllable decomposition form with $n_1 = 2$ and $n_2 = 1$. Using Theorem 2.4.1, it is easy to see that Q is indefinite for any choice of q_0. To solve the indefinite optimal control problem, we need to verify well-posedness and attainability as well as find $K^* \in \partial\Gamma$, which is broken down into four steps below. We analyze the effect of a_0, a_0', α, q_0 on the solution.

1. Check that $\partial\Gamma_1 \neq \emptyset$ and compute the maximal and minimal solutions. We explicitly compute all the solutions to $\phi_1(K_1) = 0$. Writing $K_1 = \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix}$, by direct substitution into (3.17) we have that

$$\phi_1(K_1) = \begin{bmatrix} 2(a_0+\alpha)k_1 - k_1^2 - k_2^2 & k_2((a_0+a_0'+2\alpha) - k_1 - k_3) \\ k_2((a_0+a_0'+2\alpha) - k_1 - k_3) & 2(a_0'+\alpha)k_3 - k_3^2 - k_2^2 \end{bmatrix}. \tag{3.49}$$

   It is easy to verify that there are exactly four solutions

$$K_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},\ \begin{bmatrix} 2a & 0 \\ 0 & 0 \end{bmatrix},\ \begin{bmatrix} 0 & 0 \\ 0 & 2a' \end{bmatrix},\ \begin{bmatrix} 2a & 0 \\ 0 & 2a' \end{bmatrix}.$$

   where $a := a_0 + \alpha$, and $a' := a_0' + \alpha$. Since $\alpha < 0$ and $a_0' \leq 0$, $a' < 0$. On the other hand, a can be any real number depending on the choice of $a_0 \geq 0$. When $a \leq 0$, we have that $\mathcal{N}_1(\mathcal{L}_1) = 0$ by (3.21) (where $\mathcal{L}_1 = \mathbb{R}^2$ for the free-endpoint problem restricted to the controllable subspace) and so Theorem 3.6.1 would eventually give that $K^* = K^+$ (once well-posedness is verified). Thus we will consider the case $a > 0$, which gives rise to interesting pathologies. Under this assumption, it is easy to identify that

$$K_1^+ = \begin{bmatrix} 2a & 0 \\ 0 & 0 \end{bmatrix} \geq 0,\ \ K_1^- = \begin{bmatrix} 0 & 0 \\ 0 & 2a' \end{bmatrix} \leq 0$$

   and so $\partial\Gamma_{1+} \neq \emptyset$ and $\partial\Gamma_{1-} \subset \Gamma_{1-} \neq \emptyset$. Alternatively, since $Q_1 \geq 0$ if and only if $0 \in \Gamma_1$, $Q_1 = 0$ implies that $\Gamma_{1-} \neq \emptyset$.

2. Verify well-posedness. Since $\alpha < 0$, (A, B) is stabilizable. From above we already have that $\Gamma_{1-} \neq \emptyset$, so we are left to a find a solution $K \in \Gamma_- \neq \emptyset$. For notational convenience, we write the

blocks in (3.15) as

$$\phi(K) = \begin{bmatrix} \phi_1(K_1) & \phi_{12}(K_1, K_{12}) \\ \phi_{12}(K_1, K_{12})^\top & \phi_2(K_{12}, K_2) \end{bmatrix}. \tag{3.50}$$

Using Theorem 2.4.1, we require simultaneously that

(a) $K_1 \leq 0$,

(b) $(I - K_1 K_1^\dagger) K_{12} = 0$,

(c) $K_2 - K_{12}^\top K_1^\dagger K_{12} \leq 0$,

(d) $\phi_1(K_1) \geq 0$,

(e) $(I - \phi_1(K_1)\phi_1(K_1)^\dagger)\phi_{12}(K_1, K_{12}) = 0$,

(f) $\phi_2(K_{12}, K_2) - \phi_{12}(K_1, K_{12})^\top \phi_1(K_1)^\dagger \phi_{12}(K_1, K_{12}) \geq 0$

We write $K = \begin{bmatrix} k_1 & k_2 & k_4 \\ k_2 & k_3 & k_5 \\ k_4 & k_5 & k_6 \end{bmatrix}$. Using (3.49) and the expressions for K_1^+, K_1^-, we must have that $k_2 = 0$. Thus (a) and (d) are satisfied if and only if $k_1, k_3 \leq 0$ and $k_1(2a - k_1), k_3(2a' - k_3) \geq 0$. Analyzing the quadratic inequalities, we see that the simultaneous solutions are $k_1 = 0$ and $k_3 \in [2a', 0]$. Next (b) and (c) impose that $k_4 = 0$ and

$$k_6 - k_5^2 k_3^\dagger \leq 0. \tag{3.51}$$

So far we have $K = \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_3 & k_5 \\ 0 & k_5 & k_6 \end{bmatrix}$. With this information,

$$\phi(K) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_3(2a' - k_3) & k_5(a' + \alpha - k_3) + 1 \\ 0 & k_5(a' + \alpha - k_3) + 1 & 2\alpha k_6 + q_0 - k_5^2 \end{bmatrix}.$$

Define $k' = k_3(2a' - k_3)$. Finally, imposing (e) and (f) gives

$$(1 - k'(k')^\dagger)(k_5(a' + \alpha - k_3) + 1) = 0 \tag{3.52}$$

$$k_6 + q_0 - k_5^2 - (k_5(a' + \alpha - k_3) + 1)^2 (k')^\dagger \geq 0 \tag{3.53}$$

There are several cases, as we vary $k_3 \in [2a', 0]$.

For $k_3 = 0$, we immediately violate (3.52).

For $k_3 = 2a'$, we have that $k' = 0$ and $a' + \alpha - k_3 = -a'_0$. Combining the constraints (3.51), (3.52), (3.53), the following solutions belong to Γ_-:

$$K = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2a' & 1/a'_0 \\ 0 & 1/a'_0 & k_6 \end{bmatrix}, \ \forall k_6 \leq \min(1/(2a'(a'_0)^2), (1/(a'_0)^2 - q_0)/(2\alpha)).$$

It is clear that there are an infinite number of solutions in Γ_- when $a'_0 < 0$, and hence there exists no minimal solution on Γ (or $\partial\Gamma$). Also, the freedom of choice of q_0 is irrelevant, so we may set $q_0 = 0$. In the special case that $a'_0 = 0$, we do not obtain any solutions in Γ_- when $k_3 = 2a'$. Again, this illustrates that the main result on the $(\text{LQCP})_{\mathcal{L}}$ problem should be independent of the existence of the minimal solution on Γ in the case when (A, B) is stabilizable.

For $k_3 \in (2a', 0)$, combining the constraints (3.51), (3.52), (3.53), we again obtain many solutions $K \in \Gamma_-$. This time, it is difficult to explicitly parameterize all the solutions. Some sample solutions, with $k_3 \in (2a', 0)$, $q_0 = 0$, $k_5 = 0$, have the form

$$K = \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_3 & 0 \\ 0 & 0 & k_6 \end{bmatrix}, \ \forall k_6 \leq 1/(2\alpha k').$$

With all the above, it is clear that we always have an infinite number of solutions in Γ_-, regardless of the choice of $a'_0 \leq 0$ and q_0 (still assuming that $\alpha < 0$, $a_0 \geq 0$ and $a > 0$). Notice that the presence of α in the denominators of the above expressions justifies the use of discounting ($\alpha < 0$). In summary, our problem is well-posed.

3. Verify attainability. It is easy to see that $\Delta_1 = K_1^+ - K_1^- > 0$. Thus $\text{Ker}(\Delta_1) = 0$ and Theorem 3.5.8 immediately gives that the problem is attainable.

4. Determine K^*. Assuming $a > 0$, it is easy to identify that

$$K_1^* = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \geq 0.$$

We used the well-known fact that K_1^* is the smallest solution in $\partial\Gamma_{1+}$ since $Q_1 = 0 \geq 0$. However, it is also easy to compute that $\mathcal{N}_1(\mathcal{L}_1) = \text{Im}(\begin{bmatrix} 1 & 0 \end{bmatrix}^\top)$ using (3.21) so that K_1^* is obtained using

(3.22) by applying Theorem 3.5.3. Thus $n_{1,1} = 1$, $n_{1,2} = 0$, and $n_{1,3} = 1$. The remaining blocks are determined by Theorem 3.5.6. The solution is

$$K^* = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1/(a'+\alpha) \\ 0 & -1/(a'+\alpha) & 1/(2\alpha(a'+\alpha)^2) \end{bmatrix}.$$

To illustrate the main ambiguity in determining K^* before Theorem 3.5.6 was discovered (ie. the fact that always $K^*_{12,1} = 0$), observe that $\phi_{12}(K^*_1, K^*_{12}) = 0$ is

$$\begin{bmatrix} K^*_{12,1}(a+\alpha) \\ K^*_{12,3}(a'+\alpha)+1 \end{bmatrix} = 0.$$

The second equation always has a solution for $K^*_{12,3}$ since $a'+\alpha < 0$. However for the first equation, the pathological choice of $a + \alpha = 0$ gives an infinite solution set for $K^*_{12,1}$.

For completeness, the maximal solution (which always exists by stabilizability) is

$$K^+ = \begin{bmatrix} 2a & 0 & 0 \\ 0 & 0 & -1/(a'+\alpha) \\ 0 & -1/(a'+\alpha) & 1/(2\alpha(a'+\alpha)^2) \end{bmatrix}.$$

which is different from K^*. In many cases, Theorem 3.6.1 applies to give $K^+ = K^*$, once again exhibiting the pathological nature of this example.

To conclude, the optimal control is $u = -B^\top K^* x$. In the original coordinates, the optimal control is $\hat{u} = \begin{bmatrix} 0 & 1/(a'+\alpha) \end{bmatrix}^\top$. The solution $K^*_1 = 0$ means that there is no feedback term on the state $\hat{x}$, and the solution K^*_2 has no effect on the control (explaining why it was possible to choose q_0 freely, although we set it to zero). A phase plot showing the cost level sets and closed-loop system vector field is shown Figure 3.1. The closed-loop dynamics are

$$\dot{\hat{x}} = \begin{bmatrix} a_0 & 0 \\ 0 & a_0' \end{bmatrix} \hat{x} + \begin{bmatrix} 0 \\ 1/(a'+\alpha) \end{bmatrix}$$

The solutions flow away in the $\hat{x}_1$ axis since the dynamics are unstable (since this state is not observed in the cost), and down the cost level sets in the $\hat{x}_2$ direction. ◁

Example 3.7.2. Suppose we have a system composed of two uncoupled integrators $\dot{\hat{x}} = \hat{u}$, with $\hat{x}, \hat{u} \in \mathbb{R}^2$

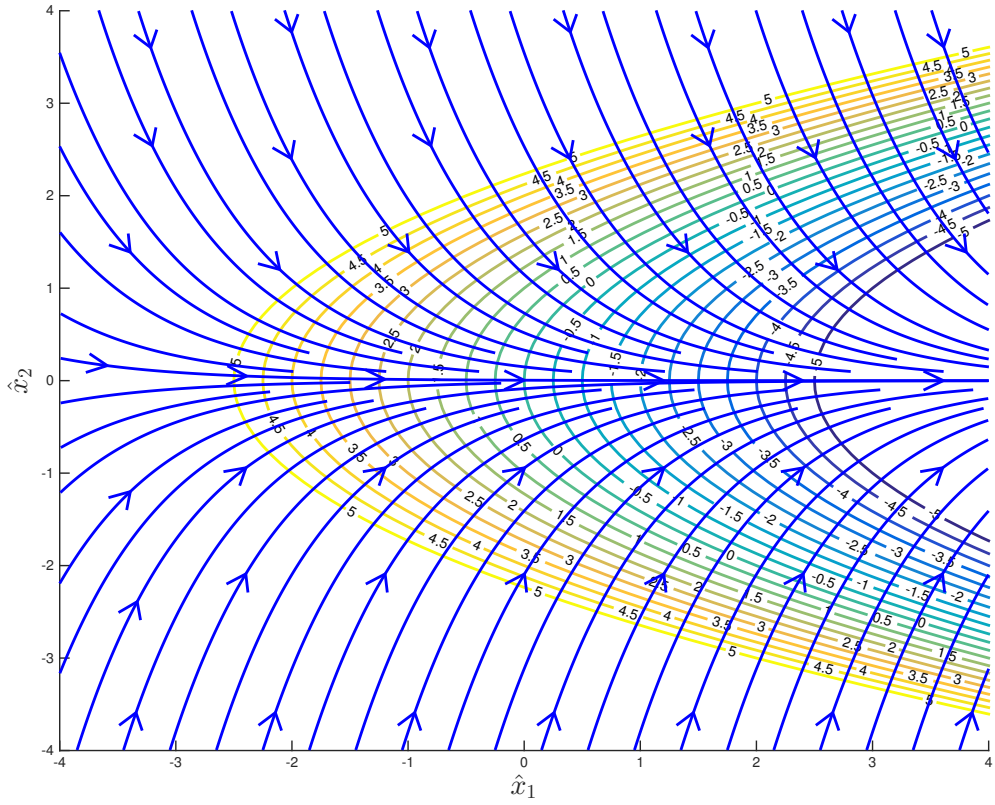

Figure 3.2: The closed-loop system under the optimal control for the single integrators with parabolic cost, for Example 3.7.2. Here $\alpha = -1$.

and $\hat{x}(0) = \hat{x}_0$. Consider a discounted cost functional with parabolic cost level sets in the state

$$\int_0^\infty \mathrm{e}^{2\alpha t}(\hat{x}^\top \hat{Q}\hat{x} + 2q^\top \hat{x} + \hat{u}^\top \hat{u})\, dt,$$

where $\alpha < 0$ is the discount factor, $\hat{Q} = \mathrm{diag}(0, 1)$, and $q = (-1, 0)$. Notice that since $q \notin \mathrm{Im}(\hat{Q})$ the cost level sets are of the (degenerate) parabolic type and are unbounded below. Once again we consider the free-endpoint problem.

As before, let $x := \mathrm{e}^{\alpha t}\begin{bmatrix}\hat{x}^\top & z\end{bmatrix}^\top$ with $\dot{z} = 0$ and $x_0 = x(0) = \begin{bmatrix}\hat{x}_0^\top & 1\end{bmatrix}^\top$, and $u := \mathrm{e}^{\alpha t}\hat{u}$. In the standard form (3.1) (3.3), the system and cost matrices are

$$A = \alpha I_3,\ B = \begin{bmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{bmatrix},\ Q = \begin{bmatrix}0 & 0 & -1\\ 0 & 1 & 0\\ -1 & 0 & 0\end{bmatrix}.$$

This is already in controllable decomposition form with $n_1 = 2$ and $n_2 = 1$. Using Theorem 2.4.1, it is easy to see that Q is indefinite. To solve the indefinite optimal control problem, we need to verify well-posedness and attainability as well as find $K^* \in \partial\Gamma$. We analyze the effect of α on the solution.

1. Check that $\partial\Gamma_1 \neq \emptyset$ and compute the maximal and minimal solutions. We explicitly compute all the solutions to $\phi_1(K_1) = 0$. Writing $K_1 = \begin{bmatrix} k_1 & k_2 \\ k_2 & k_3 \end{bmatrix}$, by direct substitution into (3.17) we have that
$$\phi_1(K_1) = \begin{bmatrix} 2\alpha k_1 - k_1^2 - k_2^2 & k_2(2\alpha - k_1 - k_3) \\ k_2(2\alpha - k_1 - k_3) & 2\alpha k_3 - k_3^2 - k_2^2 + 1 \end{bmatrix}.$$
It is easy to verify that the four solutions are generated with $k_1 = 0, 2\alpha$, $k_2 = 0$, and $k_3 = \alpha \pm \sqrt{\alpha^2+1}$. Thus
$$K_1^+ = \begin{bmatrix} 0 & 0 \\ 0 & \alpha + \sqrt{\alpha^2+1} \end{bmatrix} \geq 0, \quad K_1^- = \begin{bmatrix} 2\alpha & 0 \\ 0 & \alpha - \sqrt{\alpha^2+1} \end{bmatrix} < 0$$
and so $\partial\Gamma_{1+} \neq \emptyset$ and $\partial\Gamma_{1-} \subset \Gamma_{1-} \neq \emptyset$.

2. Verify well-posedness. Since $\alpha < 0$, (A, B) is stabilizable. From above we already have that $\Gamma_{1-} \neq \emptyset$, so we are left to a find a solution $K \in \Gamma_- \neq \emptyset$. We follow the same procedure as in the previous example. It can be verified that no solution to $\phi(K) = 0$ exists with $K_1 = K_1^-$. Working through the conditions (a) - (f), it can be verified that at least the following solutions belong to Γ_-
$$K = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_3 & 0 \\ 0 & 0 & 1/(2\alpha k_1(2\alpha - k_1)) \end{bmatrix}, \; k_1 \in (2\alpha, 0), \; k_3 \in (\alpha - \sqrt{\alpha^2+1}, 0).$$
As one would intuitively expect, we need discounting ($\alpha < 0$) in order to have well-posedness.

3. Verify attainability. It is easy to see that $\Delta_1 = K_1^+ - K_1^- > 0$. Thus $\mathrm{Ker}(\Delta_1) = 0$ and Theorem 3.5.8 immediately gives that the problem is attainable.

4. Determine K^*. Since $\mathcal{X}(A_1) \subset \mathbb{C}^-$ due to discounting, we have that $\mathcal{N}_1(\mathcal{L}_1) = 0$. Then Theorem 3.6.1 gives that $K^\star = K^+$. It can be easily computed that $K_{12}^+ = (1/(2\alpha), 0)$ and $K_2^+ = 1/(8\alpha^3)$.

To conclude, the optimal control is $u = -B^\top K^* x$. In the original coordinates, the optimal control

is $\hat{u} = -K_1^+ \hat{x} - K_{12}^+$. The closed-loop dynamics are

$$\dot{\hat{x}} = -\begin{bmatrix} 1/(2\alpha) \\ (\alpha + \sqrt{\alpha^2 + 1})\hat{x}_2 \end{bmatrix}$$

A phase plot showing the cost level sets and closed-loop system vector field is shown Figure 3.2. As expected, trajectories flow down the parabolic level sets. Surprisingly, without the indefinite LQ theory for stabilizable dynamics, this simple example's intuitive solution would have remained unanswered. ◁

3.8 Conclusion

In this chapter we addressed a problem in the area of linear quadratic optimal control which has been open for the last 20 years. Specifically, we consider the regular, infinite-horizon, stability-modulo-a-subspace, indefinite LQ problem when the dynamics are stabilizable. Previous works have also addressed this problem, but under the restrictive assumption that the dynamics are controllable. The generalization from controllable to stabilizable dynamics is significant in that there is a lack of structure in the solutions of the algebraic Riccati equation in the stabilizable case. Consequently the connection between the ARE solution set and the LQ problem under consideration has remained elusive. We resolved this gap by combining a suitable sufficient condition for a finite optimal cost with a specific decomposition to unambiguously deduce the correct form of the optimal cost and control. The determination of necessary and sufficient conditions for a finite value function in the regular, infinite-horizon, stability-modulo-a-subspace, indefinite LQ problem is still open.

Our study of this indefinite LQ control problem was motivated by the difficulties encountered in exploring high dimensional state spaces with safety constraints in the context of hybrid control. However, this line of research was not pursued further due to following main reasons:

- *No guarantee on safety.* Although the LQ approach purposely relaxed hard safety constraints into soft ones, this is generally not suitable for safety-critical applications such as motion planning in the presence of many vehicles.

- *No guarantee on progress.* For a hybrid control scheme to be successful, the current feedback controller corresponding to some cost should eventually lead to the domain of operation of another feedback controller with some other cost. The optimal controller corresponding to an indefinite cost does not guarantee any properties on the direction that solutions flow, which may differ depending on initial conditions.

- *No guarantee on control continuity.* When switching from one feedback controller to another, there may be large discontinuities in the control. These discontinuities are undesirable for applications, particularity those involving quadrocopters.

While indefinite costs that encourage trajectories to flow in a particular direction seem promising, ultimately the formulation of an optimal control problem with no explicit safety constraints is not sufficient, even under the consideration of the stability-modulo-a-subspace constraint.

To remedy the first issue on lack of safety, some investigation was done on cone-constrained systems, which makes use of nonnegative matrices [12]. The idea was to formulate an (indefinite) LQ problem with a cone constraint to encourage trajectories to flow within the cone, but away from the cone apex rather than towards it. The established main result on the indefinite LQ theory is an essential starting point by providing a lower bound on the optimal cost when no additional cone-constraint is enforced. However, most optimization problems with constraints nearly always do not have a nice analytic solution, although there may be special cases worth investigating.

The two other issues are also not easily addressed within the realm of analytical optimal control. There is some literature on constrained optimization problems [23, 96, 50, 40], but none have considered state space constraints with an indefinite cost. Likewise, model predictive control [88] is a popular numerical technique for addressing optimal control problems with constraints, but to our knowledge has only been considered for positive definite cost functionals. Also it would be interesting to explore the relationship between the indefinite costs we have considered here and control barrier functions [6].

Chapter 4

A Modular Framework for Motion Planning

4.1 Introduction

This chapter presents a modular framework for motion planning and control of robotic systems. While motion planning has received a great deal of attention by many researchers, because the problem is highly complex especially when there are several robotic agents working together in a cluttered environment, significant challenges remain. Modularity is an architectural strategy to overcome this complexity. All frameworks for motion planning should aim to balance flexibility in the control specification at the high level, guarantees on correctness and safety at the low level, and computational feasibility overall.

Historically motion planning was focused on high level planning algorithms, while suppressing details on the dynamic capabilities of the robots at the low level [65]. Taking full account of low level dynamics in combination with solving the high level planning problem can lead to a computationally intractable problem. Despite the wealth of available research [25, 93, 11, 58], computationally efficient solutions to the motion planning problem with tight integration of high and low levels are highly sought after.

We propose a modular framework so that one can independently plug and play both low level controllers and high level planning algorithms in order to realize a balance between flexibility at the high level, safety at the low level, and computational feasibility. To make a customizable approach feasible, we introduce three assumptions. First, the output space of the underlying dynamical system has translational symmetry, namely position invariance, a property satisfied by many robotic models [33]. Second, the output space is gridded uniformly into rectangular boxes. Finally, the control capabilities

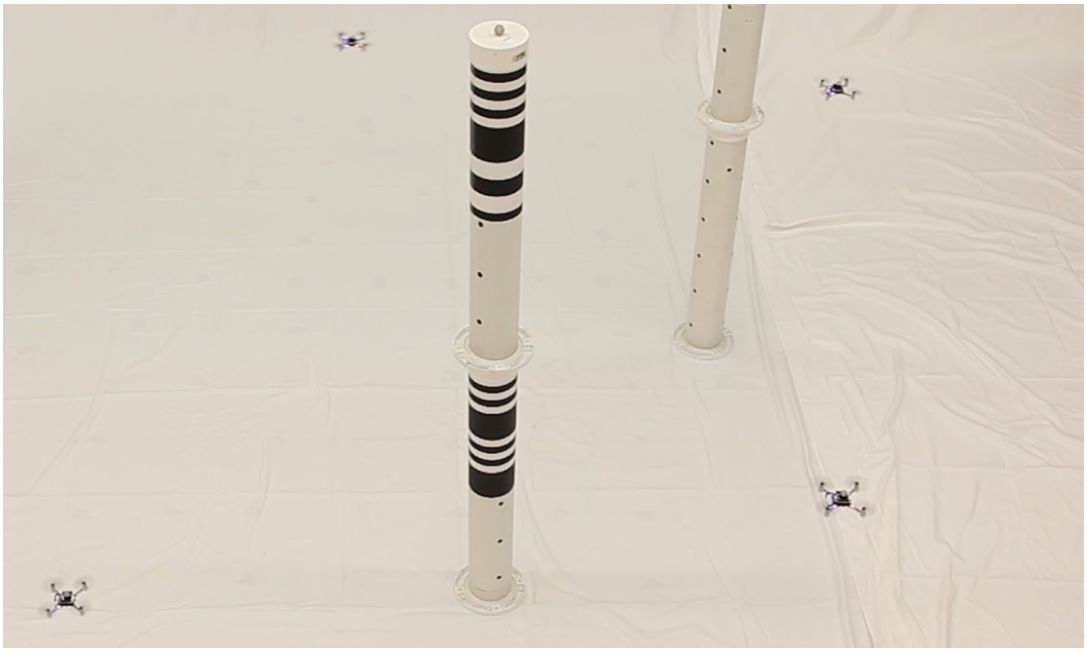

Figure 4.1: Crazyflie quadrocopters navigate a cluttered environment. Video results are available at http://tiny.cc/modular-3alg

.

are discretized into a finite set of *motion primitives*, where the low level describes the implementation of the motion primitives while the high level selects the motion primitives. Together, these assumptions imply that motion primitives can be designed over a single box, so that they can then be reapplied to any other box.

Now we give an overview of the features and techniques we employ, and we highlight other frameworks that share those features. We provide a general formulation of motion primitives for nonlinear systems so that they can be applied to multi-agent systems but potentially also to other robotic systems. We focus on reach-avoid specifications in a priori known environments, in which the system must reach a desired configuration in a safe manner [11, 25, 52, 68]. Reach-avoid offers a fairly rich behavior set so that, for instance, a fragment of linear temporal logic (LTL) can be encoded as a sequence of reach-avoid problems [116], as we also show in our applications.

As we have mentioned, we abstract the output space into rectangular regions [93] rather than more general polytopic regions [25, 32, 58, 68] in order to exploit symmetry. Motion primitives have been employed in various ways [44, 93, 52] and we encode feasible sequences of motion primitives by a *maneuver automaton* [33]. In our implementation, the low-level control design of motion primitives is based on *reach control theory*, which provides a highly flexible and intuitive set of design tools that have two notable advantages over tracking: first, it is not necessary to find feasible open-loop trajectories to track; second, safety constraints on the system states during the execution and concatenation of motion

primitives can be guaranteed by design. Finally, planning at the high level is based on standard *shortest path algorithms* [65, 17] applied to the graph arising from the synchronous product of the discrete part of the maneuver automaton and the graph arising from the output space partition. The high-level plan generates a *control policy*, which selects the motion primitives over the gridded output space. The modularity of our approach enables one to employ other closed-loop methods such as potential methods [25] or (untimed) path following [122] for low-level control design, and standard or customized graph search algorithms to generate a high-level plan.

There are three main contributions of this work. First, we provide the complete theoretical details on the requirements for the low-level control design and high-level plan, and show that these two levels operate consistency to solve the reach-avoid problem. Second, we formulate the parallel composition of maneuver automata, which is a method of composing motion primitives for individual subsystems. Since the design of motion primitives can be challenging for high order systems and since many systems have a decoupled structure, such as in the case of multiple vehicles, parallel composition provides a way to describe the combined motion capabilities of the system. Finally, the modularity and effectiveness of our framework is experimentally validated on a collection of quadrocopters in several illustrative scenarios. In particular, we feature a versatile design of motion primitives based on double integrators and we show how the customizability of the high level plan generation can be used to easily trade-off solution quality with computational efficiency. This chapter is based on [111], which is an extension of our previous work [108, 109].

The chapter is organized as follows. In the next section we highlight our contributions relative to the literature. In Section 4.3 we present a formal problem statement. The modular framework is introduced in Section 4.4. We define the output transition system, the maneuver automaton, the product automaton, and the high level plan, each which contribute to realizing a solution of the motion planning problem. In Section 4.5, we prove that our overall methodology solves the motion planning problem. In Section 4.6 we give the procedure for composing motion primitives. In Section 4.7 we present some specific motion primitives. In Section 4.8 we consider several methods to generate high level plans, which are experimentally demonstrated on quadrocopters. We summarize the chapter in Section 4.9.

4.2 Related Literature

The literature on motion planning is vast and encompasses many research communities. As such, we have categorized some common approaches and discussed how they relate to our method.

4.2.1 Graph Search and Trajectory Planning

Motion planning has often been addressed as a discrete planning problem, for which many standard graph search algorithms exist [65]. Recent work on the multi-agent reach-avoid problem has developed novel algorithms in the context of applications such as manufacturing and warehouse automation, aiming to balance computational efficiency with solution quality. For example, a centralized approach is given in [120], discretizing the workspace into a lattice and using integer linear programming to minimize the total time for robots to traverse in high densities. In [28], a sampling-based roadmap is constructed in the joint robot space using individual robot roadmaps, which is shown to be asymptotically optimal. Prioritized planning enables to safely coordinate many vehicles and is considered in a centralized and decentralized fashion in [21]. Subdimensional expansion computes mainly decentrally, but coordinates in the joint search space when agents are neighboring [113]. While such approaches typically provide various theoretical guarantees on the proposed algorithms, dynamical models and application on real robotic systems is often not considered.

The modularity of our framework is complementary, as it potentially enables existing multi-agent literature on gridded workspaces to be used directly or adapted for the generation of a high-level plan when used in conjunction with our proposed formulation of motion primitives. However, the consideration of continuous time dynamics may complicate the application of discrete planning methods in two ways. First, we must contend with constraints on successive motion primitives so that the continuous time behavior is acceptable - for example, avoiding abrupt changes in velocity. Second, we must contend with non-deterministic transitions to neighboring boxes, because motion primitives may allow more than one next box to be reached [59] - for example, modeling the joint asynchronous motion capabilities of a multi-robot system.

Trajectory tracking methods have also been applied to the formation change problem on real vehicles with complex dynamics. A sequential convex programming approach is given in [10], which computes discretized, non-colliding positional trajectories for a modest number of quadrocopters. More recently, an impressive number of quadrocopters were coordinated in [85], by first computing a sequence of grid-based waypoints and then refining it into smoother piecewise polynomials. However, since these open-loop trajectories are computed offline, deviations from the computed trajectories could result in crashes. On the other hand, our approach is more robust as it is completely untimed, carefully monitoring the progress of vehicles over the grid in a reactive way based on the measured box transitions.

4.2.2 Formal Methods

A growing body of research has explored the use of formal methods in motion planning. This chapter has been particularly inspired by [58], which provides a general framework for solving control problems for affine systems with LTL specifications. Their approach involves constructing a transition system over a polyhedral partition of the state space that arises from linear inequality constraints that constitute the atomic propositions of the LTL specification. Transitions between states of the transition system can occur if there exists an affine or piecewise affine feedback steering all continuous time trajectories from one polyhedral region to a contiguous one. Similar works to [58] include [25, 47, 68], which consider the simpler reach-avoid problem. Single and multi-robot applications followed shortly after in [32] and [11] respectively.

The appeal of these approaches is derived from their generality and faithful account of the low level system capabilities. On the downside, these methods generally do not scale well to larger state space dimensions, and so they would have limited applicability to large multi-robot systems. Our approach specializes these ideas by exploiting symmetry in the system dynamics and grid partition in order to strike a better balance between generality and computational efficiency. In particular, our feedback controllers are given as motion primitives, which can be designed independently of the obstacle and goal locations.

More recent works have also built on these formal method approaches, investigating more complex and realistic multi-robot problems. For example, service requests by multiple car robots in a city-like environment with communication constraints was considered in [22]. A cooperative task for ground vehicles was addressed in a distributed manner, enabling knowledge sharing amongst neighbors and reconfiguration of the motion plan in real time [45]. Tasks such as picking up objects are considered in conjunction with motion requirements in [87]. Since these works consider only fairly simple vehicle dynamics, they place greater emphasis on the synthesis of discrete plans satisfying the task specification. On the other hand, this chapter considers the simpler reach-avoid problem in order to develop a formulation of motion primitives for nonlinear systems with symmetries.

4.2.3 Motion Primitives

The usage of motion primitives has become popular recently in robotics, as they serve to simplify the motion planning problem by using predefined executable motion segments. Many variations exist, which have designed motion primitives using timed reference trajectories to control a formation of quadrocopters [93], paths on a state space lattice for a mobile robots [76, 24], and funnels in the state

space centered about a reference trajectory for a car [52] and a small airplane [71].

We have been inspired by ideas in [33], from which we borrowed the term "maneuver automaton". They define a motion primitive either as an equivalence class of trajectories or a timed maneuver between two classes, whereas we define a motion primitive as a feedback controller over a polyhedral region in the state space. In our formulation, concatenations between motion primitives are possible only across contiguous boxes in the output space, which provides a strict safety guarantee during concatenation. Moreover, this enables our approach to simplify obstacle avoidance to a discrete planning problem over safe boxes as in [24], bypassing the need to concatenate motion primitive trajectories using numerical optimization techniques as in [33].

Our presentation of the maneuver automaton gives explicit constraints on the design of motion primitives so that they can used reliably for high level planning. We have also introduced the notion of parallel composition of maneuver automata to build motion primitives for multi-robot systems. While our construction resembles existing methods of parallel composition [119, 103], we additionally prove that our construction preserves desired properties that enable consistency between the low and high levels. To the authors' best knowledge, this chapter is the first rigorous treatment of feedback-based motion primitives defined on a uniformly gridded output space.

4.3 Problem Statement

Consider the general nonlinear control system

$$\dot{x} = f(x, u), \qquad y = h(x), \tag{4.1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^\mu$ is the input, and $y \in \mathbb{R}^p$ is the output. Let $\phi(\cdot, x_0)$ and $y(\cdot, x_0)$ denote the state and output trajectories of (4.1) starting at initial condition $x_0 \in \mathbb{R}^n$ and under some open-loop or feedback control.

Let $\mathcal{P} \subset \mathbb{R}^p$ be a feasible set in the output space and let $\mathcal{G} \subset \mathcal{P}$ be a goal set. In multi-vehicle motion planning contexts, $\mathcal{P}$ represents the feasible joint output configurations of the system, which can arise from specifications involving obstacle avoidance, collision avoidance, communication constraints, and others. We consider the following problem.

Problem 4.3.1 (Reach-Avoid)**.** *We are given the system* (4.1)*, a non-empty feasible set* $\mathcal{P} \subset \mathbb{R}^p$ *and a non-empty goal set* $\mathcal{G} \subset \mathcal{P}$*. Find a feedback control* $u(x)$ *and a set of initial conditions* $\mathcal{X}_0 \subset \mathbb{R}^n$ *such that for each* $x_0 \in \mathcal{X}_0$ *we have*

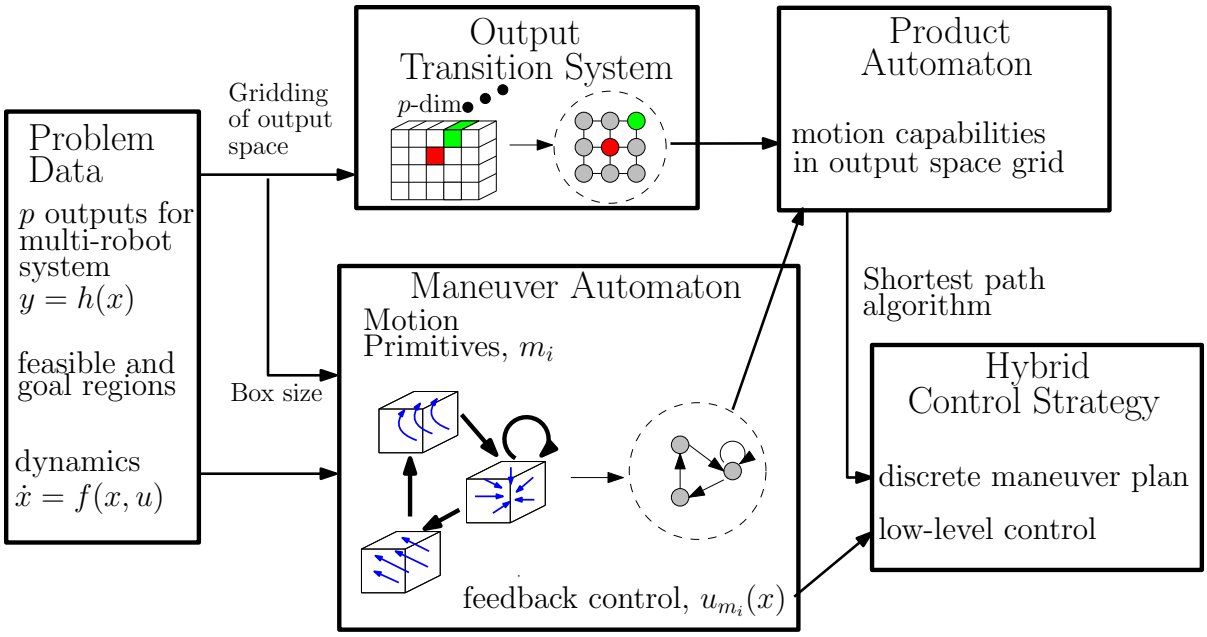

Figure 4.2: Our modular framework consists of five modules.

(i) **Avoid***: $y(t, x_0) \notin \mathbb{R}^p \setminus \mathcal{P}$ for all $t \geq 0$,*

(ii) **Reach***: there exists $T \geq 0$ such that for all $t \geq T$, $y(t, x_0) \in \mathcal{G}$.*

We make an assumption regarding the outputs of the system (4.1) in order to exploit symmetry; see [33] for an exposition on nonlinear control systems with symmetries.

Assumption 4.3.2. *First, we assume that there is an injective map $o : \{1, \ldots, p\} \to \{1, \ldots, n\}$ associating each output to a distinct state, so that $h(x) = (x_{o(1)}, \ldots, x_{o(p)})$. We define the (injective) insertion map $h_o^{-1} : \mathbb{R}^p \to \mathbb{R}^n$ as $h_o^{-1}(y) = x$, which satisfies $h(x) = y$ and $x_i = 0$ for all $i \in \{1, \ldots, n\} \setminus \{o(1), \ldots, o(p)\}$). Second, we assume that the system has a* translational invariance *with respect to its outputs. That is, for all $x \in \mathbb{R}^n$, $u \in \mathbb{R}^\mu$ and $y \in \mathbb{R}^p$, we have $f(x, u) = f(x + h_o^{-1}(y), u)$.*

The assumption that the outputs of the system are a subset of the states is used in our framework to be able to design feedback controllers in the full state space that achieve desirable behavior in the output space. The second statement says that the vector field is invariant to the value of the output. In the literature this condition is called a *symmetry of the system* or *translational invariance*. This assumption is satisfied for many robotic systems, for example, when the outputs are positions. Also, we will see in Section 4.7.2 that it significantly simplifies our control design.

4.4 Modular Framework

In this section we present our methodology to solve the motion planning problem in the form of an architecture that breaks down Problem 4.3.1. This architecture consists of five main modules, as depicted in Figure 4.2.

- The *Problem Data* include the system (4.1) with p outputs satisfying Assumption 4.3.2 and a reach-avoid task to be executed in the output space.

- The *Output Transition System (OTS) is a directed graph whose nodes (called locations) represent p-dimensional boxes on a gridded output space and whose edges describe which boxes in the output space are contiguous.*

- The *Maneuver Automaton* (MA) is a hybrid system whose modes correspond to so-called motion primitives. Each motion primitive is associated with a closed-loop vector field by applying a feedback law to (4.1). The edges of the MA represent feasible successive motion primitives. Each motion primitive generates some desired behavior of the output trajectories of the closed-loop system over a box in the output space. Because of the uniform gridding of the output space into boxes and because of the symmetry in the outputs described in Assumption 4.3.2, motion primitives can be designed over only one canonical box Y^*.

- The *Product Automaton* (PA) is a graph which is the synchronous product of the OTS and the discrete part of the MA. It represents the combined constraints on feasible motions in the output space and feasible successive motion primitives.

- The *Hybrid Control Strategy* is a combination of low level controllers obtained from the design of motion primitives, and a high level plan on the product automaton.

Next we describe in greater detail the OTS, MA, and PA.

4.4.1 Output Transition System

The OTS provides an abstract description of the *workspace* or *output space* associated with the system (4.1). It serves to capture the feasible motions of output trajectories of the system (4.1) in a gridded output space, as in Figure 4.3. Specifically, we partition the output space into p-dimensional boxes with lengths $d = (d_1, \ldots, d_p)$, where $d_i > 0$ is the length of i-th edge. We use a finite number of boxes to under-approximate the feasible set $\mathcal{P}$. Enumerating the boxes as $\{Y_1, \ldots, Y_{n_L}\}$, the j-th box can be

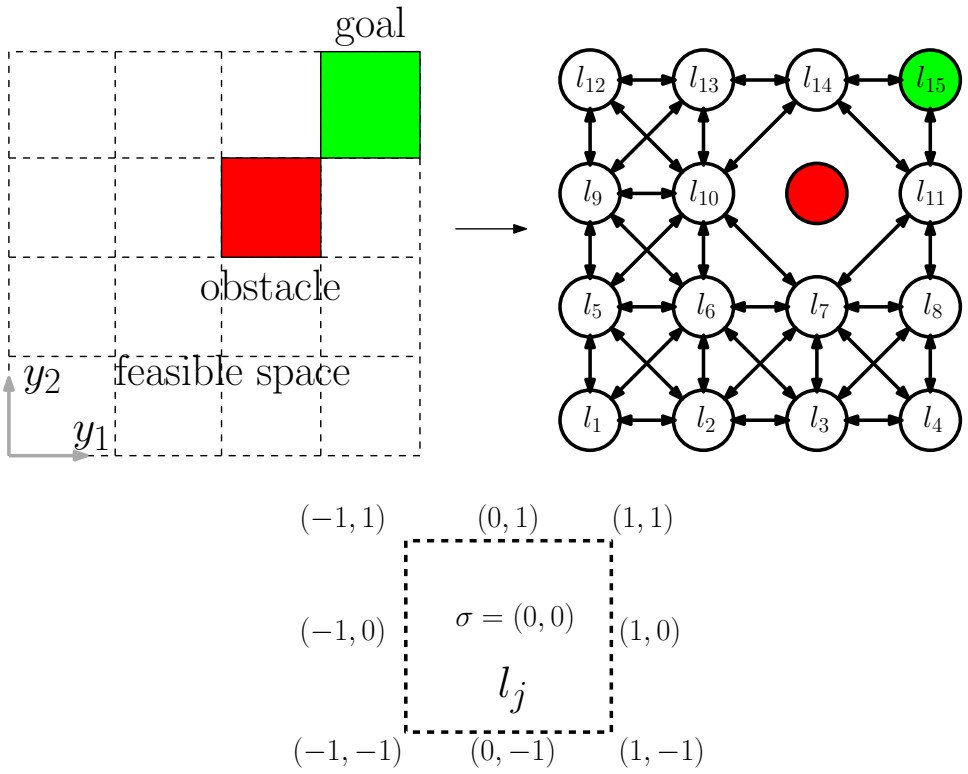

Figure 4.3: A two output ($p = 2$) example of a reach-avoid task. Shown on the left is the feasible space $\mathcal{P}$ consisting of 15 non-obstacle boxes (not red) and the goal region $\mathcal{G}$ (green). The output transition system (OTS), which abstracts the box regions and their neighbour connectivity, is shown on the right. Shown below, the possible offsets towards a neighbouring box are labelled using $\Sigma = \{-1, 0, 1\}^2$.

expressed in the form

$$Y_j := \prod_{i=1}^{p} [\eta_{ji} d_i, (\eta_{ji} + 1) d_i] \,, \tag{4.2}$$

where $\eta_{ji} \in \mathbb{Z}$, $i = 1, \ldots, p$ are constants. Thus we have that

$$\bigcup_{j=1}^{n_L} Y_j \subset \mathcal{P} \,.$$

Among these boxes, we assume there is a non-empty set of indices $I_g \subset \{1, \ldots, n_L\}$, so that we may similarly under-approximate the goal region as

$$\bigcup_{j \in I_g} Y_j \subset \mathcal{G} \subset \mathcal{P} \,.$$

We define a canonical p-dimensional box with edge lengths $d_i > 0$ given by

$$Y^* = \prod_{i=1}^{p} [0, d_i] \,.$$

Referring to (4.2), we can see that each box Y_j, $j = 1, \ldots, n_L$ is a translation of Y^* by an amount $\eta_{ji} d_i$ along the i-th axis.

Definition 4.4.1. *Given the lengths d and a non-empty goal index set I_g, an* output transition system *(OTS) is a tuple $\mathcal{A}_{\text{OTS}} = (L_{\text{OTS}}, \Sigma, E_{\text{OTS}}, L^g_{\text{OTS}})$ with the following components:*

State Space $L_{\text{OTS}} := \{l_1, \ldots, l_{n_L}\} \subset \mathbb{Z}^p$ *is a finite set of nodes called* locations. *Each location $l_j \in L_{\text{OTS}}$ is associated with a safe box $Y_j \subset \mathcal{P}$ in the output space and hence we write $l_j = (\eta_{j1}, \ldots, \eta_{jp})$.*

Labels $\Sigma := \{-1, 0, 1\}^p \subset \mathbb{Z}^p$ *is a finite set of* labels. *A label $\sigma \in \Sigma$ is used to identify the offset between neighbouring boxes.*

Edges $E_{\text{OTS}} \subset L_{\text{OTS}} \times \Sigma \times L_{\text{OTS}}$ *is a set of* directed edges *where $(l_j, \sigma, l_{j'}) \in E_{\text{OTS}}$ if $j \neq j'$, $Y_j \cap Y_{j'} \neq \emptyset$, and $\sigma = l_{j'} - l_j \in \Sigma$. Thus, for each $i = 1, \ldots, p$, the neighbouring box $l_{j'}$ is either one box to the left ($\sigma_i = -1$), the same box ($\sigma_i = 0$), or one box to the right ($\sigma_i = 1$). In this manner σ records the offset between contiguous boxes.*

Final Condition $L^g_{\text{OTS}} = \{l_j \in L_{\text{OTS}} \mid j \in I_g\}$ *denotes the set of locations associated with goal boxes.*

Remark 4.4.2. *We observe that the OTS is deterministic. That is, for a given $l \in L_{\text{OTS}}$ and $\sigma \in \Sigma$, there is at most one $l' \in L_{\text{OTS}}$ such that $(l, \sigma, l') \in E_{\text{OTS}}$. This follows immediately from the fact that $\sigma = l' - l$ records the offset between the neighbouring boxes.*

Figure 4.3 shows a sample OTS for a simple scenario. The OTS locations are associated with 15 feasible boxes, including a goal box for the reach-avoid task. The OTS edges are shown as bidirectional arrows; for example, interpreting $l_1 = (0, 0)$ and $l_6 = (1, 1)$ on the grid, then $e = (l_6, (-1, -1), l_1) \in E_{\text{OTS}}$.

4.4.2 Maneuver Automaton

The *maneuver automaton* (MA) is a hybrid system consisting of a finite automaton and continuous time dynamics in each discrete state. The discrete states of the finite automaton correspond to *motion primitives*, while transitions between discrete states correspond to the allowable transitions between motion primitives. The continuous time dynamics are given by closed-loop vector fields (4.1) with a control law designed based on reach control theory (any other feedback control design method can be used).

Before presenting the MA, we first explain how this module is used in the overall framework. To solve Problem 4.3.1, we assign motion primitives to the boxes Y_j of the partitioned output space such that obstacle regions are avoided and the goal region is eventually reached. The discrete part of the MA encodes the constraints on successive motion primitives. Such constraints might arise from a non-chattering requirement, continuity requirement, or requirement on correct switching between regions of the state space. A dynamic programming algorithm for assignment of motion primitives on boxes is addressed in Section 4.4.4; the salient point about this algorithm at this stage is that it only uses the discrete part of the MA.

In contrast, the continuous time part of the MA is used both for simulation of the closed-loop dynamics to verify that the motion primitives are well designed, as well as for the implementation of the low level feedback in real-time. The motion primitives are defined only on the canonical box Y^* to simplify the design. This simplification is possible because of the translational symmetry provided by Assumption 4.3.2 and the fact that each box Y_j is a translation of Y^*. In simulation, a given motion primitive can cause output trajectories to reach certain faces of Y^*. If a face is reached, the output trajectory is interpreted as being reset to the opposite face and another motion primitive is selected to be implemented over Y^* (of course, the real experimental output trajectories do not undergo resets but move continuously from box to box in the output space). The selection of the next motion primitive is constrained by a combination of the previous motion primitive and the face of Y^* that is reached. The discrete transitions in the MA encode these constraints.

Definition 4.4.3. *Consider the system* (4.1) *satisfying Assumption 4.3.2 and the box Y^* with lengths d. The* maneuver automaton *(MA) is a tuple $\mathcal{H}_{\text{MA}} = (Q_{\text{MA}}, \Sigma, E_{\text{MA}}, X_{\text{MA}}, I_{\text{MA}}, G_{\text{MA}}, R_{\text{MA}}, Q^0_{\text{MA}})$, where*

State Space $Q_{\text{MA}} = M \times \mathbb{R}^n$ *is the hybrid state space, where* $M = \{m_1, \ldots, m_{n_M}\}$ *is a finite set of nodes, each corresponding to a motion primitive.*

Labels Σ*, the same labels used in the OTS.*

Edges $E_{\text{MA}} \subset M \times \Sigma \times M$ *is a finite set of edges.*

Vector Fields $X_{\text{MA}} : M \to \mathcal{X}(\mathbb{R}^n)$ *is a function assigning a globally Lipschitz closed-loop vector field to each motion primitive* $m \in M$*. That is, for each* $m \in M$*, we have* $X_{\text{MA}}(m) = f(\cdot, u_m(\cdot))$ *where* $u_m(\cdot)$ *is a feedback controller associated with* $m \in M$*.*

Invariants $I_{\text{MA}} : M \to \mathcal{P}(\mathbb{R}^n)$ *assigns a bounded invariant set* $I_{\text{MA}}(m)$ *to each* $m \in M$*. We impose that* $I_{\text{MA}}(m) \subset h^{-1}(Y^*)$*. The set* $I_{\text{MA}}(m)$ *defines the region on which the vector field* $X_{\text{MA}}(m)$ *is defined. Note that there is no requirement that the invariant is a closed set.*

Enabling Conditions $G_{\text{MA}} : E_{\text{MA}} \to \{g_e\}_{e \in E_{\text{MA}}}$ *assigns to each edge* $e = (m, \sigma, m') \in E_{\text{MA}}$*, a non-empty enabling or guard condition* $g_e \subset \mathbb{R}^n$*. We require that* $g_e \subset I_{\text{MA}}(m)$*. We make an additional requirement that* g_e *lies on a certain face of* Y^* *determined by the label* $\sigma = (\sigma_1, \ldots, \sigma_p) \in \Sigma$*. Defining the face associated with* σ *as*

$$\mathcal{F}_\sigma = \left\{ y \in Y^* \;\middle|\; \begin{cases} y_i = 0, & \text{if } \sigma_i = -1 \\ y_i = d_i, & \text{if } \sigma_i = 1 \\ y_i \in [0, d_i], & \text{if } \sigma_i = 0 \end{cases} \right\},$$

we require that also $g_e \subset h^{-1}(\mathcal{F}_\sigma)$*.*

Reset Conditions $R_{\text{MA}} : E_{\text{MA}} \to \{r_e\}_{e \in E_{\text{MA}}}$ *assigns to each edge* $e = (m, \sigma, m') \in E_{\text{MA}}$ *a reset map* $r_e : \mathbb{R}^n \to \mathbb{R}^n$*. We require that* $r_e(x) = x - h_o^{-1}(d \circ \sigma)$*, where* $\circ$ *is the Hadamard product. This definition says that the* i*-th output component is reset to the right face of* Y^**,* $x_{o(i)} = d_i$*, if* $\sigma_i = -1$*, reset to the left face* $x_{o(i)} = 0$ *if* $\sigma_i = 1$*, and unchanged otherwise. Overall, resets of states are determined by the event* $\sigma \in \Sigma$ *and only affect the output coordinates in order to maintain output trajectories inside the canonical box* Y^**.*

Initial Conditions $Q^0_{\text{MA}} \subset Q_{\text{MA}}$ *is the set of initial conditions given by* $Q^0_{\text{MA}} = \{(m, x) \in Q_{\text{MA}} \mid x \in I_{\text{MA}}(m)\}$*.*

Example 4.4.4. Suppose the system is a double integrator and the first state is the translationally invariant output y. The box Y^* is simply a segment. Let $M = \{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$, where *Hold* ($\mathscr{H}$) stabilizes

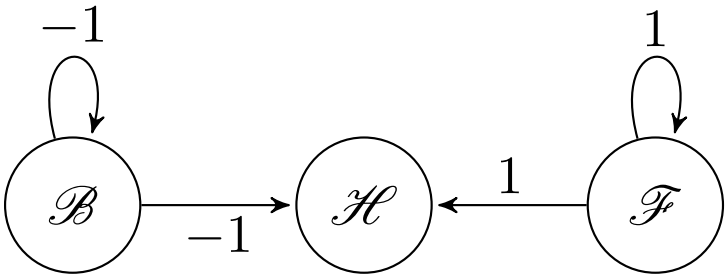

Figure 4.4: The maneuver automaton edges E_{MA} for the double integrator dynamics with $p = 1$. There are three motion primitives: *Hold* ($\mathscr{H}$), *Forward* ($\mathscr{F}$), and *Backward* ($\mathscr{B}$).

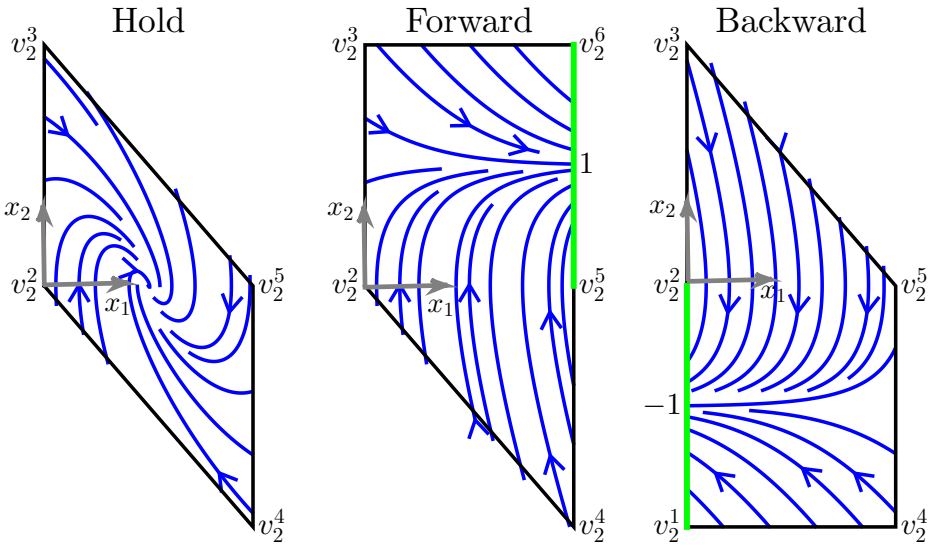

Figure 4.5: The closed-loop vector fields in the (x_1, x_2) position-velocity state space for the *Hold*, *Forward*, and *Backward* motion primitives.

y, *Forward* ($\mathscr{F}$) increases y, and *Backward* ($\mathscr{B}$) decreases y. Referring to Figure 4.4, if $\mathscr{F}$ is the current motion primitive and y reaches the right face of Y^*, then the event $1 \in \Sigma$ occurs and we may select $\mathscr{H}$ or $\mathscr{F}$ as the next motion primitive. To correctly implement the discrete evolution of the MA in the continuous state space, an invariant and feedback control must be associated with each motion primitive, while an enabling and reset condition must be associated with each edge; see Figure 4.5. Formal details are given in Section 4.7.2. ◁

We now formulate assumptions on the motion primitives so that correct continuous time behavior is ensured at the low level for consistency with the high level. For each $m \in M$, define the set of possible events as

$$\Sigma_{\text{MA}}(m) := \{\sigma \in \Sigma \mid (\exists m' \in M)(m, \sigma, m') \in E_{\text{MA}}\}. \tag{4.3}$$

Assumption 4.4.5.

(i) For all $m \in M$, $\varepsilon := (0, \ldots, 0) \notin \Sigma_{\text{MA}}(m)$.

(ii) *For all* $e_1, e_2 \in E_{\text{MA}}$ *such that* $e_1 = (m_1, \sigma, m_2)$ *and* $e_2 = (m_1, \sigma, m_3)$, $g_{e_1} = g_{e_2}$.

(iii) *For all* $e_1, e_2 \in E_{\text{MA}}$ *such that* $e_1 = (m_1, \sigma_1, m_2)$ *and* $e_2 = (m_1, \sigma_2, m_3)$, *if* $\sigma_1 \neq \sigma_2$, *then* $g_{e_1} \cap g_{e_2} = \emptyset$.

(iv) *For all* $e_1, e_2 \in E_{\text{MA}}$ *such that* $e_1 = (m_1, \sigma_1, m_2)$ *and* $e_2 = (m_2, \sigma_2, m_3)$, $r_{e_1}(g_{e_1}) \cap g_{e_2} = \emptyset$.

(v) *For all* $e = (m_1, \sigma, m_2) \in E_{\text{MA}}$, $r_e(g_e) \subset I_{\text{MA}}(m_2)$.

(vi) *For all* $m \in M$, *if* $\Sigma_{\text{MA}}(m) = \emptyset$ *then for all* $x_0 \in I_{\text{MA}}(m)$ *and* $t \geq 0$, $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(m)$.

(vii) *For all* $m \in M$, *if* $\Sigma_{\text{MA}}(m) \neq \emptyset$, *then for all* $x_0 \in I_{\text{MA}}(m)$ *there exist (a unique)* $\sigma \in \Sigma_{\text{MA}}(m)$ *and (a unique)* $T \geq 0$ *such that for all* $e = (m, \sigma, m') \in E_{\text{MA}}$ *and for all* $t \in [0, T]$, $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(m)$ *and* $\phi_{\text{MA}}(T, x_0) \in g_e$.

Condition (i) disallows tautological chattering behavior that arises by erroneously interpreting continuous evolution of trajectories in the interior of Y^* as "discrete transitions" of the MA (see Section 4.5 for definitions). Condition (ii) imposes that guard sets are independent of the next motion primitive. Since guard sets arise as the set of exit points of closed-loop trajectories from Y^* under a given motion primitive, it is reasonable that these exit points should depend only on the current motion primitive $m \in M$, and not on the choice of next motion primitive. Condition (iii) imposes that all guard sets corresponding to different labels are non-overlapping. This ensures that when the continuous trajectory reaches a guard g_e, then it is unambiguous which edge of the MA is taken next; namely $e \in E_{\text{MA}}$. Conditions (v), (vi), and (vii) are placed to guarantee that the MA is non-blocking. These conditions are based on known results in the literature [70]; see Lemma 4.5.5. In order for condition (vii) to make sense, there must exist a unique label $\sigma \in \Sigma$ and a unique time $T \geq 0$ for an MA trajectory to reach a guard set. First, we have uniqueness of solutions since the vector fields are globally Lipschitz. Second, the unique MA trajectory can only reach one guard set by condition (iii); this in turn means there is a unique σ. Obviously there exists a unique time to reach the guard set. Conditions (vi) and (vii) work together to state that either all trajectories do not leave, or all trajectories do eventually leave. Referring to Figure 4.5, all closed-loop state trajectories within the invariant of $\mathscr{F}$ reach the guard set shown in green on the right. For either choice of next feasible motion primitive, $\mathscr{H}$ or $\mathscr{F}$, trajectories enter the next invariant on the left due to the reset. Finally, condition (iv) eliminates potential chattering Zeno behavior, see Remark 4.5.3.

Remark 4.4.6. *We make several further observations about the MA.*

(i) The MA is non-deterministic in the sense that given $m \in M$ *and* $\sigma \in \Sigma$, *there may be multiple* $m' \in M$ *such that* $(m, \sigma, m') \in E_{\text{MA}}$. *The discrete part of the MA is non-deterministic in a second sense: for*

each $m \in M$, the cardinality of the set $\Sigma_{\text{MA}}(m)$ may be greater than one. The latter situation corresponds to the fact that for different initial conditions $x_1, x_2 \in I_{\text{MA}}(m)$ of the continuous part, the associated output trajectories can reach different guard sets. In essence, which guard is enabled is interpreted, at the high level, as an uncontrollable event [119]. Referring to Figure 4.8, the motion primitive $(\mathscr{F}, \mathscr{F})$ may cause the events $(1,0)$, $(0,1)$, or $(1,1) \in \Sigma$. Remark 4.4.8 further illustrates these two types of non-determinism in the case of the PA.

(ii) The set of events Σ in the MA correspond to the same events Σ in the OTS. This correspondence is used in the product automaton PA, described in the next section, to synchronize transitions in the MA with transitions in the OTS. The interpretation is that when a continuous trajectory of the MA (over the box Y^*) undergoes a reset with the label $\sigma \in \Sigma$, the associated continuous trajectory of (4.1) in the box Y_j enters a neighboring box $Y_{j'}$ with the offset $\sigma = l_{j'} - l_j$. Obviously, this interpretation assumes that the vector of box lengths d is the same in both OTS and MA.

4.4.3 Product Automaton

In this section we introduce the *product automaton* (PA). It is constructed as the synchronous product of the OTS and the discrete part of the MA, namely $(M, \Sigma, E_{\text{MA}})$. The purpose of the PA is to merge the constraints on successive motion primitives with the constraints on transitions in the OTS in order to enforce feasible and safe motions. As such, it captures the overall feasible motions of the system – any high level plan must adhere to these feasible motions.

Definition 4.4.7. *We are given an OTS $\mathcal{A}_{\text{OTS}}$ and an MA $\mathcal{H}_{\text{MA}}$ satisfying Assumption 4.4.5. We define the* product automaton *(PA) to be the tuple $\mathcal{A}_{\text{PA}} = (Q_{\text{PA}}, \Sigma, E_{\text{PA}}, Q^f_{\text{PA}})$, where*

State Space *$Q_{\text{PA}} \subset L_{\text{OTS}} \times M$ is a finite set of PA states. A PA state $q = (l, m) \in Q_{\text{PA}}$ satisfies the following: if there exists $\sigma \in \Sigma$ and $m' \in M$ such that $(m, \sigma, m') \in E_{\text{MA}}$, then there exists $l' \in L_{\text{OTS}}$ such that $(l, \sigma, l') \in E_{\text{OTS}}$. That is, $(l, m) \in Q_{\text{PA}}$ if all faces that can be reached by motion primitive $m \in M$ lead to a neighboring box of the box associated with location $l \in L_{\text{OTS}}$ of the OTS.*

Labels *Σ is the same set of labels used by the OTS and the MA.*

Edges *$E_{\text{PA}} \subset Q_{\text{PA}} \times \Sigma \times Q_{\text{PA}}$ is a set of directed edges defined according to the following rule. Let $q = (l, m) \in Q_{\text{PA}}$, $q' = (l', m') \in Q_{\text{PA}}$, and $\sigma \in \Sigma$. If $(l, \sigma, l') \in E_{\text{OTS}}$ and $(m, \sigma, m') \in E_{\text{MA}}$, then $(q, \sigma, q') \in E_{\text{PA}}$.*

Final Condition *$Q^f_{\text{PA}} \subset L^g_{\text{OTS}} \times M$ is the set of final PA states.*

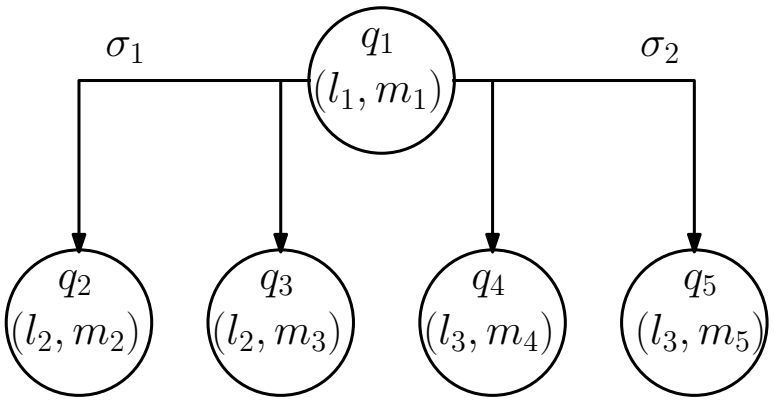

Figure 4.6: A fragment of a generic PA, showing a state and its neighbours.

Remark 4.4.8. *Formally an automaton is said to be non-deterministic if there exists a state with more than one outgoing edge with the same label. The PA is non-deterministic. First, consider a PA state* $q = (l, m) \in Q_{\text{PA}}$*. Because the MA allows for more than one feasible next motion primitive* m' *such that* $(m, \sigma, m') \in E_{\text{MA}}$*, the PA will also have multiple next PA states* $q' = (l', m')$ *such that* $(q, \sigma, q') \in E_{\text{PA}}$*. Second, there can be multiple possible labels* $\sigma \in \Sigma$ *such that* $e = (q, \sigma, q') \in E_{\text{PA}}$ *for some* $q' \in Q_{\text{PA}}$*. Thus, the PA inherits the two types of non-determinism of the MA that we discussed in Remark 4.4.6. For example, consider the PA fragment in Figure 4.6. For the first type of non-determinism, observe that there are two PA edges* $(q_1, \sigma_1, q_2) \in E_{\text{PA}}$ *and* $(q_1, \sigma_1, q_3) \in E_{\text{PA}}$ *with the same label. For the second type, observe that there are two possible events* $\sigma_1, \sigma_2 \in \Sigma$ *from* q_1*, each with its own set of PA edges. Note also some additional structure: since the OTS is deterministic, the box state is* l_2 *in both* q_2 *and* q_3*, corresponding to the OTS edge* $(l_1, \sigma_1, l_2) \in E_{\text{OTS}}$*.*

4.4.4 High-Level Plan

In this section we formulate the notion of a control policy on the PA, which gives a rule for selecting subsequent PA states by choosing the next motion primitive. Informally, the objective of the high level plan is to produce a control policy and find a set of initial PA states such that a goal PA state is eventually reached. To this end, in this section we also develop a Dynamic Programming Principle (DPP) suitable for use on the PA. Because of the two types of non-determinism of the PA, existing algorithms cannot be applied directly [17, 116]. By adapting the algorithm in [17], we obtain two formulations of the DPP, one of which is more computationally efficient as it exploits certain structure in the PA; further details are provided in Remark 4.4.12.

First some notation will be useful. Recall from (4.3), given $m \in M$, $\Sigma_{\text{MA}}(m)$ is the set of all labels $\sigma \in \Sigma$ on outgoing edges $e \in E_{\text{MA}}$ starting at m. Similarly, $\Sigma_{\text{PA}}(q)$ is the set of all labels $\sigma \in \Sigma$ on

outgoing edges $e \in E_{\text{PA}}$ starting at q. That is,

$$\Sigma_{\text{PA}}(q) := \{\sigma \in \Sigma \mid (\exists q' \in Q_{\text{PA}})(q, \sigma, q') \in E_{\text{PA}}\}.$$

Now we formalize the semantics of the PA. A *state* of the PA is a pair $q = (l, m) \in Q_{\text{PA}}$ where $l \in L_{\text{OTS}}$ is a location in the OTS and $m \in M$ is a motion primitive. A *run* π of $\mathcal{A}_{\text{PA}}$ is a finite or infinite sequence of states $\pi = q^0 q^1 q^2 \ldots$, with $q^i = (l^i, m^i) \in Q_{\text{PA}}$ and for each i, there exists $\sigma^i \in \Sigma_{\text{PA}}(q^i)$ such that $(q^i, \sigma^i, q^{i+1}) \in E_{\text{PA}}$. We define the length of a run to be n_π; for infinite runs n_π is defined to be ∞. We consider a subset of runs $\Pi_{\text{PA}}(q)$ starting at $q \in Q_{\text{PA}}$ that satisfy one further property. If the run π is infinite, then $\pi \in \Pi_{\text{PA}}(q)$ if $q^0 = q$. Instead if the run π is finite, then $\pi \in \Pi_{\text{PA}}(q)$ if $q^0 = q$ and additionally, $\Sigma_{\text{PA}}(q^{n_\pi}) = \emptyset$. It is the latter requirement – that the last PA state of a finite run may not have outgoing edges in the PA – which is of interest. The interpretation is that we regard the event labels between PA states as uncontrollable, so if any event is possible, then it must occur eventually. Thus without loss of generality, each run $\pi = q^0 q^1 \ldots$ is the prefix of a run $\pi' \in \Pi_{\text{PA}}(q^0)$. Further elaboration is given in Remark 4.4.9 (ii).

Given $q \in Q_{\text{PA}}$ and $\sigma \in \Sigma_{\text{PA}}(q)$, the set of *admissible motion primitives* is

$$\mathcal{M}(q, \sigma) := \{m' \in M \mid (\exists q' = (l', m'))\ (q, \sigma, q') \in E_{\text{PA}}\}.$$

More generally, given $q \in Q_{\text{PA}}$ and $\Sigma_{\text{PA}}(q) = \{\sigma_1, \ldots, \sigma_k\}$, the set of admissible motion primitives at q is

$$\mathcal{M}(q) := \{(m_1, \ldots, m_k) \mid m_i \in \mathcal{M}(q, \sigma_i),\ i = 1, \ldots, k\},$$

Next we introduce the notion of a control policy. Given $q \in Q_{\text{PA}}$ and $\Sigma_{\text{PA}}(q) = \{\sigma_1, \ldots, \sigma_k\}$, an *admissible control assignment* at q is a vector

$$c(q) = (c(q, \sigma_1), \ldots, c(q, \sigma_k)),$$

where $c(q, \sigma_i) \in \mathcal{M}(q, \sigma_i)$, or equivalently $c(q) \in \mathcal{M}(q)$. Notice that $c(q)$ is a vector whose dimension varies as a function of the cardinality of the set $\Sigma_{\text{PA}}(q)$. An *admissible control policy* $c : Q_{\text{PA}} \times \Sigma \to M$ is a map that assigns an admissible control assignment at each $q \in Q_{\text{PA}}$. Thus, for each $q \in Q_{\text{PA}}$ and $\sigma \in \Sigma_{\text{PA}}$, $c(q, \sigma) \in \mathcal{M}(q, \sigma)$. The set of all admissible control policies is denoted by $\mathcal{C}$.

Consider an admissible control policy $c \in \mathcal{C}$ and a state $q \in Q_{\text{PA}}$. We denote the set of runs in $\Pi_{\text{PA}}(q)$ induced by c as $\Pi_c(q)$. Formally, $\pi = q^0 q^1 \cdots \in \Pi_c(q)$ if $q^0 = q$, and for all $i \geq 0$ and $i < n_\pi$,

$m^{i+1} = c(q^i, \sigma^i)$. Similarly, we denote the subset of runs in $\Pi_c(q)$ that eventually reach a state in Q^f_{PA} as $\Pi^f_c(q)$. Formally, $\pi \in \Pi^f_c(q)$ if there exists an integer $i \in \{0, \dots, n_\pi\}$ such that $q^i \in Q^f_{\text{PA}}$. For $\pi \in \Pi^f_c(q)$, we define

$$r_\pi := \min\{i \in \{0, \dots, n_\pi\} \mid q^i \in Q^f_{\text{PA}}\}\,.$$

Next we define an instantaneous cost $D_{\text{PA}} : E_{\text{PA}} \to \mathbb{R}$, which satisfies $D_{\text{PA}}(e) > 0$ for all $e \in E_{\text{PA}}$, and a terminal cost $H_{\text{PA}} : Q_{\text{PA}} \to \mathbb{R}$. Now consider the run $\pi = q^0 q^1 \dots q^{n_\pi} \in \Pi^f_c(q)$ with $q^0 = q$, $c(q^i, \sigma^i) = m^{i+1}$, and $e^i := (q^i, \sigma^i, q^{i+1}) \in E_{\text{PA}}$. We define a *cost-to-go* $J : Q_{\text{PA}} \times \mathcal{C} \to \mathbb{R}$ by

$$J(q, c) = \begin{cases} \max\limits_{\pi \in \Pi_c(q)} \left\{ \sum\limits_{j=0}^{r_\pi - 1} D_{\text{PA}}(e^j) + H_{\text{PA}}(q^{r_\pi}) \right\}, & \Pi_c(q) = \Pi^f_c(q) \\ \infty, & \text{otherwise}\,. \end{cases}$$

Remark 4.4.9. *There are several notable features of our formula for the cost-to-go.*

(i) For a given $q \in Q_{\text{PA}}$, there may be multiple runs $\pi \in \Pi_c(q)$ due to the (second, non-standard type of) non-determinism of the PA. As such, we take the maximum over $\Pi_c(q)$ in the cost-to-go because of this non-determinacy in $\mathcal{A}_{\text{PA}}$: it is uncertain which, among the possibly multiple trajectories allowed by c, that will be taken so we assume the worst-case situation. Moreover, we require $\Pi_c(q) = \Pi^f_c(q)$ for a finite cost-to-go, otherwise there may exist a run starting at q and applying control policy c that does not reach Q^f_{PA}. Also, when $\Pi_c(q) = \Pi^f_c(q)$, r_π is well-defined.

(ii) We have assumed that finite runs must terminate on PA states that have no outgoing edges. The motivation for this choice becomes clear in light of the formulation of the cost-to-go. Suppose we included in $\Pi_c(q)$ finite prefixes of (finite or infinite) runs. These necessarily would be finite runs with final PA states that have outgoing edges. Then if we take a finite or infinite run that eventually reaches a goal PA state, certain finite prefixes of that run may not yet have reached a goal PA state, and we would get $\Pi_c(q) \neq \Pi^f_c(q)$ and an infinite cost-to-go. This anomaly arises from creating an artificial situation in which not all runs starting at an initial PA state reach a goal PA state because we included (unsuccessful) finite prefixes of successful runs.

(iii) The cost-to-go function also accounts for infinite runs by using the variable r_π to record the first time a goal PA state is reach and by taking the cost only over the associated prefix of the infinite run. Allowing infinite runs seems to contradict a reach-avoid specification in which we only want finite runs that terminate on goal PA states with no outgoing edges. The reason we also allow infinite runs in the formulation of the high level plan is that it allows us to extend our framework to a fragment of LTL where, for example, a goal PA state is reached always eventually; see Remark 4.5.9 for further details.

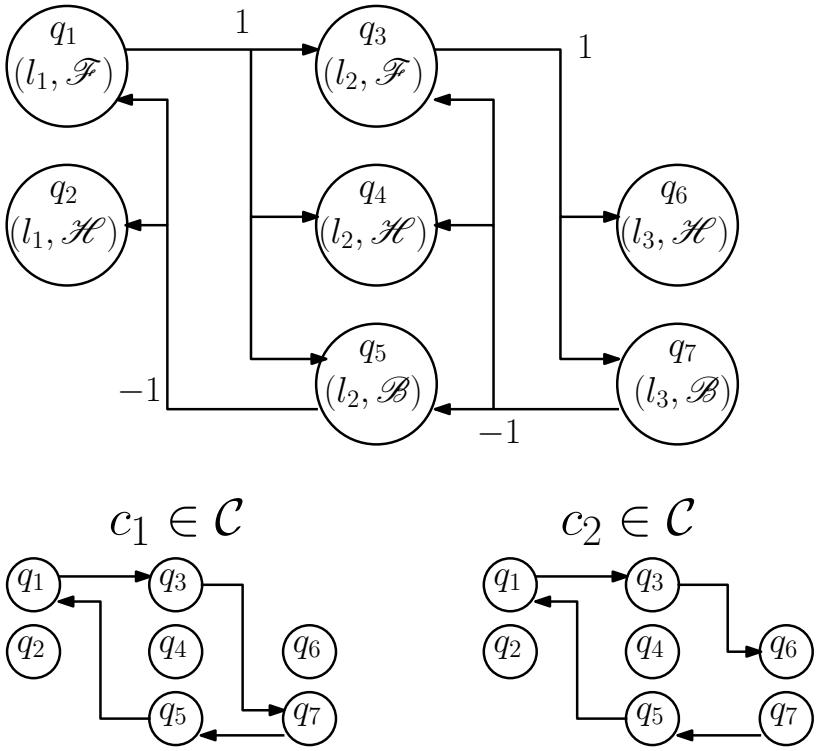

Figure 4.7: At the top, a PA is depicted for a single output system having three motion primitives $M = \{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$ over three boxes $L_{\text{OTS}} = \{l_j \mid j = 1, 2, 3\}$. The numbers $1, -1 \in \Sigma$ on the edges (shown as arrows) are the corresponding labels. The bottom pictures show the reduced set of transitions induced by control polices $c_1, c_2 \in \mathcal{C}$.

Example 4.4.10. Consider the PA shown at the top of Figure 4.7 corresponding to a single output system with the three motion primitives $M = \{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$ from Example 4.4.4 over three boxes $L_{\text{OTS}} = \{l_j \mid j = 1, 2, 3\}$. Suppose that $D_{\text{PA}}(e) = 1$ for all $e \in E_{\text{PA}}$ and that $H_{\text{PA}} = 0$ for all $q \in Q_{\text{PA}}$.

First consider the feasible control policy $c_1 \in \mathcal{C}$ with the control assignments: $c_1(q_1, 1) = \mathscr{F}$, $c_1(q_3, 1) = \mathscr{B}$, $c_1(q_5, -1) = \mathscr{F}$, and $c_1(q_7, -1) = \mathscr{B}$. The bottom left of Figure 4.7 shows how the control policy trims away possible edges in the PA. Now suppose that $Q^f_{\text{PA}} = \{q_7\}$. Choosing the initial condition $q_1 \in Q_{\text{PA}}$ and under the assumption that we do not include finite runs that terminate at PA states with outgoing edges, we can see that $\Pi_{c_1}(q_1)$ consists of only the single infinite run $\pi = q_1 q_3 q_7 q_5 q_1 \ldots$. Even though this run is infinite, $\pi \in \Pi^f_{c_1}(q_1)$, $r_\pi = 2$, and $J(q_1, c_1) = 2$. Similarly, we compute $J(q_5, c_1) = 3$, $J(q_3, c_1) = 1$, $J(q_7, c_1) = 0$, and $J(q_2, c_1) = J(q_4, c_1) = J(q_6, c_1) = \infty$.

Next, consider the feasible control policy $c_2 \in \mathcal{C}$ with the control assignments: $c_2(q_1, 1) = \mathscr{F}$, $c_2(q_3, 1) = \mathscr{H}$, $c_2(q_5, -1) = \mathscr{F}$, and $c_2(q_7, -1) = \mathscr{B}$. This control policy is shown on the bottom right of Figure 4.7. Suppose that $Q^f_{\text{PA}} = \{q_6\}$. Then we find $J(q_7, c_2) = 4$, $J(q_5, c_2) = 3$, $J(q_1, c_2) = 2$, $J(q_3, c_2) = 1$, $J(q_6, c_2) = 0$, and $J(q_2, c_2) = J(q_4, c_2) = \infty$. The difference between c_1 and c_2 is that runs are infinite under c_1 but finite under c_2.

Finally, suppose we had omitted the extra condition that finite runs must terminate on PA states with no outgoing edges. Considering $c_1 \in \mathcal{C}$ at $q_1 \in Q_{\text{PA}}$, then $\Pi_{c_1}(q_1)$ would contain an infinite number of finite runs $\{q_1, q_1 q_3, q_1 q_3 q_7, \ldots\}$ as well as the infinte run already noted. In particular, the two runs $q_1, q_1 q_3 \notin \Pi^f_{c_1}(q_1)$, so by definition of the cost-to-go, we would obtain the undesired result $J(q_1, c_1) = \infty$. A similar problem would arise with the control policy c_2. ◁

Next we define the *value function* $V : Q_{\text{PA}} \to \mathbb{R}$ to be

$$V(q) := \min_{c \in \mathcal{C}} J(q, c) \,.$$

The value function satisfies a dynamic programming principle (DPP) that takes into account the nondeterminacy of $\mathcal{A}_{\text{PA}}$; see [17] where a slightly different result is proved.

Theorem 4.4.11. *Consider $q \in Q_{\text{PA}} \setminus Q^f_{\text{PA}}$ and suppose $|\Sigma_{\text{PA}}(q)| > 0$. Then V satisfies*

$$V(q) = \min_{c(q) \in \mathcal{M}(q)} \left\{ \max_{\sigma \in \Sigma_{\text{PA}}(q)} \{D_{\text{PA}}(e) + V(q')\} \right\} \tag{4.4}$$

$$= \max_{\sigma \in \Sigma_{\text{PA}}(q)} \left\{ \min_{\bar{m} \in \mathcal{M}(q, \bar{\sigma})} \{D_{\text{PA}}(\bar{e}) + V(\bar{q})\} \right\} , \tag{4.5}$$

where $q' = (l', c(q, \sigma)) \in Q_{\text{PA}}$, $e = (q, \sigma, q') \in E_{\text{PA}}$, $\bar{q} = (\bar{l}, \bar{m}) \in Q_{\text{PA}}$, and $\bar{e} = (q, \sigma, \bar{q}) \in E_{\text{PA}}$.

Notice that for all $q \in Q_{\text{PA}} \setminus Q^f_{\text{PA}}$ such that $|\Sigma_{\text{PA}}(q)| = 0$, $V(q) = \infty$ (since there can be no paths to the goal). Also, for all $q \in Q^f_{\text{PA}}$, $V(q) = H_{\text{PA}}(q)$. The proof is due to Zach Kroeze [61] and is omitted here.

Remark 4.4.12. *In* (4.4) *of Theorem 4.4.11, it is shown that $V(q)$ can be computed using the local information of $\mathcal{M}(q)$ instead of using all of $\mathcal{C}$. In* (4.5)*, the result is taken one step further by showing that $V(q)$ can be calculated using only $\mathcal{M}(q, \sigma)$ for each $\sigma \in \Sigma_{\text{PA}}(q)$. The benefit of* (4.5) *becomes clear when we compare the cardinality of the sets over which the minimizations occur. Given $q \in Q_{\text{PA}}$, let $\Sigma_{\text{PA}}(q) = \{\sigma_1, \ldots, \sigma_k\}$. In* (4.4) *the minimization is over $\mathcal{M}(q)$, and therefore the cardinality of the minimization set is $\prod_{i=1}^{k} |\mathcal{M}(q, \sigma_k)|$. In* (4.5) *the minimization is over $\mathcal{M}(q, \sigma)$ for each $\sigma \in \Sigma_{\text{PA}}(q)$, and therefore the cardinality of the set is $|\mathcal{M}(q, \sigma)|$. While both* (4.4) *and* (4.5) *can be used to compute $V(q)$, in general* (4.5) *will be more computationally efficient.*

Corollary 4.4.13. *Consider the control policy c^* such that for all $q \in Q_{\text{PA}}$, and $\sigma \in \Sigma_{\text{PA}}(q)$*

$$c^*(q, \sigma) \in \operatorname*{argmin}_{m' \in \mathcal{M}(q,\sigma)} \{D_{\text{PA}}(e) + V(q')\},$$

where $q' = (l', m')$, and $e = (q, \sigma, q')$. Then c^* is an optimal control policy such that for all $q \in Q_{\text{PA}}$, $V(q) = J(q, c^*)$.

Figure 4.8 shows a possible control policy for the scenario in Figure 4.3. Since there are two outputs, we use the motion primitives from Example 4.4.4 in each output; formal details are given in Section 4.6. The control policy was hand-computed. Notice that different routes may be taken from the same product state depending on the face reached, but ultimately the control policy ensures that all paths lead to the goal.

4.5 Main Results

In this section we present our main results on a solution to Problem 4.3.1. Our final result combines the notion of a control policy at the high level with feedback controllers executing correct continuous time behavior at the low level. First, in accordance with the reach-avoid objective (see Remark 4.4.9 (iii)), we restrict the final PA states to be goal OTS states equipped with motion primitives that have no guard sets

$$Q^f_{\text{PA}} = \{(l, m) \in L^g_{\text{OTS}} \times M \mid \Sigma_{\text{MA}}(m) = \emptyset\}. \tag{4.6}$$

Now suppose we have an admissible control policy $c \in \mathcal{C}$ derived using Theorem 4.4.11 or otherwise with Q^f_{PA} as above. We present a complete solution to Problem 4.3.1 including an initial condition set

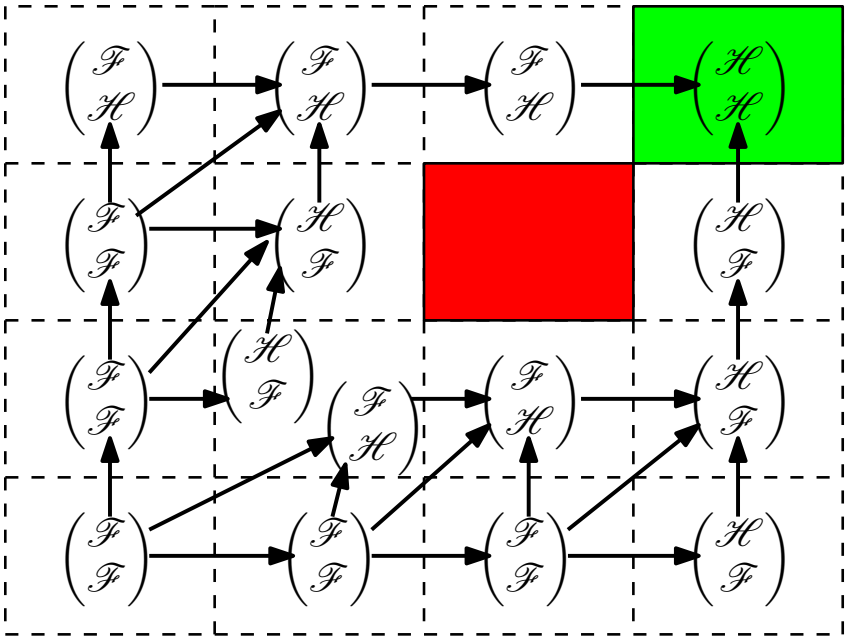

Figure 4.8: This figure shows a discrete control strategy for the scenario shown in Figure 4.3.

$\mathcal{X}_0 \subset \mathbb{R}^n$, a feedback control $u(x)$, and conditions on the motion primitives so that the reach-avoid specifications of Problem 4.3.1 are met.

First we specify the initial condition set $\mathcal{X}_0$. The set of feasible initial PA states is

$$Q^0_{\text{PA}} := \{q \in Q_{\text{PA}} \mid \Pi_c(q) = \Pi^f_c(q)\}\,. \tag{4.7}$$

That is, a feasible initial PA state satisfies that every run (induced by the control policy) starting at the PA state eventually reaches a goal PA state.

Now consider a state $x_0 \in \mathbb{R}^n$. It can be used as an initial state of the system if there is some $(l_j, m) \in Q^0_{\text{PA}}$ for which the state is both in the box Y_j and in the invariant of m. Recall that for all $y \in \mathbb{R}^p$ and $j \in \{1, \ldots, n_L\}$, $y \in Y^*$ if and only if $y + d \circ l_j \in Y_j$. With this in mind, we define the set of initial states to be:

$$\mathcal{X}_0 = \bigcup_{(l_j,m)\in Q^0_{\text{PA}}} \left\{x + h_o^{-1}(d \circ l_j) \mid x \in I_{\text{MA}}(m)\right\}. \tag{4.8}$$

Next we specify the feedback controllers to solve Problem 4.3.1. Consider any $q = (l_j, m) \in Q^0_{\text{PA}}$. Then for all $x \in \mathbb{R}^n$ such that $x - h_o^{-1}(d \circ l_j) \in I_{\text{MA}}(m)$, we define the feedback

$$u(x, q) := u_m(x - h_o^{-1}(d \circ l_j))\,. \tag{4.9}$$

This defines a family of feedback controllers parametrized by x, the state of (4.1) and by the PA state

$q = (l_j, m)$. These feedbacks work in tandem with the control policy $c \in \mathcal{C}$, which effectively determines the next feasible PA state $q' \in Q^0_{\text{PA}}$. For example, suppose $q = (l_j, m) \in Q^0_{\text{PA}}$ and suppose the label $\sigma \in \Sigma$ is measured. This event corresponds to $x \in g_e$ for $e = (m, \sigma, c(q, \sigma)) \in E_{\text{MA}}$. Let $m' := c(q, \sigma)$ and let $l' \in L_{\text{OTS}}$ be the unique location of the OTS such that $(l, \sigma, l') \in E_{\text{OTS}}$. Then the next PA state is $q' = (l', m') \in Q^0_{\text{PA}}$ and the controller that is applied in the next location $l' \in L_{\text{OTS}}$ is $u_{m'}(\cdot)$.

The main result of this chapter is the following.

Theorem 4.5.1. *Consider the system* (4.1) *satisfying Assumption 4.3.2, the non-empty feasible set* $\mathcal{P} \subset \mathbb{R}^p$*, and the goal set* $\mathcal{G} \subset \mathcal{P}$*. Let* d *be the vector of box lengths such that the goal indices* I^g *is non-empty. Consider an associated OTS* $\mathcal{A}_{\text{OTS}}$*, an MA* $\mathcal{H}_{\text{MA}}$ *satisfying Assumption 4.4.5, a PA* $\mathcal{A}_{\text{PA}}$ *with* Q^f_{PA} *as in* (4.6)*, and an admissible control policy* $c \in \mathcal{C}$*. Then the initial condition set* $\mathcal{X}_0$ *given in* (4.8) *and the feedback controllers* (4.9) *solve Problem 4.3.1.*

In the remainder of this section we prove Theorem 4.5.1. We now give a roadmap for these results. The verification of correctness at the low level is broken down into two steps that we now describe. First, we show that the MA is non-blocking in Lemma 4.5.5. The key requirements are summarized in Assumption 4.4.5. The non-blocking condition ensures that MA trajectories continually evolve in time and stay within the invariant regions. We also put conditions to avoid chattering in which two discrete transitions can occur in immediate succession. While physical systems never undergo infinite switching in finite time, if our model predictions diverge from reality, then we have no grounds to claim that Problem 4.3.1 is indeed solved. Second, in Lemma 4.5.6 we show that to each closed-loop trajectory of (4.1) under the feedback controllers (4.9) and a control policy $c \in \mathcal{C}$, we can associate a unique execution of the MA (defined below) and run of the PA.

We begin by describing the semantics of the MA. These definitions are standard; see [70]. A *state* of the MA is a pair (m, x), where $m \in M$ and $x \in \mathbb{R}^n$. Trajectories of the MA are called *executions* and are defined over hybrid time domains that identify the time intervals when the trajectory of a hybrid system is in a fixed motion primitive $m \in M$. Precisely, a *hybrid time domain* of the MA is a finite or infinite sequence of intervals $\tau = \{\mathcal{I}_0, \ldots, \mathcal{I}_{n_\tau}\}$, such that

(i) $\mathcal{I}_i = [\tau_i, \tau'_i]$, for all $0 \leq i < n_\tau$,

(ii) if $n_\tau < \infty$, then either $\mathcal{I}_{n_\tau} = [\tau_{n_\tau}, \tau'_{n_\tau}]$ or $I_{n_\tau} = [\tau_{n_\tau}, \tau'_{n_\tau})$,

(iii) $\tau_i \leq \tau'_i = \tau_{i+1}$, for all $0 \leq i < n_\tau$.

Definition 4.5.2. *An* execution *of the MA is a collection* $\chi = (\tau, m(\cdot), \phi_{\text{MA}}(\cdot, x_0))$ *such that*

(i) *the initial condition of the execution satisfies:* $(m(0), x_0) \in Q^0_{\text{MA}}$.

(ii) *the continuous evolution of the execution satisfies: for all* $i \in \{0, \dots, n_\tau\}$ *with* $\tau_i < \tau'_i$, *then for all* $t \in [\tau_i, \tau'_i]$, $m(\cdot)$ *is constant and* $\frac{d}{dt}\phi_{\text{MA}}(t, x_0) = f(\phi_{\text{MA}}(t, x_0), u_{m(t)}(\phi_{\text{MA}}(t, x_0)))$, *while for all* $t \in [\tau_i, \tau'_i)$, $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(m(t))$.

(iii) *a discrete transition of the execution satisfies: for all* $i \in \{0, \dots, n_\tau - 1\}$, *there exists* $\sigma_i \in \Sigma_{\text{MA}}(m(\tau'_i))$ *such that* $(m(\tau'_i), \sigma_i, m(\tau_{i+1})) =: e_i \in E_{\text{MA}}$, $\phi_{\text{MA}}(\tau'_i, x_0) \in g_{e_i}$, *and* $\phi_{\text{MA}}(\tau_{i+1}, x_0) = r_{e_i}(\phi_{\text{MA}}(\tau'_i, x_0))$.

Given an execution $\chi = (\tau, m(\cdot), \phi_{\text{MA}}(\cdot, x_0))$, we associate to it the *output trajectory of the MA* given by $y_{\text{MA}}(\cdot, x_0) := h(\phi_{\text{MA}}(\cdot, x_0))$ (the subscript MA is included to avoid confusion with output trajectories $y(\cdot, x_0)$ of the physical system (4.1) which do not undergo resets). The *execution time* of an execution χ is defined as $\mathscr{T}(\chi) := \sum_{i=0}^{n_\tau}(\tau'_i - \tau_i) = \lim_{i \to n_\tau} \tau'_i$. An execution is called *finite* if τ is a finite sequence ending with a compact time interval. An execution is called *infinite* if either τ is an infinite sequence or if $\mathscr{T}(\chi) = \infty$. Finally, an execution is called *Zeno* if it is infinite but $\mathscr{T}(\chi) < \infty$.

Remark 4.5.3. *There are two types of Zeno behavior. In one type that we call chattering, transitions are instantaneous. The second more subtle type is when the times between discrete transitions of the MA converge to zero, but the transitions are not instantaneous. Assumptions 4.4.5 (i) and (iv) ensure that we cannot have chattering. True Zeno behavior with convergent transition times is more difficult to identify in the setting when the MA is formed as a parallel composition. Fortunately, for our reach-avoid objective, the induced MA executions cannot be Zeno since there are a finite number of transitions by construction, see Lemma 4.5.6.*

Definition 4.5.4. *The MA is* non-blocking *if for all* $(m(0), x_0) \in Q^0_{\text{MA}}$, *the set of all infinite executions of the MA with initial condition* $(m(0), x_0)$ *is non-empty.*

Lemma 4.5.5. *Under Assumption 4.4.5, the MA is non-blocking.*

Proof. Let $(m, x) \in Q^0_{\text{MA}}$. If $\Sigma_{\text{MA}}(m) = \emptyset$, then by Assumption 4.4.5 (vi), $I_{\text{MA}}(m)$ is invariant, so the trajectory $\phi_{\text{MA}}(t, x)$ starting at (m, x) remains in $I_{\text{MA}}(m)$ for all future time. Therefore, trivially, the MA is non-blocking for this initial condition. If $\Sigma_{\text{MA}}(m) \neq \emptyset$, then by Assumption 4.4.5 (vii), $\phi_{\text{MA}}(t, x)$ remains in $I_{\text{MA}}(m)$ until it reaches a guard set. Additionally, by Assumption 4.4.5 (v), the trajectory is mapped under the reset into the next invariant. By Lemma 1 of [70], the MA is again non-blocking for this initial condition. Overall, the MA is non-blocking. □

The purpose of the Assumptions 4.4.5 is to guarantee consistency between low level continuous time behavior and the high level discrete plan. This consistency is formalized by way of a one-to-one correspondence between infinite MA executions and finite PA runs, both starting from the same initial condition.

Lemma 4.5.6. *Suppose we have an admissible control policy $c \in \mathcal{C}$, and we have an MA satisfying Assumption 4.4.5. For each $(l^0, m^0) \in Q^0_{\text{PA}}$ and $x_0 \in I_{\text{MA}}(m^0)$ there exist a unique infinite MA execution $\chi = (\tau, m(\cdot), \phi_{\text{MA}}(\cdot, x_0))$ and a unique finite PA run $\pi = q^0 q^1 \dots q^N$.*

Proof. Let $(l^0, m^0) \in Q^0_{\text{PA}}$ and $x_0 \in I_{\text{MA}}(m^0)$. The initial MA state of the MA execution is $(m(0), x_0) = (m^0, x_0) \in Q^0_{\text{MA}}$, and the initial PA state of the PA run π is $q^0 = (l^0, m^0)$. The hybrid time domain of χ will be denoted as $\tau = \{\mathcal{I}_i\}_{i=0}^{n_\tau}$, and is initialized as $\tau = \{\mathcal{I}_0\}$, where $\mathcal{I}_0 = \{\tau_0\}$ and $\tau_0 = 0$. With the base case $k = 0$ established, we construct the remainder of the MA execution and PA run by induction.

The run so far is $\pi = q^0 \dots q^k$, where $q^i = (l^i, m^i)$ for $i = 0, \dots, k$, and $\mathcal{I}_k = \{\tau_k\}$. Suppose $\Sigma_{\text{MA}}(m^k) = \emptyset$. Then by Assumption 4.4.5 (vi), $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(m^k)$ for all t in the extended interval $\mathcal{I}_k = [\tau_k, \infty)$. The complete PA run is $\pi = q^0 \cdots q^k$ and the induction terminates. Suppose instead $\Sigma_{\text{MA}}(m^k) \neq \emptyset$. Then by Assumption 4.4.5 (vii), there exist unique $\sigma^k \in \Sigma_{\text{MA}}(m^k)$ and $T^k \geq 0$, such that $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(m^k)$ for all t in the extended interval $\mathcal{I}^k = [\tau_k, \tau'_k]$, $\tau'_k := \tau_k + T^k$. Also, for each $e = (m^k, \sigma^k, m') \in E_{\text{MA}}$, there exists a guard set g_e such that $\phi_{\text{MA}}(\tau'_k, x_0) \in g_e$. Assumption 4.4.5 (ii) tells us that for all such m', the guard set is the same. Also, Assumption 4.4.5 (iii) ensures that σ^k is unique. Now we invoke the control policy to select a specific m'. Let $m^{k+1} := c(q^k, \sigma^k)$ so that $e^k := (m^k, \sigma^k, m^{k+1}) \in E_{\text{MA}}$ and $\phi_{\text{MA}}(\tau'_k, x_0) \in g_{e^k}$. Define $x_0^{k+1} := r_{e^k}(\phi_{\text{MA}}(\tau'_k, x_0))$. By Assumption 4.4.5 (iv), $x_0^{k+1} \notin g_e$ for any $e = (m^{k+1}, \sigma, m') \in E_{\text{MA}}$. The next PA state is $q^{k+1} = (l^{k+1}, m^{k+1})$, where $l^{k+1} \in L_{\text{OTS}}$ is uniquely determined through $(l^k, \sigma^k, l^{k+1}) \in E_{\text{OTS}}$, by the determinism of the OTS. The PA run so far is $\pi = q^0 \cdots q^{k+1}$ and the new interval $\mathcal{I}_{k+1} = \{\tau'_k\}$ is added to τ.

The above inductive process is guaranteed to terminate with a finite PA run by definition of Q^0_{PA}. That is, since $(l^0, m^0) \in Q^0_{\text{PA}}$ there will be a smallest N such that $(l^N, m^N) \in Q^f_{\text{PA}}$. Moreover, by definition of Q^f_{PA} (4.6), we have that $\Sigma_{\text{MA}}(m^N) = \emptyset$ and so the run cannot be extended further. The resulting MA execution is infinite with a finite number of intervals in the hybrid time domain τ, and it is non-blocking by Lemma 4.5.5. □

Before we can prove Theorem 4.5.1 we need one further preliminary result stating that because of the translational invariance of Assumption 4.3.2, the continuous part of an MA execution has a unique correspondence to a closed-loop trajectory of the system (4.1). For a proof, refer to the analogous Lemma 5.5.5 in Chapter 5.

Lemma 4.5.7. *Let $m \in M$, $x_0 \in I_{\text{MA}}(m)$, $y \in \mathbb{R}^p$, and $\tilde{x}_0 = x_0 + h_o^{-1}(y)$. Consider the trajectory $\phi(t, \tilde{x}_0)$ of (4.1) with the feedback control $u(x) = u_m(x - h_o^{-1}(y))$. Also consider the MA trajectory $\phi_{\text{MA}}(t, x_0)$ with feedback control $u_m(x)$. For all $t \geq 0$ such that $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(m)$,*

$$\phi(t, \tilde{x}_0) = \phi_{\text{MA}}(t, x_0) + h_o^{-1}(y).$$

Finally we are ready to prove Theorem 4.5.1.

Proof of Theorem 4.5.1. We must show that (i) output trajectories of system (4.1) remain within $\mathcal{P}$, and (ii) output trajectories eventually reach and remain within the goal set $\mathcal{G}$. Let $\tilde{x}_0 \in \mathcal{X}_0$. Choose any $(l_{j^0}, m^0) \in Q^0_{\text{PA}}$ such that $x_0 := \tilde{x}_0 - h_o^{-1}(d \circ l_{j^0}) \in I_{\text{MA}}(m^0)$. By Lemma 4.5.6, we may associate a unique MA execution χ and a unique PA run π to $(l_{j^0}, m^0) \in Q^0_{\text{PA}}$ and $x_0 \in I_{\text{MA}}(m^0)$. Denote the hybrid time domain as $\tau = \{\mathcal{I}_0, \ldots, \mathcal{I}_N\}$ with $\mathcal{I}_k = [\tau_k, \tau'_k]$ for $k = 0, \ldots, N-1$ (with $\tau_0 = 0$) and $\mathcal{I}_N = [\tau_N, \infty)$. The last interval follows from the definition of $(l_{j^N}, m^N) \in Q^f_{\text{PA}}$ (4.6), since $\Sigma_{\text{MA}}(m^N) = \emptyset$ and thus Assumption 4.4.5 (vi) implies that we must have that $\mathcal{I}_N = [\tau_N, \infty)$. As in the proof of Lemma 4.5.6, denote the corresponding sequence of events as $\sigma^0 \cdots \sigma^{N-1}$.

Using Lemma 4.5.7 with $y = d \circ l_{j^0}$, we have that $\phi(t, \tilde{x}_0) = \phi_{\text{MA}}(t, x_0) + h_o^{-1}(d \circ l_{j^0})$. We claim that for all $k = 0, \ldots, N$ and $t \in \mathcal{I}_k$,

$$\phi(t, \tilde{x}_0) = \phi_{\text{MA}}(t, x_0) + h_o^{-1}(d \circ l_{j^k}). \tag{4.10}$$

Clearly the result is true for $k = 0$.

We derive two facts to assist in proving this claim. Recall that by definition of the OTS edges, we have that for all $k = 0, \ldots, N-1$, $\sigma^k = l_{j^{k+1}} - l_{j^k}$. Furthermore, by rearranging, multiplying component-wise by d, and taking the preimage h_o^{-1}, we have the first fact: for all $k = 0, \ldots, N-1$ that $h_o^{-1}(d \circ l_{j^{k+1}}) = h_o^{-1}(d \circ l_{j^k}) + h_o^{-1}(d \circ \sigma^k)$. Also by definition of the reset map and MA execution, we get the second fact: for all $k = 0, \ldots, N-1$, $r_{e^k}(\phi_{\text{MA}}(\tau'_k, x_0)) = \phi_{\text{MA}}(\tau'_k, x_0) - h_o^{-1}(d \circ \sigma^k) = \phi_{\text{MA}}(\tau_{k+1}, x_0)$.

Returning to (4.10), by induction we assume that it is true for $0 \leq k < N$ and show that it is true for $k+1$. Using the above facts and (4.10) for k at $t = \tau'_k = \tau_{k+1}$ yields

$$\begin{aligned}
\phi(\tau_{k+1}, \tilde{x}_0) = \phi(\tau'_k, \tilde{x}_0) &= \phi_{\text{MA}}(\tau'_k, x_0) + h_o^{-1}(d \circ l_{j^k}) \\
&= (\phi_{\text{MA}}(\tau_{k+1}, x_0) + h_o^{-1}(d \circ \sigma^k)) + h_o^{-1}(d \circ l_{j^k}) \\
&= \phi_{\text{MA}}(\tau_{k+1}, x_0) + h_o^{-1}(d \circ l_{j^{k+1}}).
\end{aligned}$$

Applying Lemma 4.5.7 with $y = h_o^{-1}(d \circ l_{j^{k+1}})$ at the new initial condition $\phi_{\text{MA}}(\tau_{k+1}, x_0) \in I_{\text{MA}}(m^{k+1})$, we have that for $k+1$ and for all $t \in \mathcal{I}_{k+1}$ that (4.10) holds. When $k+1 = N$, the induction terminates and the claim is proven.

Using (4.10) and projecting to the output space we conclude that for all $k = 0, \ldots, N$ and $t \in \mathcal{I}_k$, $y(t, \tilde{x}_0) \in Y_{j^k}$. Since all the boxes are contained in $\mathcal{P}$ by construction, then for all $t \geq 0$ we have (i). Moreover, since $l_{j^N} \in L^g_{\text{OTS}}$ implies the goal box Y_{j^N} is contained in $\mathcal{G}$ and $\mathcal{I}_N = [\tau_N, \infty)$, we have (ii). □

Remark 4.5.8. *The above result does not depend on the method of construction of the admissible control policy $c \in \mathcal{C}$, nor does it require the control policy to be optimal. This allows for different path planning techniques on the PA, as we show in Section 4.8.2.*

Remark 4.5.9. *The extension to a sequence of reach-avoid problems is straightforward, following the idea in [116]. First, the reach property (ii) of Problem 4.3.1 is relaxed to $y(T, x_0) \in \mathcal{G}$. Next, suppose there is a finite sequence of goals $L^{g,i}_{\text{OTS}}$, $i = 1, ..., n_g > 1$. In contrast to* (4.6)*, we set the final PA states to be $Q^{f,i}_{\text{PA}} = \{(l, m) \in L^{g,i}_{\text{OTS}} \times M \mid \Sigma_{\text{MA}}(m) \neq \emptyset\}$ for $i = 1, \ldots, n_g - 1$. Finally, one must design control policies c_i with associated initial conditions $Q^{0,i}_{\text{PA}}$* (4.7) *such that $Q^{f,i}_{\text{PA}} \subset Q^{0,i+1}_{\text{PA}}$ for $i = 1, \ldots, n_g - 1$. For $i = n_g$, one may impose solutions to remain invariant or connect back to the first goal.*

4.6 Parallel Composition of Motion Primitives

In this section we describe the operation of parallel composition of two maneuver automata. By repeated application of this operation, more complex higher-dimensional MA's can be constructed by starting from simple low dimensional atomic motion primitives, such as those described in Section 4.7.2. The key challenge is to ensure that the resulting parallel composed MA satisfies Assumptions 4.4.5, if the two constituent MA's do. This is proved in Theorem 4.6.2. First we give some preliminary definitions and we fix some notation, followed by the formal definition of parallel composition of MA's.

We consider two independent systems

$$\dot{x}^j = f^j(x^j, u^j), \;\; y^j = h^j(x^j), \tag{4.11}$$

where $x^j \in \mathbb{R}^{n^j}$, $u^j \in \mathbb{R}^{\mu^j}$, and $y^j \in \mathbb{R}^{p^j}$ for $j = 1, 2$. We use superscripts to identify the distinct subsystems. Assume that each system satisfies Assumption 4.3.2. That is, for $j = 1, 2$, $y^j_i = x^j_i$,

$i = 1, \ldots, p^j$. Associated with each system $j = 1, 2$ is the MA

$$\mathcal{H}^j_{\text{MA}} = (Q^j_{\text{MA}}, \Sigma^j, E^j_{\text{MA}}, X^j_{\text{MA}}, I^j_{\text{MA}}, G^j_{\text{MA}}, R^j_{\text{MA}}, Q^{0,j}_{\text{MA}}). \tag{4.12}$$

We additionally assume that $\mathcal{H}^1_{\text{MA}}$ and $\mathcal{H}^2_{\text{MA}}$ satisfy Assumption 4.4.5. Denote the canonical boxes in the respective output spaces as $Y^{*,j} = \prod_{i=1}^{p^j}[0, d^j_i]$. The event sets labelling the faces of $Y^{*,j}$ are $\Sigma^j = \{-1, 0, 1\}^{p^j}$. The empty strings are denoted as $\varepsilon^j := (0, \ldots, 0) \in \Sigma^j$, $j = 1, 2$, and the empty string is $\varepsilon := (\varepsilon^1, \varepsilon^2)$. Other sets are similarly denoted with a superscript to identify the system, such as the set of possible events $\Sigma^j_{\text{MA}}(m^j)$ for $m^j \in M^j$ and the output indices o^j. For the parallel composition we also require some extra notation. First, for $j = 1, 2$ and for each $m^j \in M^j$, define the invariant set minus all the guard sets

$$I^j(m^j) := I^j_{\text{MA}}(m^j) \setminus \left(\bigcup_{e^j = (m^j, \sigma^j, m^j_2) \in E^j_{\text{MA}}} g_{e^j} \right). \tag{4.13}$$

Next, we need three sets: an augmented set of edges that includes a transition with the empty string, an augmented set of possible events for a motion primitive $m \in M^j$, and an augmented set of next feasible motion primitives. That is, for $j = 1, 2$, we define

$$\begin{aligned}
\overline{E}^j_{\text{MA}} &:= E^j_{\text{MA}} \cup \big\{ (m^j, \varepsilon^j, m^j_2) \mid m^j, m^j_2 \in M^j, I^j(m^j) \subset I^j_{\text{MA}}(m^j_2), \\
&\qquad (\forall e^j_2 = (m^j_2, \sigma^j_2, m^j_3) \in E^j_{\text{MA}})\ I^j(m^j) \cap g_{e^j_2} = \emptyset \big\} \\
\overline{\Sigma}^j_{\text{MA}}(m^j) &:= \Sigma^j_{\text{MA}}(m) \cup \{\varepsilon^j\}, \qquad m^j \in M^j \\
\overline{M}^j(m^j, \sigma^j) &:= \{ m^j_2 \in M^j \mid (m^j, \sigma^j, m^j_2) \in \overline{E}^j_{\text{MA}} \}, \qquad m^j \in M^j, \sigma^j \in \overline{\Sigma}^j_{\text{MA}}(m^j).
\end{aligned}$$

We also define the products of these sets:

$$\begin{aligned}
\overline{\Sigma}_{\text{MA}}(m) &:= \overline{\Sigma}^1_{\text{MA}}(m^1) \times \overline{\Sigma}^2_{\text{MA}}(m^2), \qquad m = (m^1, m^2) \in M, \\
\overline{M}(m, \sigma) &:= \overline{M}^1(m^1, \sigma^1) \times \overline{M}^2(m^2, \sigma^2), \qquad m = (m^1, m^2) \in M, \sigma = (\sigma^1, \sigma^2) \in \overline{\Sigma}_{\text{MA}}(m).
\end{aligned}$$

Finally, the canonical box in the output space of the parallel composition is $Y^* = Y^{*,1} \times Y^{*,2}$. We can now define the parallel composition of two MA's.

Definition 4.6.1. *Consider two MA's $\mathcal{H}^1_{\text{MA}}$ and $\mathcal{H}^2_{\text{MA}}$ each satisfying Assumption 4.4.5. The parallel composition $\mathcal{H}^1_{\text{MA}} \mid\mid \mathcal{H}^2_{\text{MA}}$ is $\mathcal{H}_{\text{MA}} = (Q_{\text{MA}}, \Sigma, E_{\text{MA}}, X_{\text{MA}}, I_{\text{MA}}, G_{\text{MA}}, R_{\text{MA}}, Q^0_{\text{MA}})$ where*

State Space $Q_{\text{MA}} = M \times \mathbb{R}^n$ *with* $M = M^1 \times M^2$ *and* $n = n^1 + n^2$.

Labels $\Sigma = \Sigma^1 \times \Sigma^2 = \{-1, 0, 1\}^p$ *with* $p = p^1 + p^2$.

Edges $E_{\text{MA}} \subset M \times \Sigma \times M$, *where* $e = (m, \sigma, m') \in E_{\text{MA}}$ *if* $\sigma \neq \varepsilon$, $\sigma \in \overline{\Sigma}_{\text{MA}}(m)$, *and* $m' \in \overline{M}(m, \sigma)$. *Observe that for all* $m \in M$, $\Sigma_{\text{MA}}(m) = \overline{\Sigma}_{\text{MA}}(m) \setminus \{\varepsilon\}$.

Vector Fields *For all* $m = (m^1, m^2) \in M$, $X_{\text{MA}}(m) = \begin{bmatrix} f^1(x^1, u_{m^1}(x^1)) \\ f^2(x^2, u_{m^2}(x^2)) \end{bmatrix}$. *The state is* $x := (x^1, x^2) \in \mathbb{R}^n$, *the control input is* $u := (u^1, u^2) \in \mathbb{R}^\mu$ *where* $\mu = \mu^1 + \mu^2$, *and the output is* $y := (y^1, y^2) \in \mathbb{R}^p$. *The output map is* $h(x) = \begin{bmatrix} h^1(x^1) \\ h^2(x^2) \end{bmatrix}$, *with* $o(i) = o^1(i)$ *for* $i = 1, \ldots, p^1$ *and* $o(i) = n^1 + o^2(i - p^1)$ *for* $i = p^1 + 1, \ldots, p$.

Invariants *For all* $m = (m^1, m^2) \in M$, $I_{\text{MA}}(m) = I^1_{\text{MA}}(m^1) \times I^2_{\text{MA}}(m^2)$.

Enabling and Reset Conditions *Consider an edge* $e = (m_1, \sigma, m_2) \in E_{\text{MA}}$, *where* $m_1 = (m_1^1, m_1^2) \in M$, $\sigma = (\sigma^1, \sigma^2) \in \overline{\Sigma}_{\text{MA}}(m)$, $m_2 = (m_2^1, m_2^2) \in \overline{M}(m_1, \sigma)$, *and* $e^j = (m_1^j, \sigma^j, m_2^j) \in \overline{E}^j_{\text{MA}}$ *for* $j = 1, 2$. *If* $\sigma^j \in \overline{\Sigma}^j_{\text{MA}}(m_1^j)$ *and* $\sigma^j = \varepsilon^j$, *then we define*

$$g_{e^j} := I^j(m_1^j), \quad r_{e^j}(x^j) := x^j.$$

Otherwise if $\sigma^j \in \Sigma^j_{\text{MA}}(m_1^j)$, *we have* $g_{e^j} = G^j_{\text{MA}}(e^j)$ *and* $r_{e^j} = R^j_{\text{MA}}(e^j)$, *corresponding to their definitions in* $\mathcal{H}^j_{\text{MA}}$. *Finally, we define* $g_e = g_{e^1} \times g_{e^2}$ *and* $r_e(x) = \begin{bmatrix} r_{e^1}(x^1) \\ r_{e^2}(x^2) \end{bmatrix}$.

Initial Conditions $Q^0_{\text{MA}} \subset Q_{\text{MA}}$ *is the set of initial conditions given by* $Q^0_{\text{MA}} = \{(m, x) \mid (m^j, x^j) \in Q^{0,j}_{\text{MA}}, i = 1, 2\}$.

First, notice that for each $\mathcal{H}^j_{\text{MA}}$ and for each $m^j \in M^j$, the definition of $\overline{E}^j_{\text{MA}}$ automatically includes self-loop edges $(m, \varepsilon^j, m) \in \overline{E}^j_{\text{MA}}$. We include such transitions with ε^j so that the parallel composition is properly constructed. For example, suppose a proper face of $Y^{*,1}$ is crossed by the first system, but no proper face of $Y^{*,2}$ is crossed by the second system. To correctly account for such possibilities, the overall transition for the composed MA must record the lack of crossing in $Y^{*,2}$ by the empty string ε^2. Second, notice that we have allowed for additional edges with ε^j to allow for the possibility of switching to a different motion primitive over the same box $Y^{*,j}$ if the invariants overlap and are not mapped immediately to a guard set, as can be observed by the definition of $\overline{E}^j_{\text{MA}}$. Referring to Figure 4.8, an edge such as $((\mathscr{F}, \mathscr{H}), (1, 0), (\mathscr{H}, \mathscr{F})) \in E_{\text{MA}}$ consists of $(\mathscr{F}, 1, \mathscr{H}) \in E^1_{\text{MA}}$ and $(\mathscr{H}, 0, \mathscr{F}) \in \overline{E}^2_{\text{MA}}$, which encodes a turn from Right to Up.

The main result is now stated.

Theorem 4.6.2. *We are given $\mathcal{H}^1_{\text{MA}}$ and $\mathcal{H}^2_{\text{MA}}$, two MA's that satisfy Assumption 4.4.5. The parallel composition $\mathcal{H}_{\text{MA}} = \mathcal{H}^1_{\text{MA}} \mid\mid \mathcal{H}^2_{\text{MA}}$ defined above is an MA that also satisfies Assumption 4.4.5.*

Proof. We employ the following two standard facts regarding products, intersections, and subsets of sets. Formally, if A, B, C, D are sets, then

$$(A \cap C) \times (B \cap D) = (A \times B) \cap (C \times D), \tag{4.14}$$

$$A \subset C \text{ and } B \subset D \Rightarrow (A \times B) \subset (C \times D). \tag{4.15}$$

First we show that the resulting $\mathcal{H}_{\text{MA}}$ is in fact an MA according to the definition. Clearly the composed vector fields are also globally Lipschitz and the composed invariants are bounded. The non-trivial points to show are that (a) the stacked system satisfies Assumption 4.3.2, (b) the invariants project within the canonical box, (c) the enabling conditions lie both within the invariant and on an appropriate face determined by $\sigma \in \Sigma$, (d) the reset conditions are determined only by the event $\sigma \in \Sigma$, and (e) the initial conditions are the entire invariants. We prove each of these in turn.

(a) We show that Assumption 4.3.2 for the stacked system holds. For the first condition, it can be verified by direct expansion that the definition of h necessarily produces the injective output map $o : \{1, \ldots, p\} \to \{1, \ldots, n\}$ defined earlier. For the second condition, letting $x = (x^1, x^2)$, $u = (u^1, u^2)$ and $y = (y^1, y^2)$, we must show that $f(x, u) = f(x + h^{-1}(y), u)$. First, by Assumption 4.3.2 on each system, $f^j(x^j, u^j) = f^j(x^j + (h^j_{o^j})^{-1}(y^j), u^j)$. Second, it is easy (but tedious) to show that $h_o^{-1}(y) = ((h^1_{o^1})^{-1}(y^1), (h^2_{o^2})^{-1}(y^2))$. Putting these two facts together gives the desired result.

(b) We show that for all $m \in M$, $I_{\text{MA}}(m) \subset h^{-1}(Y^*)$. Letting $m = (m^1, m^2) \in M$, we have by the fact that each system is an MA that $I^j_{\text{MA}}(m^j) \subset (h^j)^{-1}(Y^{*,j})$ for $j = 1, 2$. It is easy (but tedious) to show that $h^{-1}(Y^*) = (h^1)^{-1}(Y^{*,1}) \times (h^2)^{-1}(Y^{*,2})$. The result then follows by applying (4.15).

(c) We show that for all $e = (m_1, \sigma, m_2) \in E_{\text{MA}}$, $g_e \subset h^{-1}(\mathcal{F}_\sigma) \cap I_{\text{MA}}(m_1)$. Let $e = (m_1, \sigma, m_2) \in E_{\text{MA}}$ and decompose it as $e^j = (m^j_1, \sigma^j, m^j_2) \in \overline{E}^j_{\text{MA}}$ for $j = 1, 2$. For $j = 1, 2$, if $\sigma^j \neq \varepsilon^j$, then $g_{e^j} \subset (h^j)^{-1}(\mathcal{F}_{\sigma^j}) \cap I^j_{\text{MA}}(m^j_1)$ since each system is an MA. Otherwise, if $\sigma^j = \varepsilon^j$, observe that $\mathcal{F}_{\varepsilon^j} = Y^{*,j}$ and $g_{e^j} = I^j(m^j_1) \subset I^j_{\text{MA}}(m^j_1) \subset (h^j)^{-1}(Y^{*,j})$ by construction. Consequently $g_{e^j} \subset (h^j)^{-1}(\mathcal{F}_{\sigma^j}) \cap I^j_{\text{MA}}(m^j_1)$ again. Next, by definition $g_e = g_{e^1} \times g_{e^2}$ and $I_{\text{MA}}(m) = I^1_{\text{MA}}(m^1_1) \times I^2_{\text{MA}}(m^2_1)$. It is also easy (but tedious) to show that $h^{-1}(\mathcal{F}_\sigma) = (h^1)^{-1}(\mathcal{F}_{\sigma^1}) \times (h^2)^{-1}(\mathcal{F}_{\sigma^2})$. The result then follows by applying (4.14) and (4.15).

(d) We show that for all $e = (m_1, \sigma, m_2) \in E_{\text{MA}}$, $r_e(x) = x - h_o^{-1}(d \circ \sigma)$. Let $e = (m_1, \sigma, m_2) \in E_{\text{MA}}$

and decompose it as $e^j = (m_1^j, \sigma^j, m_2^j) \in \overline{E}^j_{\text{MA}}$ for $j = 1, 2$. First, by definition $r_e(x) = (r_{e^1}(x^1), r_{e^2}(x^2))$. Then for $j = 1, 2$, if $\sigma^j \neq \varepsilon^j$, then $r_{e^j}(x^j) = x^j - (h^j_{o^j})^{-1}(d^j \circ \sigma^j)$ since each system is an MA. This is also the case when $\sigma^j = \varepsilon^j$ because $r_{e^j}(x^j) = x^j$ and $(h^j_{o^j})^{-1}(d^j \circ \varepsilon^j) = 0$. Next, since $d = (d^1, d^2)$ and $\sigma = (\sigma^1, \sigma^2)$, component-wise multiplication gives $d \circ \sigma = (d^1 \circ \sigma^1, d^2 \circ \sigma^2)$. Using $x = (x^1, x^2)$ and the decomposition $h_o^{-1}(y) = ((h^1_{o^1})^{-1}(y^1), (h^2_{o^2})^{-1}(y^2))$ established in (a) with $y = d \circ \sigma$ proves the result.

(e) We must show that $Q^0_{\text{MA}} = \{(m, x) \in Q_{\text{MA}} \mid x \in I_{\text{MA}}(m)\}$. This follows immediately from the definitions of Q_{MA}, I_{MA}, and Q^0_{MA}.

Next we prove that (i)-(vii) of Assumption 4.4.5 hold.

(i) We must show that for all $m \in M$, $\varepsilon \notin \Sigma_{\text{MA}}(m)$. This follows immediately from the definition of the edges since for all $m \in M$, $\Sigma_{\text{MA}}(m) = \overline{\Sigma}_{\text{MA}}(m) \setminus \{\varepsilon\}$.

(ii) We must show that for all $e_1, e_2 \in E_{\text{MA}}$ such that $e_1 = (m_1, \sigma, m_2)$ and $e_2 = (m_1, \sigma, m_3)$, $g_{e_1} = g_{e_2}$. To that end, we write $e_1^j = (m_1^j, \sigma^j, m_2^j) \in \overline{E}^j_{\text{MA}}$ and $e_2^j = (m_1^j, \sigma^j, m_3^j) \in \overline{E}^j_{\text{MA}}$ for $j = 1, 2$. To show that $g_{e_1} = g_{e_2}$, we must show that $g_{e_1^j} = g_{e_2^j}$ for $j = 1, 2$. Let $j \in \{1, 2\}$. If $\sigma^j = \varepsilon^j$, then by construction $g_{e_1^j} = I^j(m_1^j) = g_{e_2^j}$. Otherwise, if $\sigma^j \neq \varepsilon^j$, then $g_{e_1^j} = g_{e_2^j}$ follows from Assumption 4.4.5 (ii) on the j-th system.

(iii) We must show that for all $e_1, e_2 \in E_{\text{MA}}$ such that $e = (m_1, \sigma_1, m_2)$ and $e_2 = (m_1, \sigma_2, m_3)$, if $\sigma_1 \neq \sigma_2$, then $g_{e_1} \cap g_{e_2} = \emptyset$. To that end, we write $e_1^j = (m_1^j, \sigma_1^j, m_2^j) \in \overline{E}^j_{\text{MA}}$ and $e_2^j = (m_1^j, \sigma_2^j, m_3^j) \in \overline{E}^j_{\text{MA}}$ for $j = 1, 2$. If $\sigma_1 \neq \sigma_2$, then suppose w.l.o.g. that $\sigma_1^1 \neq \sigma_2^1$. To show $g_{e_1} \cap g_{e_2} = \emptyset$, by (4.14) it suffices to show that $g_{e_1^1} \cap g_{e_2^1} = \emptyset$. If both σ_1^1 and σ_2^1 are not equal to ε^1, then by Assumption 4.4.5 (iii) $g_{e_1^1} \cap g_{e_2^1} = \emptyset$. If one of σ_1^1 or σ_2^1 is ε^1, say σ_1^1, then we cannot invoke Assumption 4.4.5 (iii). However, by definition $g_{e_1^1} = I^1(m_1^1)$ is not intersecting with any other guards, so that $g_{e_1^1} \cap g_{e_2^1} = \emptyset$, as desired.

(iv) We must show that for all $e_1, e_2 \in E_{\text{MA}}$ such that $e_1 = (m_1, \sigma_1, m_2)$ and $e_2 = (m_2, \sigma_2, m_3)$, $r_{e_1}(g_{e_1}) \cap g_{e_2} = \emptyset$. To that end, we write $e_1^j = (m_1^j, \sigma_1^j, m_2^j) \in \overline{E}^j_{\text{MA}}$ and $e_2^j = (m_2^j, \sigma_2^j, m_3^j) \in \overline{E}^j_{\text{MA}}$ for $j = 1, 2$. By (4.14), it suffices to show that $r_{e_1^j}(g_{e_1^j}) \cap g_{e_2^j} = \emptyset$ for at least one of $j = 1, 2$. Since ε cannot be an event by Assumption 4.4.5 (i), at least one of $j_1 = 1, 2$ must have $\sigma_1^{j_1} \neq \varepsilon^{j_1}$, and at least one of $j_2 = 1, 2$ must have $\sigma_2^{j_2} \neq \varepsilon^{j_2}$. Formally, there are several cases, but they can be summarized as follows. If there is at least one matching $j = j_1 = j_2$, where both $\sigma_1^j, \sigma_2^j \neq \varepsilon^j$, then both $e_1^j, e_2^j \in E^j_{\text{MA}}$, and we invoke Assumption 4.4.5 (iv) to get $r_{e_1^j}(g_{e_1^j}) \cap g_{e_2^j} = \emptyset$. There are two remaining cases which are similar so we only look at one of them. Suppose $\sigma_1^1 = \varepsilon^1$ and $\sigma_2^2 = \varepsilon^2$. Then $e_1^1 \notin E^1_{\text{MA}}$ and $e_2^2 \notin E^2_{\text{MA}}$ so we cannot invoke Assumption 4.4.5 (iv). However, by construction $e_1^1 \in \overline{E}^1_{\text{MA}}$ implies that $r_{e_1^1}(g_{e_1^1}) = I^1(m_1^1)$ is not intersecting with any guards in E^1_{MA}. In particular, $e_2^1 \in E^1_{\text{MA}}$ since it must be that $\sigma_2^1 \neq \varepsilon^1$, and so $r_{e_1^1}(g_{e_1^1}) \cap g_{e_2^1} = \emptyset$.

(v) We must show that for all $e = (m_1, \sigma, m_2) \in E_{\text{MA}}$, $r_e(g_e) \subset I_{\text{MA}}(m_2)$. To that end, we write $e^j = (m_1^j, \sigma^j, m_2^j) \in \overline{E}^j_{\text{MA}}$ for $j = 1, 2$. For $j = 1, 2$, if $\sigma^j \neq \varepsilon^j$, then $r_{e^j}(g_{e^j}) \subset I^j_{\text{MA}}(m_2^j)$ follows from Assumption 4.4.5 (v) on the individual system. This is also the case when $\sigma^j = \varepsilon^j$, since by definition of $\overline{E}^j_{\text{MA}}$ we have $r_{e^j}(g_{e^j}) = I^j(m_1^j) \subset I^j_{\text{MA}}(m_2^j)$. Next by definition we have that $I_{\text{MA}}(m_2) = I^1_{\text{MA}}(m_2^1) \times I^2_{\text{MA}}(m_2^2)$. Also, it is easy to verify that $r_e(g_e) = r_{e^1}(g_{e^1}) \times r_{e^2}(g_{e^2})$. The result follows by applying (4.15).

(vi) We must show that for all $m \in M$, if $\Sigma_{\text{MA}}(m) = \emptyset$ then $I_{\text{MA}}(m)$ is invariant. Let $m = (m^1, m^2) \in M$, $x_0 = (x_0^1, x_0^2) \in I_{\text{MA}}(m)$, and suppose that $\Sigma_{\text{MA}}(m) = \emptyset$. Then for $j = 1, 2$, $\Sigma^j_{\text{MA}}(m^j) = \emptyset$. To see this, suppose one was not empty, say $\sigma^1 \in \Sigma^1_{\text{MA}}(m^1) \subset \overline{\Sigma}^1_{\text{MA}}(m^1)$, where by Assumption 4.4.5 (i) $\sigma^1 \neq \varepsilon^1$. By construction, $\varepsilon^2 \in \overline{\Sigma}^2_{\text{MA}}(m^2)$ and by Assumption 4.4.5 (i) proven above $(\sigma^1, \varepsilon^2) \in \overline{\Sigma}_{\text{MA}}(m) \setminus \{\varepsilon\} = \Sigma_{\text{MA}}(m)$, so we get a contradiction. Appealing to the Assumption 4.4.5 (vi) for $j = 1, 2$, for all $t \geq 0$ the individual trajectories satisfy $\phi^j_{\text{MA}}(t, x_0^j) \in I^j_{\text{MA}}(m^j)$. Thus for all $t \geq 0$, $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(m)$.

(vii) We must show that for all $m \in M$, if $\Sigma_{\text{MA}}(m) \neq \emptyset$ then $I_{\text{MA}}(m)$ forces all trajectories to exit in finite time through some guard. Let $m = (m^1, m^2) \in M$, $x_0 = (x_0^1, x_0^2) \in I_{\text{MA}}(m)$, and suppose that $\Sigma_{\text{MA}}(m) \neq \emptyset$. Reversing the argument used in Assumption 4.4.5 (vi), now we can conclude for at least one of $j = 1, 2$ that $\Sigma^j_{\text{MA}}(m^j) \neq \emptyset$. Suppose first that only $\Sigma^1_{\text{MA}}(m^1) \neq \emptyset$, then applying Assumption 4.4.5 (vii) for $j = 1$ and Assumption 4.4.5 (vi) for $j = 2$ furnishes the event $\sigma = (\sigma^1, \varepsilon^2) \in \Sigma_{\text{MA}}(m)$, with exit time T^1. A similar argument holds if only $\Sigma^2_{\text{MA}}(m^2) \neq \emptyset$. If for both $j = 1, 2$, $\Sigma^j_{\text{MA}}(m^j) \neq \emptyset$, then Assumption 4.4.5 (vii) for both $j = 1, 2$ furnishes the event $\sigma = (\sigma^1, \sigma^2)$ with $\sigma^j \in \Sigma^j_{\text{MA}}(m^j)$, with exit times T^j. If $T^1 < T^2$, then the overall exit time is $T = T^1$ with event $\sigma = (\sigma^1, \varepsilon^2) \in \Sigma_{\text{MA}}(m)$. If $T^1 > T^2$, then the exit time is $T = T^2$ with event $\sigma = (\varepsilon^1, \sigma^2) \in \Sigma_{\text{MA}}(m)$. Otherwise, $T = T^1 = T^2$, and the event is $\sigma = (\sigma^1, \sigma^2) \in \Sigma_{\text{MA}}(m)$. In all the cases above, it is then easy (but tedious) to verify that for all $e = (m, \sigma, m') \in E_{\text{MA}}$ and for all $t \in [0, T]$, $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(m)$ and $\phi_{\text{MA}}(T, x_0) \in g_e$.

□

Remark 4.6.3. *We have defined the event set as $\Sigma = \Sigma^1 \times \Sigma^2$, but the usual parallel composition of automata would have $\Sigma = \Sigma^1 \cup \Sigma^2$ [119]. Given the interpretation of the event set as crossing faces of Y^*, the cartesian product is the more natural choice.*

4.7 Motion Primitives for Integrator Systems

In this section we provide a design of a maneuver automaton for single integrator systems and two designs of maneuver automata for double integrator systems. These designs are able to be succinctly expressed within the MA formalism since the underlying single and double integrator systems satisfy

Assumption 4.3.2.

The single integrator MA was developed in [61] and is needed for Chapter 6. The first MA for double integrators was also originally developed by Zach Kroeze [61], and was introduced earlier in this chapter in Example 4.4.4. The second MA for double integrators is a novel addition to this chapter. Only the first MA for double integrators was used for experiments on quadrocopters, as the single integrator is not dynamically suitable and the second MA for double integrators would overburden the computation of control policies as the number of vehicles grows.

4.7.1 Single Integrator: Hold, Forward, and Backward

In this section we present a MA for a single integrator system. It consists of three motion primitives *Hold* ($\mathscr{H}$), *Forward* ($\mathscr{F}$), and *Backward* ($\mathscr{B}$), as its double integrator counterpart considered in Example 4.4.4. It mainly varies in its implementation at the continuous level.

The nonlinear control system (4.1) is the single integrator system

$$\dot{x} = u, \;\; y = x,$$

where $x, u, y \in \mathbb{R}$, and the output y is the position. Each motion primitive's invariant region is simply the segment $Y^* = [0, d]$. Let $u^* > 0$ be the maximum control. For each $m \in M := \{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$, we design an affine feedback $u_m(x) = K_m x + g_m$ over each invariant using the methodology of the RCP. Our design enforces that trajectories stabilize to the middle of Y^* for *Hold*, increase in y for *Forward*, and decrease in y for *Backward*. It suffices to choose a control value at the vertices $\{0, d\}$ and then use (2.3) to obtain K_m and g_m: we select $\{u^*, -u^*\}$ for *Hold*, $\{u^*, u^*\}$ for *Forward*, and $\{-u^*, -u^*\}$ for *Backward* for the two vertices, respectively. This yields

$$\begin{aligned} u_{\mathscr{H}}(x) &= K_{\mathscr{H}} x + g_{\mathscr{H}} = (-2u^*/d)x + u^* = (-2u^*/d)(x - d/2) \\ u_{\mathscr{F}}(x) &= K_{\mathscr{F}} x + g_{\mathscr{F}} = u^* \\ u_{\mathscr{B}}(x) &= K_{\mathscr{B}} x + g_{\mathscr{B}} = -u^* \end{aligned}$$

Now we construct the MA. The state space is $Q_{\text{MA}} = M \times \mathbb{R}$. The labels are $\Sigma = \{-1, 0, 1\}$. The set of edges E_{MA} are the same as those shown in Figure 4.4 for the double integrator version. For each $m \in M$, the closed-loop vector fields are given by $[X_{\text{MA}}(m)](x) = u_m(x)$, which are clearly globally Lipschitz, while the invariants are simply $I_{\text{MA}}(m) = [0, d]$. The enabling conditions are $g_e = \{d\}$ for edges of the form $e = (\mathscr{F}, 1, \cdot) \in E_{\text{MA}}$ and $g_e = \{0\}$ for edges of the form $e = (\mathscr{B}, -1, \cdot) \in E_{\text{MA}}$. The reset and initial

conditions are constructed according to their definition. Finally, it is quite trivial to show that this MA design satisfies Assumption 4.4.5, so details are omitted.

4.7.2 Double Integrator: Hold, Forward, and Backward

In this section we give the formal details for the MA consisting of the three motion primitives *Hold* ($\mathscr{H}$), *Forward* ($\mathscr{F}$), and *Backward* ($\mathscr{B}$) introduced in Example 4.4.4. By exploiting the parallel composition construction from Section 4.6, the usefulness of this MA is demonstrated in the context of multi-agent systems in Section 4.8.

Suppose the nonlinear control system (4.1) is the double integrator system:

$$\dot{x}_1 = x_2, \;\; \dot{x}_2 = u_2, \qquad y = x_1, \tag{4.16}$$

where $x := (x_1, x_2) \in \mathbb{R}^2$, $u_2 \in \mathbb{R}$, and the output y is the position. Each motion primitive's invariant region is a polytopic set in the state space defined as the convex hull of vertices v^k, $k \in \{1, \ldots, 6\}$; see Figure 4.5. The vertices are determined by the segment length $d > 0$, and a pre-specified maximum control value $u^* > 0$. Let $v^* := \sqrt{du^*}$ be the maximum speed, and let $k_1 := -2u^*/d$ and $k_2 := -2u^*/v^*$ be controller gains. The vertices are $v^1 = (0, -v^*)$, $v^2 = (0, 0)$, $v^3 = (0, v^*)$, $v^4 = (d, -v^*)$, $v^5 = (d, 0)$, and $v^6 = (d, v^*)$. For each motion primitive $m \in M := \{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$, we define an affine feedback $u_m(x) = K_m x + g_m$. Then

$$\begin{aligned}
u_{\mathscr{H}}(x) &= K_{\mathscr{H}} x + g_{\mathscr{H}} = \begin{bmatrix} k_1 & k_2 \end{bmatrix} x + u^* = k_1(x_1 - d/2) + k_2 x_2 \\
u_{\mathscr{F}}(x) &= K_{\mathscr{F}} x + g_{\mathscr{F}} = \begin{bmatrix} 0 & k_2 \end{bmatrix} x + u^* = k_2(x_2 - v^*/2) \\
u_{\mathscr{B}}(x) &= K_{\mathscr{B}} x + g_{\mathscr{B}} = \begin{bmatrix} 0 & k_2 \end{bmatrix} x - u^* = k_2(x_2 + v^*/2)
\end{aligned}$$

This yields the vector fields shown in Figure 4.5. It can be observed that *Hold* causes trajectories to stabilize to the middle of the segment, while *Forward* causes trajectories to stabilize to the speed $v^*/2$ and similarly for *Backward*.

Remark 4.7.1. *These controllers can alternatively be derived using reach control theory [61]. The polytopes are triangulated into simplices and control values are assigned at the vertices. Using* (2.3)*, an affine feedback is constructed on each simplex; in our case, for each motion primitive the same feedback results on each simplex to ensure smoothness in the control. Technically* Hold *does not solve the RCP on these simplices, although* Forward *and* Backward *do so.*

Now we construct the MA. The state space is $Q_{\text{MA}} = M \times \mathbb{R}^2$. The labels are $\Sigma = \{-1, 0, 1\}$. The set of edges E_{MA} are shown in Figure 4.4. In the context of parallel composition, one may compute that the augmented edges are

$$\overline{E}_{\text{MA}} = E_{\text{MA}} \cup \{(m, 0, m)\}_{m \in M} \cup \{(\mathscr{H}, 0, \mathscr{F}), (\mathscr{H}, 0, \mathscr{B})\}.$$

For each $m \in M$, the closed-loop vector fields are given by $[X_{\text{MA}}(m)](x) = (x_2, u_m(x))$, which are clearly globally Lipschitz. The invariants are given by the convex hull of vertices, as seen in Figure 4.5, and excluding the two points $(0, 0)$ and $(d, 0)$, so the invariants are clearly bounded. For example, $I_{\text{MA}}(\mathscr{H}) = \text{co}\{v^k\}_{k=2}^5 \setminus \{(0, 0), (d, 0)\}$. The enabling conditions are constructed by taking the convex hull of vertices of the exit facet and excluding again $(0, 0)$ or $(d, 0)$. Specifically, the edges $(\mathscr{F}, 1, \mathscr{H}), (\mathscr{F}, 1, \mathscr{F}) \in E_{\text{MA}}$ both have guard sets $g_e = \text{co}\{v^5, v^6\} \setminus \{(d, 0)\} = \{d\} \times (0, v^*]$, as shown highlighted in green on the invariant region of $\mathscr{F}$ in Figure 4.5, whereas $(\mathscr{B}, -1, \mathscr{H}), (\mathscr{B}, -1, \mathscr{B}) \in E_{\text{MA}}$ both have guard sets $g_e = \text{co}\{v^1, v^2\} \setminus \{(0, 0)\} = \{0\} \times [-v^*, 0)$. The reset and initial conditions are constructed according to their definition.

The following result shows that the MA design is well-posed.

Lemma 4.7.2. *The double integrator MA satisfies Assumption 4.4.5.*

Proof. (i) We must show that for all $m \in M$, $0 \notin \Sigma_{\text{MA}}(m)$. This is clearly true since there is no edge in E_{MA} containing the label 0.

(ii) We must show that for all $e_1, e_2 \in E_{\text{MA}}$ such that $e_1 = (m_1, \sigma, m_2)$ and $e_2 = (m_1, \sigma, m_3)$, $g_{e_1} = g_{e_2}$. This is clearly true since we have designed $g_{(\mathscr{F}, 1, \mathscr{H})} = g_{(\mathscr{F}, 1, \mathscr{F})}$ and $g_{(\mathscr{B}, -1, \mathscr{H})} = g_{(\mathscr{B}, -1, \mathscr{B})}$.

(iii) We must show that for all $e_1, e_2 \in E_{\text{MA}}$ such that $e = (m_1, \sigma_1, m_2)$ and $e_2 = (m_1, \sigma_2, m_3)$, if $\sigma_1 \neq \sigma_2$, then $g_{e_1} \cap g_{e_2} = \emptyset$. This is trivially true since for all $m \in M$, $|\Sigma_{\text{MA}}(m)| < 2$.

(iv) We must show that for all $e_1, e_2 \in E_{\text{MA}}$ such that $e_1 = (m_1, \sigma_1, m_2)$ and $e_2 = (m_2, \sigma_2, m_3)$, $r_{e_1}(g_{e_1}) \cap g_{e_2} = \emptyset$. Using Assumption 4.4.5 (ii) above, we only need to check two cases, that is, $e_1 = e_2 = (\mathscr{F}, 1, \mathscr{F})$ and $e_1 = e_2 = (\mathscr{B}, -1, \mathscr{B})$. Both cases satisfy the condition because of the reset action on the first coordinate; for example, the first case gives $r_{e_1}(\{d\} \times (0, v^*]) \cap \{d\} \times (0, v^*] = \emptyset$.

(v) We must show that for all $e = (m_1, \sigma, m_2) \in E_{\text{MA}}$, $r_e(g_e) \subset I_{\text{MA}}(m_2)$. This is easily verified for all four edges in E_{MA}, for example, if $e = (\mathscr{F}, 1, \mathscr{H})$, clearly $r_{e_1}(\{d\} \times (0, v^*]) \subset I_{\text{MA}}(\mathscr{H})$.

(vi) We must show that for all $m \in M$, if $\Sigma_{\text{MA}}(m) = \emptyset$ then $I_{\text{MA}}(m)$ is invariant. We have that only $\Sigma_{\text{MA}}(\mathscr{H}) = \emptyset$. As can be seen in Figure 4.5, the closed-loop vector field does not allow trajectories to cross outside of $I_{\text{MA}}(\mathscr{H})$, and therefore for all $x_0 \in I_{\text{MA}}(\mathscr{H})$, and for all $t \geq 0$, $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(\mathscr{H})$.

(vii) We must show that for all $m \in M$, if $\Sigma_{\text{MA}}(m) \neq \emptyset$ then $I_{\text{MA}}(m)$ forces all trajectories to exit in finite time through some guard. Consider $\mathscr{F}$ with $\Sigma_{\text{MA}}(\mathscr{F}) = \{1\}$. As can be seen in Figure 4.5, for all $x_0 \in I_{\text{MA}}(\mathscr{F})$, there exists $T \geq 0$ such that for all $t \in [0, T]$, $\phi_{\text{MA}}(t, x_0) \in I_{\text{MA}}(\mathscr{F})$, and $\phi_{\text{MA}}(T, x_0) \in \{d\} \times (0, v^*]$. Since both $g_{(\mathscr{F},1,\mathscr{H})} = g_{(\mathscr{F},1,\mathscr{F})} = \{d\} \times (0, v^*]$, the assumption holds. A similar argument can be made for $\mathscr{B}$. □

Remark 4.7.3. *We noted in Remark 4.5.3 that Zeno executions do not arise for reach-avoid specifications that, by construction, involve only finite MA executions. However, one may be interested in analyzing whether an MA is non-Zeno in its own right, independently of the high level plan or control specification for which it is used. It can be verified rather easily that the $p = 1$ double integrator MA design we have presented above is non-Zeno. The situation is considerably more complicated when considering an MA that is a parallel composition of these MA's or when considering an arbitrary MA. Generic conditions when hybrid systems have a Zeno execution have been studied in [54, 121, 5]. However, further study of this problem is needed in our context since existing results do not apply to all the situations that can arise in our MA.*

4.7.3 Double Integrator: Multiple Speed Levels

This section presents a natural extension of the canonical motion primitives *Hold*, *Forward*, and *Backward*, again for a double integrator model. The previous design is augmented with additional motion primitives causing forward or backward motion at various speed levels.

As in the previous section, we consider the dynamics (4.16). The state space is $Q_{\text{MA}} = M \times \mathbb{R}^2$. Let $n_s > 0$ be an integer denoting the selected number of speed levels. Then the set of motion primitives is

$$M = \{\mathscr{H}, \mathscr{F}_1, \mathscr{B}_1, \mathscr{F}_2, \mathscr{B}_2, \ldots, \mathscr{F}_{n_s}, \mathscr{B}_{n_s}\}. \tag{4.17}$$

The motion primitive $m = \mathscr{H} \in M$ is *Hold* as before and has exactly the same implementation. The motion primitives $m = \mathscr{F}_i \in M$, for $i = 1, \ldots, n_s$ are called *Forward at speed level i* and cause state trajectories to exit through the right face of Y^* in finite time. Similarly, the motion primitives $m = \mathscr{B}_i \in M$, for $i = 1, \ldots, n_s$ are called *Backward at speed level i* and cause state trajectories to exit through the left face of Y^* in finite time. The motion primitive $\mathscr{H}$ can be equivalently considered to be *Forward at speed level 0* and *Backward at speed level 0*, so it is convenient to define $\mathscr{F}_0 := \mathscr{B}_0 := \mathscr{H}$.

The set of labels is $\Sigma = \{-1, 0, 1\}$. See Figure 4.9 for a depiction of the edges when $n_s = 2$. For the context of parallel composition, we have shown the set of augmented edges of the MA, $\overline{E_{\text{MA}}}$; the set of

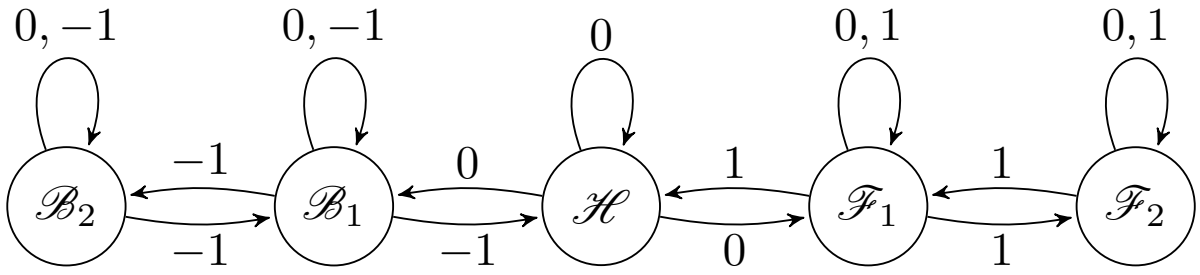

Figure 4.9: The augmented edges $\overline{E_{\text{MA}}}$ for the multi-speed motion primitives $n_s = 2$.

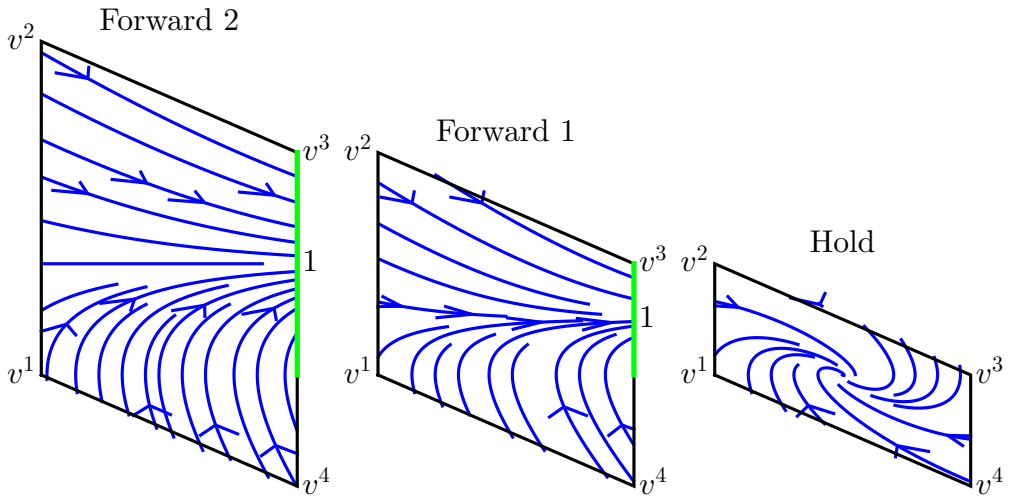

Figure 4.10: The closed-loop vector fields for the *Hold*, *Forward at speed level 1*, and *Forward at speed level 2* motion primitives over their respective invariant regions in the (x_1, x_2) axes.

edges E_{MA} consist of those edges $(m, \sigma, m') \in \overline{E_{\text{MA}}}$ such that $\sigma \neq 0$.

Speed levels in the forward direction $\mathscr{F}_i$, $i = 1, \ldots, n_s$ are connected in a chain such that upon crossing the right face of Y^* (associated to the event $1 \in \Sigma$), the next motion primitive may be at the same speed level, one higher, or one lower. This incremental change in speed is enforced for "smoother" transitions between the speed levels. Observe that slowing down one speed level from $\mathscr{F}_1$ results in the motion primitive $\mathscr{F}_0 = \mathscr{H}$. Similarly, the speed levels in the backward direction $\mathscr{B}_i$, $i = 1, \ldots, n_s$ are connected in a chain with $\mathscr{B}_0 = \mathscr{H}$ and events $-1 \in \Sigma$ corresponding to crossing the left face of Y^*.

As before, the design is scalable by the segment length $d > 0$ and maximum stabilizing control value $u^* > 0$, and we have the derived parameters v^*, k_1, and k_2. The construction for $m = \mathscr{H} \in M$ is the same as before and is not repeated. Now we give the construction for $m = \mathscr{F}_i \in M$, $i = 1, \ldots, n_s$. The design is inspired by *Forward* from the previous section, but follows an alternative approach to using the RCP. Essentially, we define a feedback controller that stabilizes all trajectories to a positive speed that is a multiple of v^*, namely $(i/2)v^*$. Visually, if the invariant regions $I_{\text{MA}}(\mathscr{F}_{n_s}), \ldots, I_{\text{MA}}(\mathscr{F}_1)$ were to

be stacked from left to right, we obtain a pyramid, see Figure 4.10. For all $i \in \{1, \ldots, n_s\}$, $I_{\text{MA}}(\mathscr{F}_i)$ is the convex hull of the four vertices $v^1_{\mathscr{F}_i} = (0,0)$, $v^2_{\mathscr{F}_i} = (0, (i+1)v^*)$, $v^3_{\mathscr{F}_i} = (d, iv^*)$, and $v^4_{\mathscr{F}_i} = (d, -v^*)$, and excluding the points $(0,0)$ and $(d,0)$ as before. The feedback controllers are

$$u_{\mathscr{F}_i}(x) = K_{\mathscr{F}_i} x + g_{\mathscr{F}_i} = \begin{bmatrix} 0 & k_2 \end{bmatrix} x + iu^* = k_2(x_2 - (i/2)v^*). \tag{4.18}$$

In this case, since there is no feedback on x_1, it can be verified that the closed-loop system asymptotically stabilizes trajectories to the positive equilibrium speed $(i/2)v^*$. The gain k_2 was chosen to be the same for each $\mathscr{F}_i$ to ensure the same rate of convergence to the equilibrium speed. The motion primitives $m = \mathscr{B}_i \in M$ are constructed similarly.

The enabling conditions, reset conditions, and initial conditions are straightforward to define. Finally, similarly to Lemma 4.7.2, it is easy to establish that this MA satisfies Assumption 4.4.5.

4.8 Quadrocopter Applications

In this section we apply our methodology to a collection quadrocopters. We first explain how motion primitives can be applied to the system, how to specify the reach-avoid objective, and the overall solution pipeline. Next, we compare and contrast three algorithms for computing a control policy. Then we present experimental results on three different scenarios. Lastly, we provide a discussion.

4.8.1 Interfacing Multiple Quadrocopters

First we recall the modelling and control of a single quadrotor presented in Section 2.7. The standard quadrotor dynamical model has six degrees of freedom, which can be described by the inertial linear positions (x_w, y_w, z_w) and the roll-pitch-yaw Euler angles (ϕ, θ, ψ). Rather than designing a maneuver automaton directly on this complex nonlinear system, we exploit the well known cascaded controller architecture and the observed relationship between the linear accelerations and rotation matrix (2.4). This enables us to use the motion primitives from Section 4.7.2 independently in the (x_w, y_w, z_w) directions to compute the linear accelerations as a feedback on the linear position and velocity states. This forms our position control module, which contrasts from most approaches which either design timed reference trajectories [73] or shape the velocity profile and then take the derivative [122]. Our approach shapes the acceleration profile, which provides more flexibility in incorporating actuation constraints through the selection of the maximum acceleration u^*. Implementation details on the attitude control module are given in Section 4.8.3.

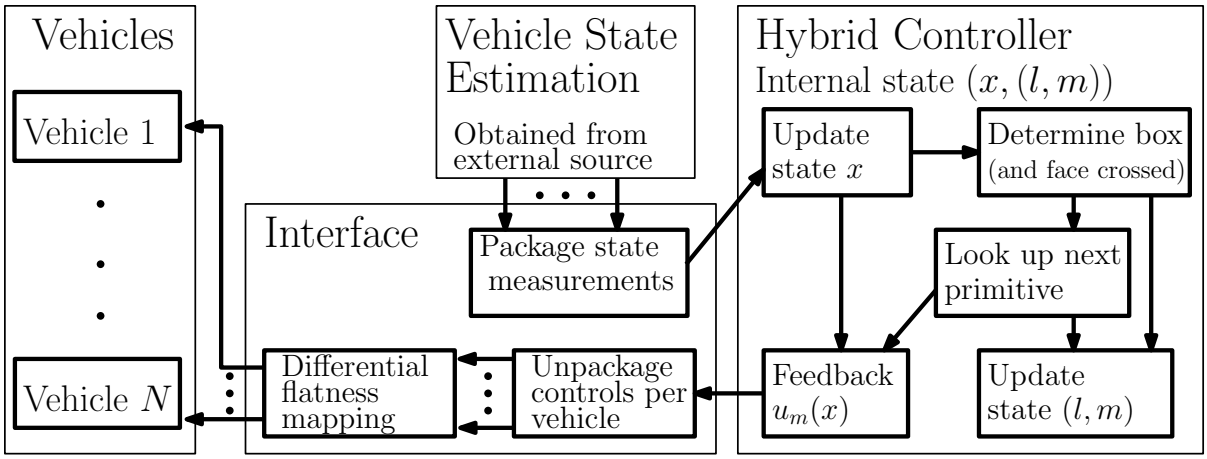

Figure 4.11: Interface between multiple vehicles and the framework with $p = 3N$ outputs. The hybrid controller internal state consists of the joint state measurement of all the vehicles and includes the current (joint) box, l, and the current (joint) motion primitive, m. The internal state is updated via external state measurements (assumed to be given) and is used to compute the feedback controls.

We consider a centralized reach-avoid objective among N quadrocopters. A copy of the gridded 3D workspace must be associated with each vehicle, resulting in a total of $p = 3N$ outputs. The p-dimensional MA representing the asynchronous motion capabilities of the multi-vehicle system is obtained by parallel composing p times the single-output MA from Section 4.7.2.

To specify the reach-avoid objective, we must identify the obstacle and goal boxes in $p = 3N$ dimensions. First we assume that the physical obstacles and goals for each vehicle are labelled on the physical 3D grid. Obstacle boxes in the output space correspond to any vehicle occupying a physical obstacle box or any two or more vehicles occupying the same physical box simultaneously. To avoid the effects of downwash, we do not allow vehicles to simultaneously occupy boxes that are displaced only in the z_w direction. Goal boxes in the output space correspond to all the combinations of individual vehicle 3D goal boxes. For simplicity, we assume that each vehicle has a single 3D goal box.

The multi-vehicle reach-avoid problem is solved offline using our proposed methodology. The runtime workflow is depicted in Figure 4.11. Each runtime component requires negligible computation, even for a large number of vehicles and outputs.

4.8.2 Control Policy Generation

We highlight three options for generating a control policy in the context of the multi-vehicle reach-avoid problem. For each, we give some implementation details and analyze its computational complexity. These are then compared in the experiments.

Exhaustive Non-Deterministic Dijkstra (NDD)

The first strategy follows the proposed methodology of Section 4.4. We highlight our main implementation steps. First, we compute the OTS states and edges for the associated output space obstacle boxes described earlier. Second, the p times parallel composed MA states and edges are computed. Third, the PA states and edges are computed. Fourth, the value function V is computed using (4.5). This is done by initializing the value function to be zero at goal states and infinite elsewhere, and then propagating backwards along PA edges using a non-deterministic Dijkstra (NDD) algorithm [17, 116]. Once the value function is computed at all states, we compute the optimal control policy $c^\star$ using Corollary 4.4.13. The initial PA states (4.7) correspond precisely to those states $q \in Q_{\text{PA}}$ with $V(q) < \infty$.

The computational complexity grows exponentially as the number of inputs $p = 3N$ increases. Suppose that the physical grid has (n_x, n_y, n_z) boxes in the (x_w, y_w, z_w) directions. Since there are 3^p motion primitives, the number of PA states is bounded by $|Q_{\text{PA}}| < (n_x n_y n_z)^N 3^p := k_1$. The number of edges from an OTS state is bounded by $3^p - 1$ (the neighboring directions), whereas the number of edges from a MA state is bounded by $(2^p - 1)3^p := k_2$ (the neighboring directions times the possible next motion primitives). Since the MA neighboring directions are more restrictive, we have the number of PA edges is bounded by $|E_{\text{PA}}| < k_1 k_2$. The presence of obstacles can dramatically reduce the number of PA states and edges. The NDD algorithm generally must inspect all the PA states and edges to compute the value function. As a result, it is optimal and complete (with respect to the selected grid resolution and motion primitive capabilities), which results in the largest possible set of initial conditions $\mathcal{X}_0$.

Deterministic A*

In this strategy, we make two simplifying assumptions to compromise the quality of the control policy in exchange for better computational efficiency. First, we take the p times composed MA and prune out motion primitives enabling simultaneous motion. Second, we forego computing the largest possible set of initial conditions and instead assume that a single physical initial box is specified for each vehicle. As such, it is sufficient to compute a single path of boxes in the OTS connecting the initial and goal boxes in the $p = 3N$ dimensional output space. From this path the control policy is immediately extracted, by assigning to each box the unique motion primitive leading to the next neighboring box along the path. The path is computed using a standard A* algorithm [65], which starts from the initial box and propagates outwards until the goal box is reached. The (admissible) heuristic function is chosen to be the Manhattan distance, which is the sum of distances along each output direction from the current box to the goal box.

The number of nodes that A* must investigate is bounded by the maximum number of OTS boxes, $(n_x n_y n_z)^N$, which still has exponential complexity in the number of robots. The pruned MA has $2p+1$ motion primitives, corresponding to $\mathscr{F}$ or $\mathscr{B}$ in a single output component with $\mathscr{H}$ elsewhere, plus the motion primitive $(\mathscr{H}, \ldots, \mathscr{H})$. Thus from the current box, we must check the $2p$ neighboring directions to select a feasible direction, taking into account out-of-bounds and obstacle configurations. In this implementation, the OTS, MA, and PA serve more as conceptual constructs, and do not need to be precomputed explicitly as it is expensive. In the worst case, the A* algorithm may investigate all boxes; as a result, it also produces a control policy that is complete and optimal with respect to the chosen grid and pruned MA motion capabilities. The policy produced by A* is of minimal length, but may have a long runtime execution.

Deterministic Greedy Search

This strategy also makes use of the two simplifying assumptions as with A* above, but differs in how the path is constructed. In greedy (best first) search [65], the path is constructed by starting from the initial box in the output space and then extending it from the current box into any feasible neighboring direction that decreases the Manhattan distance to the goal box. Greedy search can often find a path very quickly, although not necessarily an optimal one. Moreover, since greedy search may fail to find a path, it is not complete.

4.8.3 Experimental Results

Our experimental platform is the Crazyflie 2.0; see Figure 4.1. We used a VICON motion capture system to obtain the state estimates of the vehicles. For the attitude control module, we assume a zero yaw reference and we adapted the attitude controller of [73] onboard, making the crude simplification of zero desired angular velocities due to communication bandwidth constraints. Our implementation was done in Python 2.7.10 and ROS Kinetic, and computations were performed on a 64-bit Lenovo ThinkPad with an 8 core 3.0 GHz Intel Xeon processor and 15.4 GiB RAM.

We illustrate three different scenarios and consider the three policy generation strategies on each of them. The corresponding video results are available at `http://tiny.cc/modular-3alg`, and are able to convey the multi-vehicle motion much more effectively than the static plots shown here. In our code implementation, the user is able to easily specify any scenario by selecting the number of vehicles, the grid parameters, the obstacle locations, the goal locations, and other parameters such as the algorithm to generate a control policy.

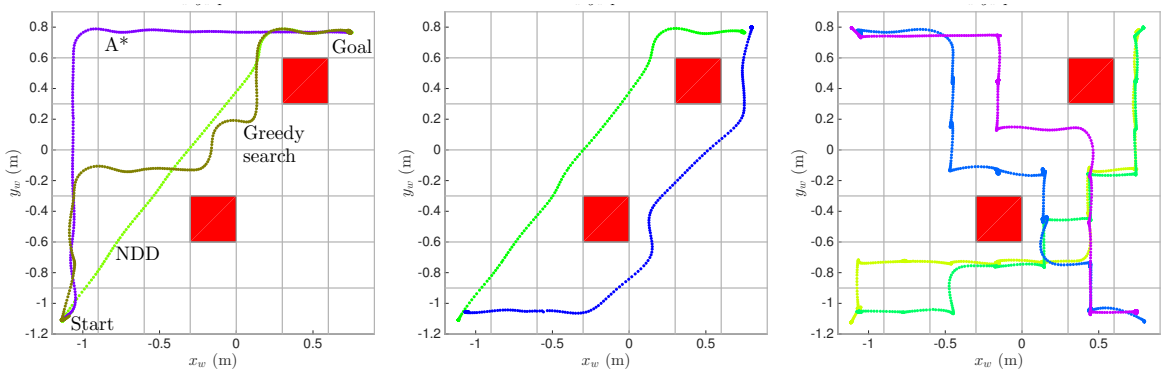

Figure 4.12: Experimental results for the open scenario, projected onto the (x_w, y_w) plane. In all plots, all the vehicles must swap corners of the room. The left plot compares trajectories for a single vehicle using the three different control policy generation strategies. The middle plot shows the resulting trajectories for two vehicles using non-deterministic Dijkstra. The right plot shows the resulting trajectories for four vehicles using greedy search, see also Figure 4.1. Although difficult to depict, the maneuvers are safe, as the trajectories do not occupy the same physical boxes at the same time.

Open Space

The first representative scenario involves an open 3D space partitioned into a $7 \times 7 \times 2$ grid and a sparse collection of pillar-shaped obstacles. The left plot of Figure 4.12 compares the resulting 3D trajectories in the (x_w, y_w) plane for the three strategies in the case of a single vehicle. The computation times were 40.63 milliseconds, 1.59 milliseconds, and 0.27 milliseconds for NDD, A*, and greedy search, respectively. The NDD algorithm offers the best quality control policy in that there is simultaneous motion in the different degrees of freedom whenever possible and the same policy can be used from any starting box. The A* and greedy search algorithms offer similar results to each other, with both producing an optimal path of length 14. Both yield less efficient grid-like motion that is defined only along a single path from the initial box, although a new policy can quickly be recomputed from different starting boxes. Based on simulation tests for a single vehicle, each of these algorithms scale well to larger spaces or finer grids; even NDD is able to compute a solution on a $100 \times 100 \times 10$ grid in about two minutes in the worst cases. Next we compare each strategy on more vehicles.

The middle plot of Figure 4.12 shows the resulting trajectories for two vehicles using NDD. The control policy was computed in about 18 minutes and is defined on about PA 180000 states. While the resulting control policy yields highly efficient motion defined over a large set of initial conditions, adding more vehicles or more boxes generally explodes the computation time and memory requirements. Thus NDD is best suited for small scenarios involving a modest number of vehicles, when one can afford to spend time precomputing the control policy.

The right plot of Figure 4.12 shows the resulting trajectories for four vehicles swapping corners of

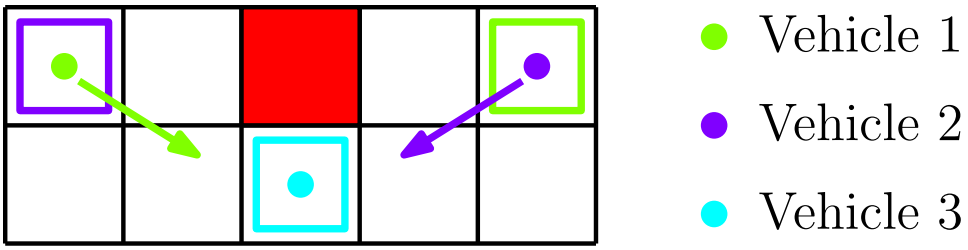

Figure 4.13: This figure shows the channel swapping experiment involving three vehicles. In particular, it shows the specification for the first reach-avoid, where goals are shown as the colored boxes. For the second reach-avoid, the initial and goal boxes of vehicle 1 and 2 are swapped.

the room using greedy search. Since the vehicles and physical obstacles occupy a single box, greedy search performs well, as each action typically results in one vehicle making progress towards the goal. The computation time was about four milliseconds. Simulation results on a $100 \times 100 \times 10$ grid with eight vehicles placed randomly demonstrate that greedy search is usually able to find a solution on the order of one second. As one would expect, greedy search typically fails to find a solution if long wall-like or non-convex obstacles are introduced, or if the goals are not spaced out sufficiently. Furthermore, the time to execute the entire maneuver scales with the number of vehicles.

Finally we consider the deterministic A* algorithm. Although the resulting trajectories follow a path of optimal length, they look quite similar to those found by greedy search and thus are not shown. Moreover, the method quickly becomes more computationally expensive beyond three vehicles.

Channel Swapping

The second representative scenario involves two rooms connected by a channel, defined over a $5 \times 2 \times 1$ grid, see Figure 4.13. Two of the vehicles must continually swap places, while the third is required to act as a gatekeeper. We specify this objective as an infinitely looping sequence of two distinct reach-avoid problems. This illustrates that reach-avoid is a useful building block for addressing more complex specifications.

The NDD algorithm produced both control policies in about 10 seconds, while the A* algorithm took about 0.03 seconds. Greedy search fails to find a solution because it is unable to coordinate the third vehicle away from its goal to make space for the other two. Since the resulting trajectories overlap in physical space, Figure 4.14 shows the trajectories as a function of time using the policy computed with NDD. The trajectories are highly non-trivial, but show that the objective is satisfied for at least one cycle of both reach-avoids. Although not shown, the trajectories computed using A* are similar but take a few seconds longer to execute the objective since the motion primitives are deterministic.

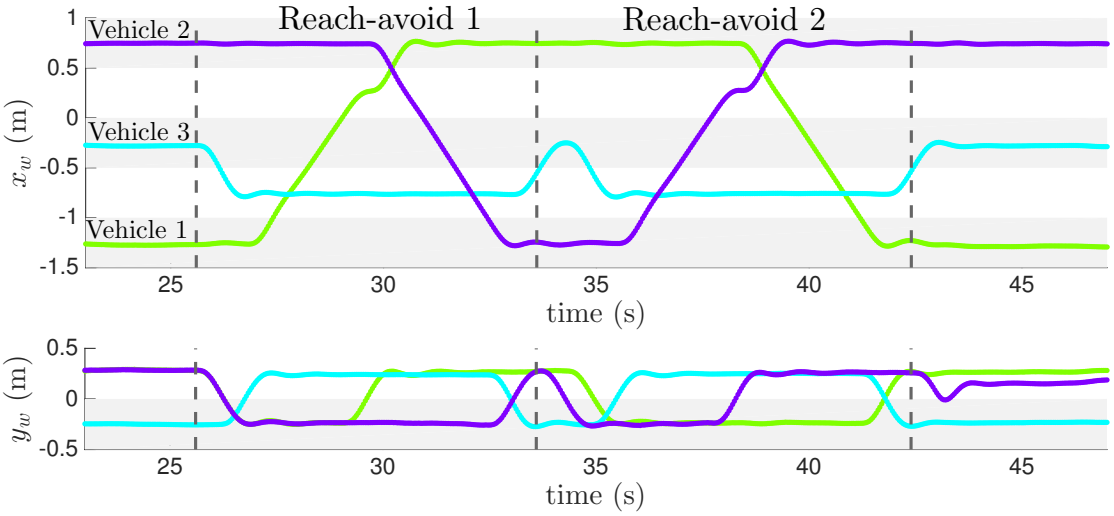

Figure 4.14: Trajectories using non-deterministic Dijkstra for the channel experiment. The alternating grey and white areas reflect the size of the grid boxes. The duration for each of the two reach-avoid specifications is highlighted.

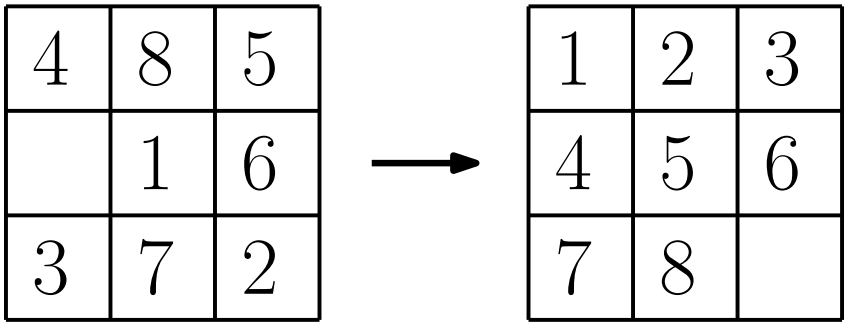

Figure 4.15: The 8-puzzle: eight vehicles must coordinate into the ordered configuration.

8-Puzzle

We conclude our experimental results with the well-known 8-puzzle. On a $3 \times 3 \times 1$ grid, eight vehicles are placed randomly and must return to an ordered configuration, see Figure 4.15. For this application, the A* algorithm is the most suitable, computing the control policy in 0.32 seconds. The NDD approach would spend too much time precomputing edges in the high dimensional output space, while greedy search would never make progress. Results are available to view in the video.

4.8.4 Discussion

Throughout the various experimental scenarios presented, we have demonstrated the modularity offered by our approach. The designer can customize their own algorithms for generating a control policy in order to trade-off solution quality with computational efficiency. Depending on the specific application scenario, a different control policy generation strategy may be more suitable.

In our analysis, all of the complexity was associated with the generation of the control policy for a given MA. The MA formalism enables us to generate control policies with no further regard to the continuous time trajectories that may result, due to the guarantees on discrete behavior encoded in the MA edges. On the other hand, the generation of a MA for an arbitrary system is a difficult challenge in its own right and is left to the discretion of the designer, although the design we have presented in Section 4.7.2 can potentially be applied to control systems that are feedback-linearizable into a collection of double integrators. Taking care that the outputs are translationally invariant and that obstacle boxes can be computed, this includes end effector control of fully actuated robotic manipulators [100] and some wheeled vehicles through the use of look-ahead points [4].

It is important to emphasize that design of the MA decouples the dynamics of the system from the reach-avoid specification, which is embodied by the OTS. As such, given a MA design that is well-posed, we may apply it to a variety of physical scenarios, including also a sequence of reach-avoids. For more complex logical specifications, such as those expressed in temporal logic, it is possible to combine our methodology with other approaches such as in [58]. Informally, the temporal logic formula would be converted to an automaton in the usual way, which can then be producted with our Product Automaton to yield the possible motions over the grid augmented with states monitoring the progress of satisfying the formula.

We also emphasize the role of non-determinism. In such a situation, motion primitives can enable multiple possible faces of an output space box to be reached, which physically corresponds to vehicles moving at the same time and/or a single vehicle moving in multiple directions in 3D. Motion primitives enabling such non-determinism are more rich and complex, affording more efficient motions and execution times, but at the expense of more computational complexity.

Our approach offers robustness through the use of feedback-based motion primitives, as the construction of invariant regions ensures a wide range of initial conditions for which output trajectories exit through appropriate guard sets into subsequent boxes. Since the motion primitives are updated during execution based on the measured box transitions and control policy, we do not require timing estimates for completing box transitions, which can be difficult to compute. These features are advantageous under model uncertainty, which we must contend with since we base our motion primitive design on the double integrator model rather than the more complex quadrocopter model, and since aerodynamic effects arise when multiple quadrocopters fly in close proximity. Our previous work also demonstrated similar robustness of operation under wind disturbances generated by a fan on a larger quadrocopter [109]. Finally, we note that our framework can easily be applied to a heterogenous team of robots; if each vehicle has its own MA, the parallel composition automatically constructs the overall MA for the

multi-vehicle system.

Of course, our solution to Problem 4.3.1 is conservative because we have restricted ourselves to a particular discretization, namely the choice of a partition into boxes and the use of motion primitives. As we have demonstrated, this is a reasonable trade-off, especially since the resolution of the output space discretization, the richness of motion primitives, and the complexity of the control policy are all design parameters. Moreover, by utilizing a richer set of motion primitives, such as the multispeed primitives proposed in Section 4.7.3, or by decreasing the grid size, we can effectively increase the resolution or space of possible trajectories, thus decreasing conservatism but with the burden of additional computational effort.

4.9 Conclusion

We have developed a modular framework for motion planning of multiple agents in known environments. It consists of several modules. An output transition system (OTS) models the allowable motions of the agents by partitioning their workspace into boxes. A set of motion primitives is designed based on reach control on polytopes. A maneuver automaton (MA) captures constraints on successive motion primitives. Finally, a control policy is generated based on the synchronous product of the OTS and the discrete part of the MA. Overall we obtain a two-level control design which is highly robust, modular, and conceptually elegant. We presented a specific maneuver automaton for the double integrator system, and we showed how this design can be composed to obtain maneuver automata for multi-agent systems. The methodology was experimentally validated on a collection of quadrocopters.

We briefly discuss some of the limitations and possible extensions on the methodology presented in this chapter. First, although we have presented a fairly general framework for nonlinear systems with symmetries, it can still be argued that such an approach can be difficult to apply directly to an arbitrary real robotic system in practice. In retrospect, the well-posedness conditions we have supplied are fairly intuitive and perhaps it would have been more useful to focus on providing an automated procedure for constructing a maneuver automaton that satisfies the required properties by design. Although we have sketched some possible ways to apply our existing motion primitives based on integrators to other robotic systems, a customized approach likely needs to be employed on different robotic platforms. Even in our application to quadrotors, we employed the known cascaded control architecture and differential flatness, as designing motion primitives directly on the full 12-dimensional system would have been a difficult challenge. Moreover, the proposed framework was applied to two planar robotic manipulators in [61], and it was observed that the computation of obstacles was much more challenging than for quadrotors.

We also note that our formulation of motion primitives was given on $\mathbb{R}^n$ rather than on a manifold; the original formulation of the MA in [33] potentially provides some tools to extend our feedback-based motion primitives to manifolds. Finally, our approach would not be suitable for objectives expressed on outputs that are not translationally invariant; for example, a flip of a quadrotor would require explicitly controlling an angular state.

In particular, application to a unicycle model is also left as future work, and we briefly outline two possible directions. One approach is to work in the full state space (2D position and heading) and consider various linear speeds and angular speeds to form suitable invariants and enabling conditions as required by the MA. Special care would be required as the heading angle means that the state is on a manifold, as noted above. A second approach is to employ the standard technique of a look-ahead point [4], in which it is easy to show that the unicycle model becomes two single integrators so that the design in Section 4.7.1 can be used. In both these approaches, the difficulty of modelling obstacle boxes in the output space presents a challenge, as collision checks with the environment would typically require knowledge of the heading angle as well. Conservative modelling of obstacle boxes may prohibit application to tight maneuvers such as parking. Ultimately, the application of our proposed method on a variety of different robotic systems in the future will ascertain its true limitations.

Second, our framework has been designed specifically on a uniformly gridded output space. While this feature has many advantages such as being simple and compatible with many planning algorithms, it is still somewhat of an arbitrary imposition. The next chapter aggregates sequences of motion primitives to obtain more general motion primitives of varying shapes, but it is still confined to a grid. It may also be interesting to investigate to what extent feedback-based motion primitives can be formulated on other types of partitions; some degree of symmetry is likely required, for otherwise there would be too many different motion primitives to be used during high level planning.

Third, in this chapter we employed fairly naive high level planning approaches. Given the number of advanced planning algorithms discussed in the related literature, it might have been worthwhile to select some compatible algorithms or devise new algorithms that could really showcase scalability to a large group of simultaneously moving vehicles as well as fast computation times. A decentralized approach is promising, where each agent computes its own control policy and models the others as moving obstacles. Fortunately, the existing motion primitives could be reused, and the control policy can possibly be recomputed online as the obstacles are updated. However, formal guarantees on completion of the objective are more difficult to establish. In terms of a decentralized or distributed computation of a control policy, it would also be interesting to incorporate communication constraints, which could be done at a discrete resolution based on the box sizes so as to assign motion primitives to maintain

connectivity on fixed or variable communication topologies [75]. These directions were not pursued, as the emphasis of the work was on establishing the correctness of integration between the high and low levels. Moreover, the next chapter shows that the addition of layers in the hierarchy is a promising approach to dealing with complexity.

Fourth, this work has focused on many technical points, and a particularly challenging aspect was the consideration of Zeno executions. We made extensive efforts to find sufficient conditions that would establish the absence of Zeno executions in maneuver automata, but did not succeed. Even the consideration of Zeno executions for our integrator-based motion primitives was non-trivial, and becomes much more complex during the parallel composition procedure. While such details are unlikely to concern the majority of the robotics community, a complete characterization of the existence of Zeno executions is an open problem in our maneuver automaton framework as well as the general hybrid systems literature.

Finally, we have focused the work on the relatively simple reach-avoid problem. As we noted in the related literature, some works have focused on more interesting real-world scenarios involving more complex actions other than just movement as well as dynamic tasks and environments. Extending our method to such scenarios would require designing motion primitives that encode completion of actions, and defining a suitable notion of a high level control policy in the face of uncertainty.

Chapter 5

Hierarchical Motion Primitives for Motion Planning

5.1 Introduction

This chapter proposes a hierarchical architecture for motion primitives for motion planning of multi-vehicle systems. There are at least three benefits conferred to the motion planning problem by a hierarchical approach. First, it allows the user to incrementally build up arbitrarily complex motion primitives from simpler ones in a systematic, rigorous, yet intuitive way. Second, a hierarchical architecture provides a way to dramatically reduce the overall complexity of motion planning by abstracting details at the low level, while at the same time not compromising on correctness and safety. The abstraction of low level behavior is obtained by aggregating collections of behaviorally similar lower level primitives, whose details need not be distinguished in high level motion planning algorithms. Finally, hierarchically constructed motion primitives can be used to constrain behavior. In particular, they can be used to characterize the aggregate behavior of a group of vehicles so that, for example, a formation is retained.

To illustrate our vision of hierarchical motion primitives, consider a control system with two outputs, say to model the planar motion of a vehicle. Suppose that the two-dimensional output space has been gridded into boxes and that we have devised feedback controllers to obtain a finite set of *atomic motion primitives*. In Figure 5.1 we show five atomic motion primitives, *Hold*, *Right*, *Up*, *Right-Up*, and *Down*. These atomic motion primitives at level 0 are responsible to drive the output trajectories of the system from one box to another. Motion primitives at level 1 are formed from sequences of level 0 motion primitives. For example, two successive *Right* atomic motion primitives from level 0 form the *Two*

Right motion primitive at level 1. Similarly, level 2 motion primitives can be built using level 1 motion primitives. We will demonstrate that planning can be made dramatically more efficient.

Though our main idea is simple, and while many concepts and methods involving hierarchy and abstractions have been well studied [3, 103, 20, 53, 119], we place extra demands on our design that take it a step beyond what has been done. First, we allow for an arbitrary number of hierarchical levels, and the design of any level depends in the same way only on the level below. One of the benefits of this uniformity of the architecture is that a designer can apply a planning algorithm at any level to obtain a control synthesis to the problem. Second, motion primitives at any level must be implementable by the low level continuous dynamics, for instance by adhering to safety and continuity constraints on positions and velocities. Effectively, we are implementing a notion of hierarchical consistency, but with the addition of a continuous state feedback at the lowest level [53]. As a final constraint, we expect a design that gives dramatic computational advantages, especially in the multi-agent setting. To this end, we construct hierarchical motion primitives on a grid, as shown in Figure 5.1.

There are three main contributions of this chapter. First, we supply the complete architectural formulation of hierarchical motion primitives, which involves careful attention to suitable, parsimonious data structures and relationships. Second, we introduce and solve a novel formulation of the motion planning problem called *behavior-constrained reach-avoid*, in which the vehicles must safely reach goal locations while maintaining desirable predefined sequences of motions on the grid; here our analysis employs fairly standard notions from hybrid control theory. Finally, we supply a library of hierarchical motion primitives for multi-agent systems and apply these motion primitives to a collection of quadrocopters to achieve a new way to perform formation flight as well as to morph between formations in obstacle rich environments. The effectiveness and scalability of our approach is experimentally demonstrated on a variety of scenarios. A video illustrating these results can be found at `http://tiny.cc/hier-moprim`.

This research is an outgrowth of our modular framework for motion planning of nonlinear systems with symmetries based on motion primitives presented in the previous chapter; the notions of atomic motion primitives and the *maneuver automaton* were introduced there. As those constructions were limited only to level 0, this chapter concerns itself with describing how to build hierarchical motion primitives at higher levels and illustrating the computational advantages of employing a multi-level hierarchy. Moreover, we have introduced the notion of behavior constraints in addition to the usual reach-avoid problem as a mechanism to address the formation control problem of quadrotors in a hierarchical manner.

This chapter is organized as follows. In the next section, we give an overview of related literature. In Section 5.3 we formulate the motion planning problem as a reach-avoid problem with behavioral

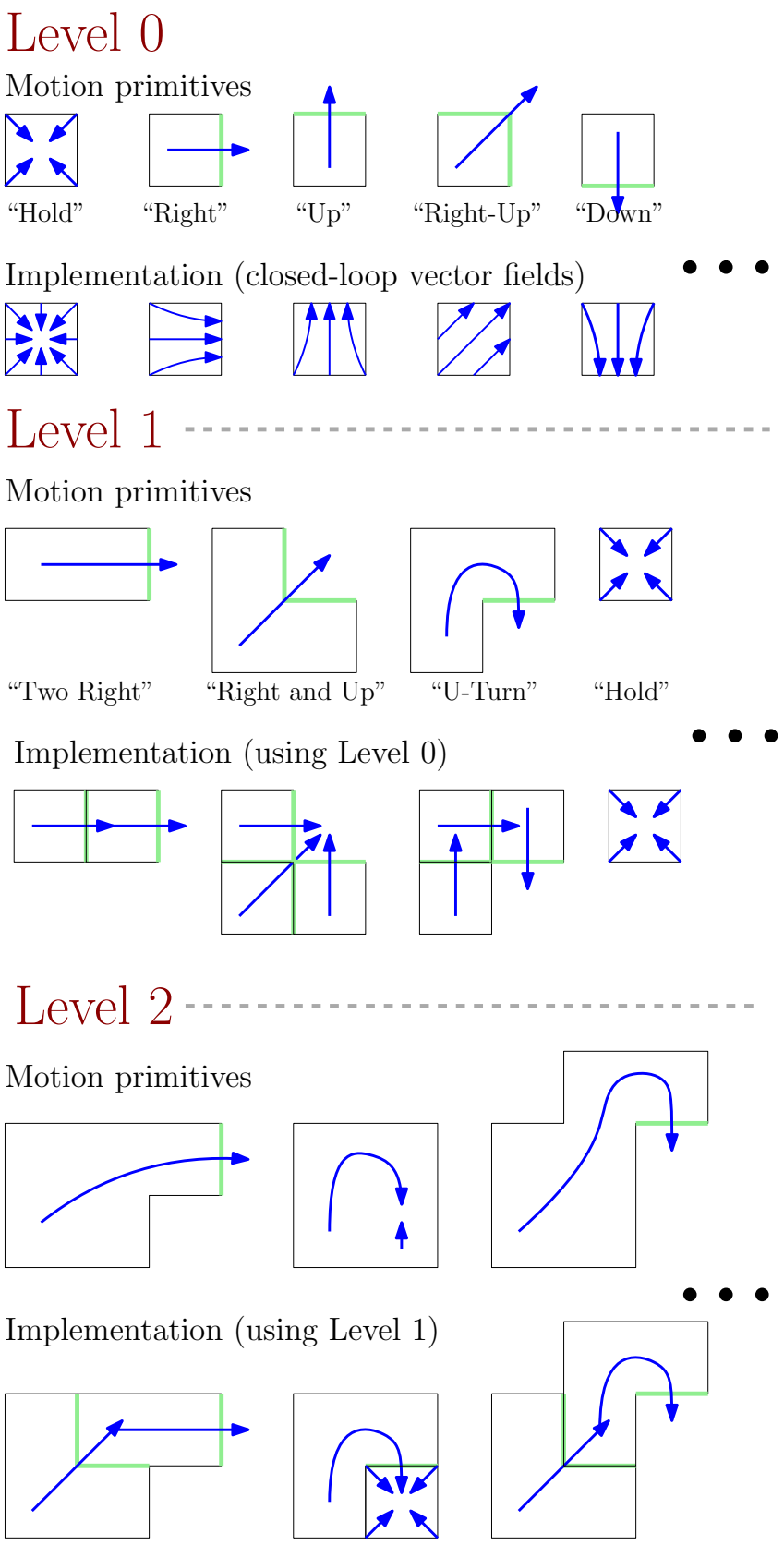

Figure 5.1: An example of motion primitives designed at three hierarchical levels over a gridded two dimensional output space.

constraints. The methodology of hierarchical motion primitives is given in Section 5.4. In Section 5.5, we apply our methodology to obtain a hierarchical controller and solvability conditions for the behavior-constrained reach-avoid problem. In Section 5.6 we discuss two formal methods for obtaining hierarchical motion primitives from simpler ones. In Section 5.7 we present a library of hierarchical motion primitives, while in Section 5.8 we apply them in the context of quadrocopter control and planning. Experimental results on quadrocopters are given in Section 5.9. Finally, we conclude in Section 5.10.

5.2 Related Literature

5.2.1 Hierarchy and Abstractions

The idea of using hierarchy to simplify and scale up system design is an intuitive and attractive concept that has been described in a variety of forms [74]. In the context of general control systems, several works have studied abstractions for discrete event systems [53, 119] and hybrid systems [3, 103, 20, 104]. While these theoretical constructions provide important insights into the process of forming hierarchical structures, application of these methods to real robotic systems remains a challenge. To this end, we provide additional constraints on the design of hierarchical motion primitives to ensure implementability.

Other theoretical works have employed similar techniques on various applications, such as using Petri nets with refinements to model manufacturing cells [124], using max-plus algebra to model railway scheduling [72], and using a multi-step process to synthesize hybrid controllers for linear temporal logic (LTL) specifications [31]. The recent work [35] has studied general factoring of hybrid systems and applied it to a robotic pick-and-place operation.

Many works that utilize hierarchy assume a fixed structure, typically with a high and low level, including our previous work [109, 111]. In our view, such fixed levels represent a missed opportunity, since exploiting a flexible hierarchical structure with an arbitrary number of levels may result in even better modeling efficiency. This chapter presents a concrete description of a multi-level hierarchy in the context of grid-based planning. It makes the relationships between the various levels explicit in order to establish correctness in motion planning.

5.2.2 Multi-agent Planning and Control

In the context of multi-agent motion planning, a number of hierarchical-based techniques exist to simplify planning. Comparable works include those that discretize the workspace into a grid, employ multi-resolution grid representations, and perform graph search [67, 56, 101, 49, 26, 107]. These include a

variety of multi-resolution grid techniques, such as terrain map clustering [49], map abstraction and refinement [101], and wavelet decompositions [26]. Similar techniques have been applied to not just to motion planning, but in combination with perception and terrain mapping [79]. Most of these works do not explicitly consider the dynamics of the agents and thus the feasibility of executing these efficiently constructed motion plans is uncertain. Since this chapter focuses on grid-based planning and integrates the continuous level explicitly, existing multi-resolution planning techniques may guide the construction of hierarchical motion primitives as presented here - this is an area of future investigation.

One of our motivating applications for hierarchical motion primitives is coordination of quadrotors. There is a wealth of literature on multi-agent formation control. Concepts from computational geometry and graph theory play a predominant role in multi-agent coordination and more specific tools such as rigidity theory have offered important theoretical insights [75, 19]. Various network topologies have been considered, in which agents make decisions based on absolute quantities in a common reference frame, relative quantities, or a mixture such as a leader-follower approach [81]. Abstractions of the overall group and control of aggregate features are considered in [57, 11]. A leader-follower scheme is approached as a constrained optimization problem in [102].

We formulate the formation control problem in a new way, taking the standard reach-avoid specification for a multi-vehicle system to safely reach a target in a known environment, and further imposing a suitably defined behavior constraint to maintain desired relative constraints among the vehicles. Behavior constraints find analogy with sublanguage specifications of discrete event systems [119], and they also have some resemblance with LTL specifications [58, 116]. We are not aware of any existing method that uses hierarchical motion primitives to solve the formation control problem. In particular, our approach enables to abstract away the number of vehicles, leading to efficient computations, but still precisely navigate a formation of vehicles, even allowing obstacles to pass tightly between the vehicles. Additionally, our method of formation control offers important practical advantages. Since it relies on discrete event sequencing rather than timed reference trajectories, communication delays play a relatively inconsequential role. Moreover, feasibility of the formation is guaranteed in a bottom-up fashion, which decouples the behavior satisfaction from the reach-avoid specification. Finally, our hierarchical theory offers simplicity in implementation as it involves mainly discrete computations rather than geometric arguments and nonlinear control theory. In general, the design of hierarchical motion primitives can be tailored by the designer to address specific behavior constraints, and a formation constraint represents just one special case within our proposed framework.

5.2.3 Motion Primitives

Motion primitives have become a popular tool in robotics for abstracting and simplifying possible motions in order to perform motion planning that is implementable by vehicle dynamics. Motion primitives have been generated and specified in a variety of ways on a number of applications, including predefined reference trajectories for quadrocopters [93, 43], segmentation and clustering of recorded trajectories for humanoid robots [97, 63], and sampled state lattice endpoints for cars-like robots [84]. Other works have also focused less on dynamics and more on describing hierarchies of actions and task decomposition [55].

Many authors have realized the importance of incorporating constraints when concatenating motion primitives. The concept of a maneuver automaton, which describes concatenation constraints among motion primitives, originates with [33]. Our formulation of the maneuver automaton in Chapter 4 was specialized to a grid partition in order to facilitate correctness of motion primitive concatenation in cluttered environments. This chapter extends these notions by introducing the hierarchical maneuver automaton. Other works have organized motion primitives and their concatenation constraints in the form of skill trees [60] and binary trees [27]. All of these approaches have the common goal of providing a hierarchical description of motion capabilities that is meaningful to the application considered.

In contrast to the above approaches, our work in Chapter 4 introduced a novel formulation of feedback-based motion primitives for motion planning of nonlinear systems with symmetries on an output space grid. Our formulation provided two important advantages: (i) a provable safety guarantee between the consistency of discrete planning using motion primitives and continuous time implementability, and (ii) a decoupling and modularity of the design of low level continuous time feedback controllers and high level discrete planning. This chapter provides a natural extension to design abstract motion primitives up to arbitrary levels, in particular to facilitate scalability of motion planning to a large collection of vehicles.

5.3 Problem Statement

Consider the general nonlinear control system

$$\dot{x} = f(x, u)\,, \qquad\qquad y = h(x)\,, \tag{5.1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^\mu$ is the input, and $y \in \mathbb{R}^p$ is the output. Let $x(\cdot, x_0)$ and $y(\cdot, x_0)$ denote the state and output trajectories respectively of (5.1) starting at initial condition $x_0 \in \mathbb{R}^n$ and under some open-loop or feedback control. As in Chapter 4, we make the standing assumption that the

outputs of (5.1) are a subset of the states and that the vector field is invariant to the value of the output. Refer to Assumption 4.3.2.

Fix a *grid length vector* $d = (d_1, \ldots, d_p)$ where $d_i > 0$. The gridded output space is constructed by translating the canonical box $Y^\star := \prod_{i=1}^{p}[0, d_i]$. That is, associated with each $l \in \mathbb{Z}^p$ is a shifted box $Y_l := \prod_{i=1}^{p}[l_i d_i, (l_i + 1)d_i]$. We call $l \in \mathbb{Z}^p$ the *box index* of Y_l. We identify a (non-empty) set of feasible boxes in terms of their box indices $L_f \subset \mathbb{Z}^p$. The feasible boxes arise from control specifications including obstacle avoidance, collision avoidance, communication constraints, and others. Similarly, we identify a (non-empty) set of goal boxes in terms of their box indices $L_g \subset L_f$.

Now consider an output trajectory $y(\cdot, x_0)$ for some control and initial condition $x_0 \in \mathbb{R}^n$. We associate to $y(\cdot, x_0)$ a discretized trajectory called a *run* that records the boxes that the output trajectory visits; see [58] for a formal definition. The run is denoted as $y_r := l_1 l_2 \cdots$, where $l_i \in \mathbb{Z}^p$ is a box index. We define the *behavior* induced by $y(\cdot, x_0)$ to be the sequence of box index increments $y_b := \kappa_1 \kappa_2 \cdots$, where $\kappa_i := l_{i+1} - l_i$. Because y_r records every box visited by the output trajectory, we have that $\kappa_i \in \{-1, 0, 1\}^p \setminus \{0\}$. That is, the increment in any coordinate is of magnitude at most 1, and the overall increment is never 0. Let $\mathcal{B}$ denote the empty sequence and all finite and infinite sequences on $\{-1, 0, 1\}^p \setminus \{0\}$. We define a *behavioral constraint* $\widehat{\mathcal{B}} \subset \mathcal{B}$ to be any non-empty subset of $\mathcal{B}$.

Problem 5.3.1 (Behavior-Constrained Reach-Avoid)**.** *Consider the system* (5.1) *with a gridded output space in terms of grid length vector* $d \in \mathbb{R}^p$*. We are given goal boxes* L_g*, feasible boxes* L_f*, and a behavioral constraint* $\widehat{\mathcal{B}} \subset \mathcal{B}$*. Find a feedback control* $u(x)$ *and a set of initial conditions* $\mathcal{X}_0 \subset \mathbb{R}^n$ *such that for each* $x_0 \in \mathcal{X}_0$*:*

(i) **Avoid***:* $y(t, x_0) \notin \mathbb{R}^p \setminus \left(\bigcup_{l \in L_f} Y_l\right)$ *for all* $t \geq 0$*,*

(ii) **Reach***: there exists* $T \geq 0$ *such that for all* $t \geq T$*,* $y(t, x_0) \in \bigcup_{l \in L_g} Y_l$*,*

(iii) **Behavior***:* $y_b \in \widehat{\mathcal{B}}$*.*

5.4 Hierarchical Motion Primitives

In this section we present the hierarchical construction of motion primitives. First we recall the definitions of maneuver automata (MA) and executions from Chapter 4 and recast it in a modified notation, as it serves as the level 0 or base case of the construction.

5.4.1 Maneuver Automaton

A *maneuver automaton* (MA) is a hybrid system whose discrete modes correspond to motion primitives. Each level 0 motion primitive has associated to it a continuous time closed-loop vector field obtained by applying a state feedback law to (5.1). The edges of the MA model feasible successive motion primitives. A motion primitive generates some desired behavior of the output trajectories of the closed-loop system over a box in the output space. Because of the uniform gridding of the output space into boxes and because of the symmetry assumption on the outputs, level 0 motion primitives are designed only over $Y^\star$. The MA applies state resets to ensure that output trajectories remain in $Y^\star$. Real, physical trajectories clearly do not undergo such resets and instead output trajectories move continuously from box to box.

Definition 5.4.1. *Consider the system* (5.1) *satisfying Assumption 4.3.2 and the box $Y^\star$ with lengths d. The* level 0 maneuver automaton *(0-MA) is a tuple $\mathcal{H}^0 = (Q^0, \Sigma^0, E^0, I^0, Q^{0,0}, X^0, G^0, R^0)$, where*

State Space $Q^0 = M^0 \times \mathcal{X}^0$ *is the hybrid state space, where M^0 is a finite index set of level 0 motion primitives, and $\mathcal{X}^0 = \mathbb{R}^n$ is the continuous state space.*

Labels $\Sigma^0 = \{(0, \kappa) \mid \kappa \in \{-1, 0, 1\}^p\}$ *is a finite set of event labels, where the first element 0 of the pair $(0, \kappa)$ identifies the starting box in terms of its box index in the gridded output space. Since all level 0 motion primitives are only defined on the box $Y^\star$, this index is always 0. The second element κ is an offset associated with a face of $Y^\star$. That is, $\kappa = (\kappa_1, \ldots, \kappa_p)$ identifies the face of $Y^\star$ given by*

$$\mathcal{F}_\kappa = \left\{ y \in Y^\star \;\middle|\; \begin{cases} y_i = 0, & \text{if } \kappa_i = -1 \\ y_i = d_i, & \text{if } \kappa_i = 1 \\ y_i \in [0, d_i], & \text{if } \kappa_i = 0 \end{cases} \right\}.$$

Edges $E^0 \subset M^0 \times \Sigma^0 \times M^0$, *is a finite set of edges.*

Invariants $I^0 : M^0 \to \mathcal{P}(\mathbb{R}^n)$ *assigns a bounded invariant set $I^0(m)$ to each $m \in M^0$ that defines the region on which the vector field $X^0(m)$ is defined. We impose that the projection of $I^0(m)$ into the output space lies in $Y^\star$; that is, $h(I^0(m)) \subset Y^\star$.*

Initial Conditions $Q^{0,0} \subset Q^0$ *assigns a set of initial conditions. Specifically, $Q^{0,0} = \{(m, x) \in Q^0 \mid x \in I^0(m)\}$.*

Vector Fields $X^0 : M^0 \to \{f^0_m\}_{m \in M^0}$ *assigns to each $m \in M^0$ a globally Lipschitz closed-loop vector*

field of the form $f_m^0(\cdot) = f(\cdot, u_m(\cdot))$*, where* $u_m(\cdot)$ *is the state feedback controller associated with* $m \in M^0$*.*

Enabling Conditions $G^0 : E^0 \to \{g_e^0\}_{e \in E^0}$ *assigns to each edge* $e = (m, s, m') \in E^0$ *a non-empty enabling or guard condition* $g_e^0 \subset I^0(m)$*. We require that the projection of* g_e^0 *into the output space lies in the face of* $Y^\star$ *determined by the label* $s = (0, \kappa)$*. That is,* $h(g_e^0) \subset \mathcal{F}_\kappa$*.*

Reset Conditions $R^0 : E^0 \to \{r_e^0\}_{e \in E^0}$ *assigns to each edge* $e = (m, s, m') \in E^0$*, a reset map* $r_e^0 : \mathbb{R}^n \to \mathbb{R}^n$*. Since resets of states are determined by the event* $s = (0, \kappa) \in \Sigma^0$ *in order to maintain output trajectories inside the canonical box* $Y^\star$*, we define* $r_e(x) = x - h_o^{-1}(d \circ \kappa)$*. In particular, this means that the* i*-th output component is reset to the right face of* $Y^\star$ *if* $\kappa_i = -1$*, reset to the left face if* $\kappa_i = 1$*, and unchanged otherwise.*

Remark 5.4.2. *For an edge* $e = (m, s, m') \in E^0$*, the label* $s = (0, \kappa) \in \Sigma^0$ *with* $\kappa \in \{-1, 0, 1\}^p$ *denotes that the starting box is* $Y^\star$ *and the next box is* Y_κ*. This edge* e *would be taken by a continuous output trajectory* $y(\cdot, x_0)$ *starting in* $Y^\star$ *using the motion primitive* $m \in M^0$*, then reaching a face shared by* $Y^\star$ *and* Y_κ*, and finally applying the motion primitive* $m' \in M^0$ *after the reset. Thus,* Σ^0 *describes all possible directions to neighboring boxes that are reachable with the current motion primitive starting from box* 0*. We will see that the sequence of labels generated are precisely what we call the* behavior *generated by an output trajectory of* (5.1)*.*

The semantics of a level 0 MA involve the standard notion of an *execution* of a hybrid system, as discussed in Chapter 4. We shall denote a level 0 execution as $\chi^0 = (\tau^0, m^0, x^0)$, where τ^0 is a *hybrid time domain* consisting of time intervals (e.g. $\tau^0 = \{[0, 2], [2, 3.5], \ldots\}$), m^0 is a sequence of level 0 motion primitives recording the motion primitive over each time interval, and x^0 is the continuous state as a function of time.

Definition 5.4.3. *A level* 0 hybrid time domain *is a finite or infinite sequence* $\tau^0 = \{\mathcal{I}_0^0, \ldots, \mathcal{I}_{n^0}^0\}$ *such that each* $\mathcal{I}_i^0$ *is a time interval and the following hold: (i)* $\mathcal{I}_i^0 = [\tau_i^0, \tilde{\tau}_i^0]$*, for all* $0 \le i < n^0$*; (ii) if* $n^0 < \infty$*, then either* $\mathcal{I}_{n^0}^0 = [\tau_{n^0}^0, \tilde{\tau}_{n^0}^0]$ *or* $\mathcal{I}_{n^0}^0 = [\tau_{n^0}^0, \tilde{\tau}_{n^0}^0)$*; (iii)* $\tau_i^0 \le \tilde{\tau}_i^0 = \tau_{i+1}^0$*, for all* $0 \le i < n^0$*.*

Definition 5.4.4. *Given* $\mathcal{H}^0$*, a* level 0 execution *is a collection* $\chi^0 = (\tau^0, m^0(\cdot), x^0(\cdot))$*, such that the following conditions hold:*

(i) the initial condition of the execution satisfies: $(m^0(\tau_0^0), x^0(\tau_0^0)) \in Q^{0,0}$*,*

(ii) the continuous evolution of the execution satisfies: for all $i \in \{0, \ldots, n^0\}$ *with* $\tau_i^0 < \tilde{\tau}_i^0$*, then for all* $t \in \mathcal{I}_i^0$*,* $m^0(t) \in M^0$ *is constant and* $\frac{d}{dt} x^0(t) = X^0(m^0(t))(x^0(t))$*, while for all* $t \in [\tau_i^0, \tilde{\tau}_i^0)$*,* $x^0(t) \in I^0(m^0(t))$*,*

(iii) a discrete transition of the execution satisfies: for all $i \in \{0, \dots, n^0 - 1\}$*, there exists* $s^0(i) \in \Sigma^0$ *such that* $(m^0(\tilde{\tau}_i^0), s^0(i), m^0(\tau_{i+1}^0)) =: e_i^0 \in E^0$*,* $x^0(\tilde{\tau}_i^0) \in G^0(e_i^0)$*, and* $x^0(\tau_{i+1}^0) = R^0(e_i^0)(x^0(\tilde{\tau}_i^0))$*.*

5.4.2 Higher Level Maneuver Automata

We want to extend the notions of motion primitives, maneuver automata, and executions of an MA from level 0 to higher levels by a hierarchical construction that builds level k using only information from level $k-1$. We have organized all the details of the k-th level into a tuple just like $\mathcal{H}^0$, but suitably generalized. We begin with the discrete part of $\mathcal{H}^0$ since its extension to level $k > 0$ is straightforward.

The discrete part of $\mathcal{H}^0$ is given by the tuple (M^0, Σ^0, E^0), which is interpreted as a graph with nodes M^0 (the level 0 motion primitives), labels Σ^0, and edges E^0. The extension to level $k > 0$ is analogous. The discrete part of $\mathcal{H}^k$ is a tuple (M^k, Σ^k, E^k), which is interpreted as a graph consisting of a finite set of motion primitives M^k, a set of event labels Σ^k, and a finite set of edges E^k modeling allowable successive level k motion primitives. Because level k motion primitives may be defined over more boxes than just $Y^\star$, the event labels Σ^k on transitions must be accordingly generalized.

To this end, we define the reference frame of a motion primitive. Let $m \in M^k$. We associate to m a reference frame $o_m y_1 \dots y_p$, where o_m is the base point and y_i are the Euclidean coordinate vectors in $\mathbb{R}^p$. Thus, o_m may be different for each primitive, while the coordinate axes of different frames are the same. Then the event labels $\Sigma^k = \mathbb{Z}^p \times \{-1, 0, 1\}^p$ describe the position, encoded as a box index, and direction, encoded as a face of a box, at which transitions between different motion primitives occur.

For example, consider the level 1 motion primitive in Figure 5.1 called *Two Right* with two outputs, $p = 2$. It is constructed by concatenating the level 0 motion primitive *Right* twice. Afterwards *Two Right* may, in turn, be concatenated with other level 1 motion primitives. Each possible concatenation is encoded by an edge $e = (m, s, m') \in E^1$, where $m \in M^1$ denotes *Two Right*, $m' \in M^1$ denotes a possible next level 1 motion primitive, and $s = (l, \kappa) \in \Sigma^1 \subset \mathbb{Z}^2 \times \{-1, 0, 1\}^2$ is the label. The first component l encodes the box index w.r.t. the frame for motion primitive m and the second component κ identifies the face of the box from which the transition from m to m' occurs.

Now consider the continuous time part of $\mathcal{H}^0$. It consists of a continuous state space $\mathcal{X}^0 = \mathbb{R}^n$, a set of continuous time closed-loop vector fields X^0, a set of invariants I^0 that specify the domain of each vector field, and the enabling conditions G^0 and reset maps R^0 that specify how continuous states must be reset upon reaching an enabling condition. These notions are extended to level k as follows.

The continuous state space $\mathcal{X}^0$ is reinterpreted at level k as a (discrete) state space $\mathcal{X}^k = \mathbb{Z}^p \times M^{k-1}$. As such, the notion of a continuous state translates, at level k, to pairs (l, μ), with $l \in \mathbb{Z}^p$ identifying

a *frame offset* and $\mu \in M^{k-1}$ identifying a level $k-1$ motion primitive. Clearly, such states are not continuous but instead represent level $k-1$ motion primitives attached on the discrete lattice $\mathbb{Z}^p$. Similarly, the notion at level 0 of a continuous state trajectory, which is a function of time, is replaced at level k by a discrete sequence of (l, μ) pairs.

The continuous time vector fields at level 0 are replaced by a set of discrete maps. For motion primitive $m \in M^k$, the discrete map $f_m^k : \mathcal{X}^k \times \Sigma^{k-1} \to \mathcal{X}^k$ defines the *internal transitions* on $(l, \mu) \in \mathbb{Z}^p \times M^{k-1}$ pairs that make up motion primitive m. These discrete maps must adhere to the constraints on successive level $k-1$ motion primitives as specified in $\mathcal{H}^{k-1}$ (formal details are below). The invariant $I^k(m)$ is a finite set of (l, μ) pairs that the discrete map f_m^k acts on. If a motion primitive $m \in M^k$ has an internal transition from $(l, \mu) \in I^k(m)$ to $(l', \mu') \in I^k(m)$, where μ and μ' are level $k-1$ motion primitives, then $l := o_\mu^m \in \mathbb{Z}^p$ and $l' := o_{\mu'}^m \in \mathbb{Z}^p$, denoting the frame offsets of the base points o_μ and $o_{\mu'}$ w.r.t. o_m.

The enabling conditions and reset maps take the analogous meanings as in $\mathcal{H}^0$. They define the *external transitions* from primitive $m \in M^k$ to other level k motion primitives (including m itself). For an external transition $e = (m, s, m') \in E^k$ with $s = (l_s, \kappa_s)$, we have an enabling condition $g_e^k \subset I^k(m)$ from which the external transition can occur. As mentioned before, $l_s \in \mathbb{Z}^p$ describes the box index with respect to the frame of m while $\kappa_s \in \{-1, 0, 1\}$ identifies the face through which the transition from m to m' occurs. If there is an external transition from $(l, \mu) \in g_e^k \subset I^k(m)$ to $(l', \mu') \in I^k(m')$, where μ and μ' are level $k-1$ motion primitives, then the reset map describes how these two states of the motion primitives m and m' are related.

Definition 5.4.5. *Suppose we are given $\mathcal{H}^{k-1}$, a level $k-1$ MA, where $k \geq 1$. A* level k MA *(k-MA) is a tuple $\mathcal{H}^k = (Q^k, \Sigma^k, E^k, I^k, Q^{k,0}, X^k, G^k, R^k)$, where*

State Space *$Q^k = M^k \times \mathcal{X}^k$ is the hybrid state space, where M^k is a finite index set of motion primitives at level k, and $\mathcal{X}^k = \mathbb{Z}^p \times M^{k-1}$ is the analogue of the notion of a continuous state space at level 0. The state space $\mathcal{X}^k$ consists of all (l, μ) pairs where l is the frame offset of the coordinate frame for motion primitive $\mu \in M^{k-1}$ w.r.t. the frame for some primitive $m \in M^k$.*

Labels *$\Sigma^k = \mathbb{Z}^p \times \{-1, 0, 1\}^p$ is a set of event labels. If $s = (l_s, \kappa_s) \in \Sigma^k$, then $l_s \in \mathbb{Z}^p$ is the box index w.r.t. a frame for some primitive $m \in M^k$ from which an external transition occurs, and κ_s identifies the face of the box through which the transition occurs.*

Edges *$E^k \subset M^k \times \Sigma^k \times M^k$ is a finite set of edges describing which level k motion primitives can be concatenated. Given $m \in M^k$, the set of transitions associated with edges of the form $e = (m, s, m')$ for some $m' \in M^k$ are called the* external transitions *of m.*

Invariants $I^k : M^k \to \mathcal{P}(\mathcal{X}^k)$ *assigns to each motion primitive* $m \in M^k$ *a non-empty, finite set of states* $I^k(m) \subset \mathcal{X}^k$ *on which the transition function* $f^k_m : \mathcal{X}^k \times \Sigma^{k-1} \to \mathcal{X}^k$ *(defined below) acts. The indices* $l \in \mathbb{Z}^p$ *for pairs* $(l, \mu) \in I^k(m)$ *correspond to the origin of the frame for* $\mu \in M^{k-1}$ *w.r.t. the frame for* $m \in M^k$*. The invariant is different from the* envelope *of* $m \in M^k$*, denoted* $L^k(m)$*, which is the collection of all box indices w.r.t. the frame for* m *accumulated from the envelopes of the lower level primitives that constitute* m*. We have that*

$$L^k(m) = \{l + l' \in \mathbb{Z}^p \mid (l, \mu) \in I^k(m),\ l' \in L^{k-1}(\mu)\}.$$

Initial Conditions $Q^{k,0} \subset Q^k$ *assigns a non-empty set of initial states, satisfying: if* $(m, x) \in Q^{k,0}$*, then* $x \in I^k(m)$*.*

Transition Functions $X^k : M^k \to \{f^k_m\}_{m \in M^k}$ *assigns to each* $m \in M^k$ *a transition function* $f^k_m : \mathcal{X}^k \times \Sigma^{k-1} \to \mathcal{X}^k$ *that specifies all the allowable transitions between* $(l, \mu) \in I^k(m)$ *pairs that together constitute the level* k *motion primitive. The transitions defined by* f^k_m *are called the* internal transitions *of primitive* $m \in M^k$*. We have the following requirement on* f^k_m*: if (i)* $(l, \mu) \in I^k(m)$*; and (ii)* $\sigma = (l_\sigma, \kappa_\sigma) \in \Sigma^{k-1}$*,* $l + l_\sigma + \kappa_\sigma \in L^k(m)$*, and there exists* $\tilde{\mu} \in M^{k-1}$ *with* $(\mu, \sigma, \tilde{\mu}) \in E^{k-1}$*; then (i')* $(l', \mu') = f^k_m((l, \mu), \sigma) \in I^k(m)$*; (ii')* $e' := (\mu, \sigma, \mu') \in E^{k-1}$*; and (iii')* $l = o^m_\mu$*,* $l' = o^m_{\mu'}$*, and* $l' - l = o^\mu_{\mu'}$*, where* $o^\mu_{\mu'} := O^{k-1}(e')$ *is a frame offset.*

The above requirement specifies the constraints (i'), (ii'), (iii') on the next value (l', μ') *only on the selected domain of* $\mathcal{X}^k \times \Sigma^{k-1}$ *given by (i), (ii). The selected domain refers to states* (l, μ) *in the invariant with events* σ *that can be followed by another primitive and lead to a feasible next box* $l + l_\sigma + \kappa_\sigma$ *in the same envelope (w.r.t. the frame for* m*).*

Enabling Conditions $G^k : E^k \to \{g^k_e\}_{e \in E^k}$ *assigns to each edge* $e = (m, s, m') \in E^k$ *a non-empty enabling or guard condition* $g_e \subset I^k(m)$*. If* $s = (l_s, \kappa_s) \in \Sigma^k$*, the first requirement is that* $l_s + \kappa_s \notin L^k(m)$*, identifying that this is an external transition. Then* g^k_e *consists of all those pairs* $(l, \mu) \in I^k(m)$ *for which an* external transition *to a consecutive level* k *motion primitive can occur. That is,* $(l, \mu) \in g^k_e$ *if there exists (a unique)* $\sigma = (l_\sigma, \kappa_\sigma) \in \Sigma^{k-1}$ *such that (i) there exists* $\tilde{\mu} \in M^{k-1}$ *such that* $(\mu, \sigma, \tilde{\mu}) \in E^{k-1}$*; (ii)* $l = o^m_\mu$*,* $l_s = l^m_s$ *is the index w.r.t. the frame for* m *for the box from which the external transition occurs, and* $l_\sigma = l^\mu_\sigma$ *is the index w.r.t. the frame for* μ *for the same box. Therefore* $l_s = l^m_s = o^m_\mu + l^\mu_s = l + l_\sigma$ *and* $\kappa_\sigma = \kappa_s$*.*

Reset Conditions $R^k : E^k \to \{r^k_e\}_{e \in E^k}$ *assigns to each edge* $e = (m, s, m') \in E^k$ *a reset map* $r^k_e : \mathcal{X}^k \times \Sigma^{k-1} \to \mathcal{X}^k$ *that implements the* external transitions *of* $m \in M^k$*. Suppose* $s = (l_s, \kappa_s) \in \Sigma^k$*.*

We require that there exists (a unique) frame offset $o^m_{m'} := O^k(e) \in \mathbb{Z}^p$ such that if (i) $(l,\mu) \in g^k_e$; and (ii) $\sigma = (l_\sigma, \kappa_\sigma) \in \Sigma^{k-1}$, with $l_\sigma = l^\mu_\sigma = l^m_\sigma - o^m_\mu = l_s - l$ and $\kappa_\sigma = \kappa_s$; then (i') $(l',\mu') = r^k_e((l,\mu),\sigma) \in I^k(m')$; (ii') $e' := (\mu,\sigma,\mu') \in E^{k-1}$; and (iii') $l = o^m_\mu$, $l' = o^{m'}_{\mu'}$, and $(l' + o^m_{m'}) - l = o^\mu_{\mu'}$, where $o^\mu_{\mu'} := O^{k-1}(e') \in \mathbb{Z}^p$ is a frame offset. As with the internal transitions, this specifies a constraint on the next value (l',μ') on a selected domain, which in this case corresponds to the enabling conditions.

Remark 5.4.6. *Observe that the data in H^k requires only the data of the lower level MA H^{k-1}. Additional data, namely the envelopes L^k and frame offsets O^k, are induced by H^k and H^{k-1}. The envelope for each motion primitive describes the total span of boxes on the grid $\mathbb{Z}^p$ in order to distinguish between internal and external transitions. We have the base case $L^0(m) = \{0\} \subset \mathbb{Z}^p$ for all $m \in M^0$ because all level 0 motion primitives span only over $Y^\star$. The frame offset for each edge describes the location of the frame of the second motion primitive w.r.t. the first in order to establish correctness of concatenation. For the base case, the frame offset for each edge $e = (m,s,m') \in E^0$, $s = (0,\kappa_s) \in \Sigma^0$, is $O^0(e) := o^m_{m'} = \kappa_s$ because κ_s describes the face of $Y^\star$ through which the edge occurs.*

Example 5.4.7. We illustrate how some of the motion primitives from Figure 5.1 on a two output system, $p = 2$, are encoded using our framework, see Figure 5.2. First suppose that the level 0 motion primitives are *Right* (m^0_1), *Up* (m^0_2), and *Right-Up* (m^0_3); that is, $M^0 = \{m^0_1, m^0_2, m^0_3\}$. Also associated to these motion primitives we have the level 0 events $s^0_1, s^0_2, s^0_3 \in \Sigma^0$, which refer to crossing the right face, upper face, and upper-right face on $Y^\star$, respectively.

Next we design two level 1 motion primitives called *Right&Up* (m^1_1) and *Two Right* (m^1_2); thus $M^1 = \{m^1_1, m^1_2\}$. See the top and middle of Figure 5.2, respectively. The underlying state space is $\mathcal{X}^1 = \mathbb{Z}^2 \times M^0$. The invariant of *Right&Up* is $I^1(m^1_1) = \{((0,0), m^0_3), ((1,0), m^0_2), ((0,1), m^0_1)\}$ and the invariant of *Two Right* is $I^1(m^1_2) = \{((-1,-2), m^0_1), ((0,-2), m^0_2)\}$. Considering *Two Right*, we see that the second components specify the constituent level 0 motion primitives, while first components specify the origins of the constituent level 0 motion primitives w.r.t. to the origin of *Two Right*. For example, the middle of Figure 5.2 shows that $o^{m^1_2}_{m^0_1}$ equals $(-1,-2)$ for the first *Right* and $(0,-2)$ for the second *Right*. The initial conditions may be arbitrarily assigned, for example $Q^{1,0} = \{(m^1_1, (0, m^0_3))\}$. The transition functions $f^1_{m^1_j}$ for $j = 1,2$ can be inferred by the green arrows internal between two level 0 motion primitives. For example, $f^1_{m^1_2}(((-1,-2), m^0_1), s^0_1) = ((0,-2), m^0_1)$. Moreover, each internal transition is implemented by a level 0 edge. For the previous example, we have the edge $e^0_1 := (m^0_1, s^0_1, m^0_1) \in E^0$. Notice that the frame offsets for level 0 edges are respected; for example $O^0(e^0_1) = o^{m^0_1}_{m^0_1} = (1,0)$ shown in *Two Right* means that *Right* follows *Right* one box to the right.

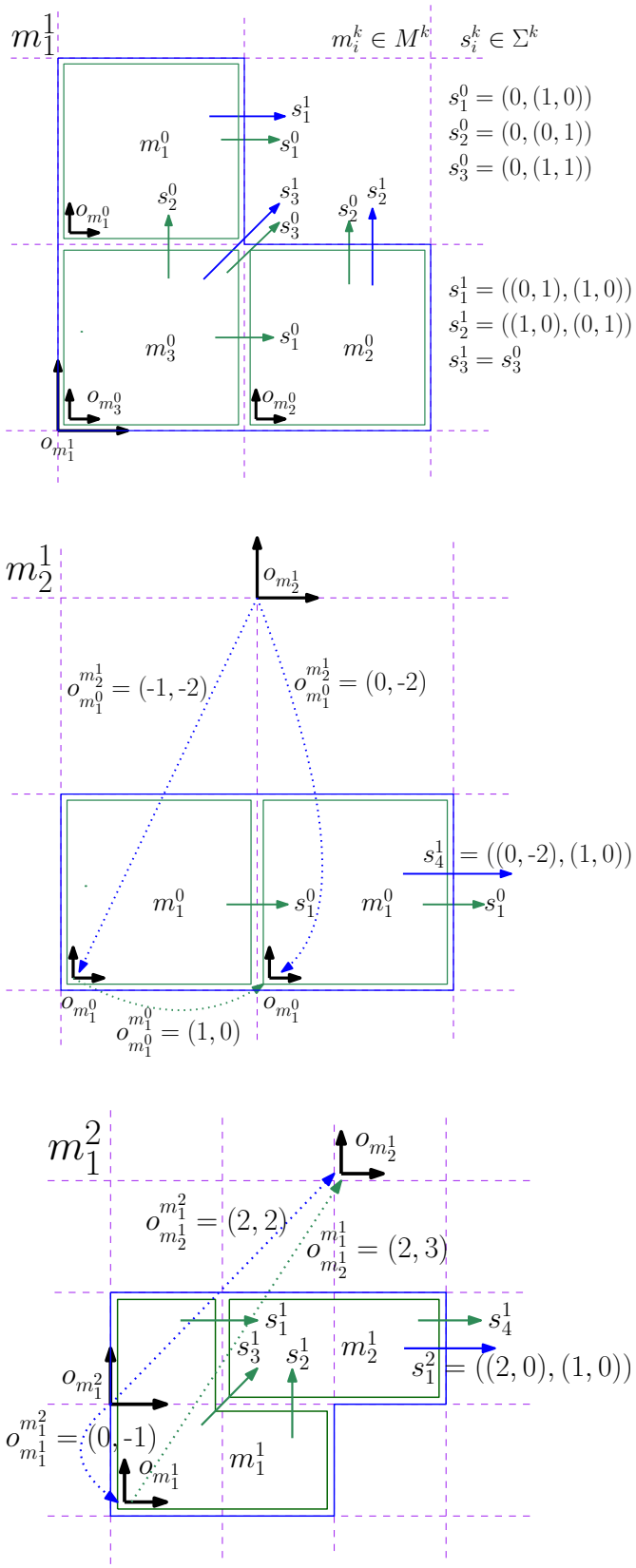

Figure 5.2: Encoding level 1 and level 2 motion primitives, inspired from the $p = 2$ output example from Figure 5.1. Refer to Example 5.4.7 for a detailed description.

The bottom of Figure 5.2 shows a level 2 motion primitive, $m_1^2 \in M^2$, which is constructed from the two level 1 motion primitives *Right&Up* and *Two Right*. In order for this construction to be consistent, care must be taken to properly define external transitions for each level 1 motion primitive. This is done by associating enabling and reset conditions to each outgoing transition, of which there are generally many possibilities. For example, we can define a level 1 edge from the upper left corner of *Right&Up* into the left side of *Two Right*. This edge is denoted $e_1^1 := (m_1^1, s_1^1, m_2^1) \in E^1$, where $s_1^1 \in \Sigma^1$ expresses that the external transition occurs from the upper left corner through the right face w.r.t. to the origin of *Right&Up*. Moreover, the enabling condition encodes all the states in the invariant of *Right&Up* for which this external transition can occur, that is, $g_{e_1^1}^1 = \{((0,1), m_1^0)\} \subset I^1(m_1^1)$. Finally, the reset condition defines how the state continues in the subsequent motion primitive *Two Right*, that is, $r_{e_1^1}^1(((0,1), m_1^0), s_1^0) = ((-1,-2), m_1^0)$. As with the internal transitions, each external transition is implemented by a level 0 edge. Referring to the edge e_1^1, the geometric layout imposes that the frame offsets between the origins of *Right&Up* and *Two Right* is $O^1(e_1^1) = o_{m_2^1}^{m_1^1} = (2,3)$. Continuing in this manner, the specific choices for the remaining level 1 edges can be inferred from $m_1^2 \in M^2$ shown at the bottom of Figure 5.2. From this point, the invariant and internal transitions at level 2 are interpreted analogously to level 1, and external transitions are defined so that potential level 3 motion primitives can be constructed. ◁

For $k \geq 1$ we define the notation $\mathcal{H}^{k-1} \preceq \mathcal{H}^k$ to mean that $\mathcal{H}^k$ is an abstraction built up from $\mathcal{H}^{k-1}$ according to Definition 5.4.5. In analogy with $\mathcal{H}^0$, we define an *execution at level* $k > 0$, denoted as $\chi^k = (\tau^k, m^k, x^k)$. There are two differences from χ^0. First, the hybrid time domain τ^k is a sequence of sets of discrete times (e.g. $\tau^k = \{\{0,1,2\}, \{3\}, \{4,5\}, \ldots\}$), where each set consists of the discrete times when internal transitions occur. Second, continuous and discrete transitions at level 0 are replaced by internal and external transitions, respectively, at level k.

Definition 5.4.8. *A* k-hybrid time domain*,* $k > 0$*, is a finite or infinite sequence* $\tau^k = \{\mathcal{I}_0^k, \ldots, \mathcal{I}_{n^k}^k\}$ *such that each* $\mathcal{I}_i^k$ *is a finite nonempty subset of discrete time points in the non-negative integers* $\{0, 1, \ldots\}$ *and the following hold: (i)* $\mathcal{I}_i^k = \{\tau_i^k, \tau_i^k + 1, \ldots, \tilde{\tau}_i^k\}$*, for all* $0 \leq i < n^k$*; (ii) if* $n^k < \infty$*, then either* $\mathcal{I}_{n^k}^k = \{\tau_{n^k}^k, \tau_{n^k}^k + 1, \ldots, \tilde{\tau}_{n^k}^k\}$ *or* $\mathcal{I}_{n^k}^k = \{\tau_{n^k}^k, \tau_{n^k}^k + 1, \ldots\}$*; (iii)* $\tau_0^k = 0$ *and* $\tau_i^k \leq \tilde{\tau}_i^k = \tau_{i+1}^k - 1$*, for all* $0 \leq i < n^k$*.*

Definition 5.4.9. *Given* $\mathcal{H}^k$ *with* $k > 0$*, a* k-execution *is a collection* $\chi^k = (\tau^k, m^k(\cdot), x^k(\cdot))$ *with* $x^k(\cdot) = (l^k(\cdot), \mu^k(\cdot))$*, such that the following hold:*

(i) the initial condition of the execution satisfies: $(m^k(\tau_0^k), x^k(\tau_0^k)) \in Q^{k,0}$*,*

(ii) *the internal evolution of the execution satisfies: for all* $i \in \{0, \dots, n^k\}$ *with* $|\mathcal{I}_i^k| > 1$*, then for all* $j \in \mathcal{I}_i^k$*,* $m^k(j) \in M^k$ *is constant and* $x^k(j) \in I^k(m^k(j))$*, while for all* $j \in \{\tau_i^k, \dots, \tilde{\tau}_i^k - 1\}$*, there exists* $\sigma^k(j) \in \Sigma^{k-1}$ *such that* $x^k(j+1) = X^k(m^k(j))(x^k(j), \sigma^k(j))$*,*

(iii) *the external evolution of the execution satisfies: for all* $i \in \{0, \dots, n^k - 1\}$*,* $x^k(\tilde{\tau}_i^k) \in I^k(m^k(\tilde{\tau}_i^k))$*, there exists* $s^k(i) \in \Sigma^k$ *such that* $(m^k(\tilde{\tau}_i^k), s^k(i), m^k(\tau_{i+1}^k)) =: e_i^k \in E^k$*, and there exists* $\sigma^k(\tilde{\tau}_i^k) \in \Sigma^{k-1}$ *such that* $x^k(\tilde{\tau}_i^k) \in G^k(e_i^k)$ *and* $x^k(\tau_{i+1}^k) = R^k(e_i^k)(x^k(\tilde{\tau}_i^k), \sigma^k(\tilde{\tau}_i^k))$*.*

5.4.3 Hierarchical Maneuver Automaton

Let $\mathcal{H} := \{\mathcal{H}^k\}_{k=0}^K$, $K \geq 0$, be a collection of k-MA such that for all $k = 1, \dots, K$, $\mathcal{H}^{k-1} \preceq \mathcal{H}^k$. We call $\mathcal{H}$ a *hierarchical maneuver automaton* (HMA). When the highest level is not clear from context, we may say $\mathcal{H}$ is a K-HMA.

For each $k = 0, \dots, K$ and $m \in M^k$, define the set of possible events as

$$\Sigma^k(m) := \{s \in \Sigma^k \mid (\exists m' \in M^k)(m, s, m') \in E^k\}. \tag{5.2}$$

Now we define the notion of runs at any level $k \geq 1$, which involves only internal transitions. Let $m \in M^k$ and consider a sequence r on $I^k(m)$, denoting its length as $|r|$. We define a *run within* m as a finite or infinite sequence $r = x^1 x^2 \cdots$ such that for all $i = 1, \dots, |r|$, $x^i = (l^i, \mu^i) \in I^k(m)$, and for all $i = 1, \dots |r| - 1$ there exists $\sigma^i \in \Sigma^{k-1}$ such that $x^{i+1} = X^k(m)(x^i, \sigma^i)$. A run within m, r, is called *maximal* if either $|r| = \infty$ or $|r| = N < \infty$ implies $\Sigma^{k-1}(\mu^N) = \emptyset$. Without loss of generality, any run can be considered the prefix of a maximal run.

Given an arbitrary HMA, it is still possible to design pathological hierarchical motion primitives that are ill suited to be used in a motion planning context. To eliminate these pathologies, we introduce the notion of a well-posed HMA. First, it consists of the seven conditions (i)-(vii) at level 0, which are recalled from Chapter 4. These serve to ensure non-blocking and uniqueness of executions. At all higher levels, two additional conditions must be enforced which are again related to non-blocking. Condition (viii) ensures that all possible events from a motion primitive are accounted for and classified either as internal or external. Condition (ix) ensures that if there is a possibility of concatenating a motion primitive to another, then all runs lead to an external transition. Overall, well-posedness enforces the property that a motion primitive either stabilizes all trajectories within its invariant or forces all trajectories to an enabling condition. Specifically, for all $k \geq 0$ and $m \in M^k$, $\Sigma^k(m) = \emptyset$ signifies remaining within m forever, while $\Sigma^k(m) \neq \emptyset$ signifies eventual exit through some level k event $s \in \Sigma^k(m)$.

Definition 5.4.10. *The HMA $\mathcal{H} := \{\mathcal{H}^k\}_{k=0}^K$ is* well-posed *if the following conditions hold:*

(i) For all $m \in M^0$, $(0,0) \notin \Sigma^0(m)$.

(ii) For all $e_1, e_2 \in E^0$ such that $e_1 = (m_1, \sigma, m_2)$ and $e_2 = (m_1, \sigma, m_3)$, $g^0_{e_1} = g^0_{e_2}$.

(iii) For all $e_1, e_2 \in E^0$ such that $e_1 = (m_1, \sigma_1, m_2)$ and $e_2 = (m_1, \sigma_2, m_3)$, if $\sigma_1 \neq \sigma_2$, then $g^0_{e_1} \cap g^0_{e_2} = \emptyset$.

(iv) For all $e_1, e_2 \in E^0$ such that $e_1 = (m_1, \sigma_1, m_2)$ and $e_2 = (m_2, \sigma_2, m_3)$, $r^0_{e_1}(g^0_{e_1}) \cap g^0_{e_2} = \emptyset$.

(v) For all $e = (m_1, \sigma, m_2) \in E^0$, $r^0_e(g^0_e) \subset I^0(m_2)$.

(vi) For all $m \in M^0$, if $\Sigma^0(m) = \emptyset$, then for all $x_0 \in I^0(m)$ and $t \geq 0$, $x(t, x_0) \in I^0(m)$ under the feedback controller $u = u_m(x)$.

(vii) For all $m \in M^0$, if $\Sigma^0(m) \neq \emptyset$, then for all $x_0 \in I^0(m)$ there exist (a unique) $\sigma \in \Sigma^0(m)$ and (a unique) $T \geq 0$ such that for all $e = (m, \sigma, m') \in E^0$ and for all $t \in [0, T]$, $x(t, x_0) \in I^0(m)$ and $x(T, x_0) \in g^0_e$ under the feedback controller $u = u_m(x)$.

(viii) For all $k = 1, \ldots, K$, $m \in M^k$, $x = (l, \mu) \in I^k(m)$, and $\sigma = (l_\sigma, \kappa) \in \Sigma^{k-1}(\mu)$, $l + l_\sigma + \kappa \in L^k(m)$ if and only if $(l + l_\sigma, \kappa) \notin \Sigma^k(m)$.

(ix) For all $k = 1, \ldots, K$ and $m \in M^k$, if $\Sigma^k(m) \neq \emptyset$, then every run within m is finite and for all $(l, \mu) \in I^k(m)$, $\Sigma^{k-1}(\mu) \neq \emptyset$.

Next, we define a *hierarchical execution* by stacking together each execution at level k. To ensure consistency across the levels, we impose that the hybrid time domains coarsen at higher levels, while both motion primitives and events that update across external transitions at any level correspond to the internal motion primitives and events at the level above. Figure 5.3 shows an example using the motion primitives detailed in Figure 5.2.

Definition 5.4.11. *Let $\mathcal{H} := \{\mathcal{H}^k\}_{k=0}^K$, for $K \geq 0$, be a well-posed HMA. A* hierarchical-execution *of $\mathcal{H}$ is a collection of k-executions $\chi := \{\chi^k\}_{k=0}^K$, such that for all $k = 1, \ldots, K$:*

(i) $\bigcup_{i=0}^{n^k} \mathcal{I}^k_i = \{0, \ldots, n^{k-1}\}$;

(ii) for all $j \in \{0, \ldots, n^{k-1}\}$ and $j' \in \mathcal{I}^{k-1}_j$, $\mu^k(j) = m^{k-1}(j')$;

(iii) for all $j \in \{0, \ldots, n^{k-1} - 1\}$, $\sigma^k(j) = s^{k-1}(j)$.

Given a hierarchical execution, we can record the sequence of offsets corresponding to the level 0 events. This generates the *behavior* of the hierarchical execution, analogously to how an output trajectory

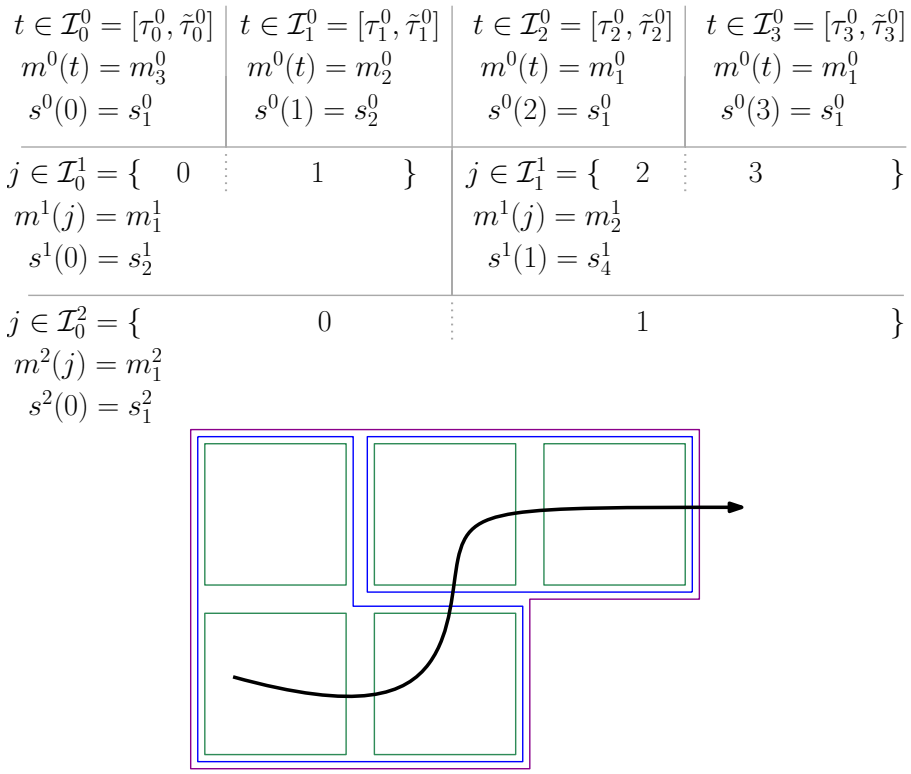

Figure 5.3: The details of a particular hierarchical execution using the motion primitives from Figure 5.2.

$y(\cdot, x_0)$ generates a behavior. Referring to Figure 5.3, the corresponding behavior is $(1,0)(0,1)(1,0)(1,0) \in \mathcal{B}$. The language of an HMA consists of all the possible behaviors that can be generated in this way.

Definition 5.4.12. *Let $\mathcal{H} := \{\mathcal{H}^k\}_{k=0}^K$, for $K \geq 0$, be a well-posed HMA. If χ is a hierarchical-execution of $\mathcal{H}$, the* behavior *of χ is defined as the sequence of offsets obtained from the level 0 events in χ^0; that is, the behavior is $\kappa^0\kappa^1\cdots$, where $s^0(i) = (0, \kappa^i) \in \Sigma^0$ for $i \in \{0, \ldots, n^0 - 1\}$. The* language *of $\mathcal{H}$, denoted by $\mathcal{L}(\mathcal{H})$, is the set of all behaviors induced by the hierarchical executions of $\mathcal{H}$. If $0 \leq k' \leq K$, then for convenience we write $\mathcal{L}(\mathcal{H}^{k'}) := \mathcal{L}(\{\mathcal{H}^k\}_{k=0}^{k'})$ to denote the language up to and including level k'.*

Similar to hierarchical executions, we define a set of hierarchically compatible states, $\mathcal{Q}(\mathcal{H}) \subset \prod_{k=0}^K Q^k$. Informally, hierarchically compatible states stack the individual hybrid states at each level while maintaining consistency between the motion primitives implemented at each level and recorded at the level below. Writing $q := (q^0, q^1, \ldots, q^K) \in \prod_{k=0}^K Q^k$, with $q^k = (m^k, x^k)$ for $k = 0, \ldots, K$ and $x^k = (l^k, \mu^k)$ for $k = 1, \ldots, K$, we have $q \in \mathcal{Q}(\mathcal{H})$ if for all $k = 1, \ldots, K$, $x^k \in I^k(m^k)$ and $m^{k-1} = \mu^k$. Let $\mathcal{Q}_0(\mathcal{H}) \subset \mathcal{Q}(\mathcal{H})$ be the corresponding initial states; that is, $q \in \mathcal{Q}_0(\mathcal{H})$ if $q \in \mathcal{Q}(\mathcal{H})$ and for all $k = 0, \ldots, K$, $q^i \in Q^{k,0}$.

We also state the following result, which says that the language at each level is contained in the language of any level below. This implies that behaviors become more restricted as one adds more levels to a HMA.

Lemma 5.4.13. *Let $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$ be a HMA that is well-posed. Then $\mathcal{L}(\mathcal{H}^{k''}) \subset \mathcal{L}(\mathcal{H}^{k'})$ for all $0 \leq k' < k'' \leq K$.*

Proof. Let $0 \leq k' < k'' \leq K$ and define the HMAs $\mathcal{H}' = \{\mathcal{H}^k\}_{k=0}^{k'}$, $\mathcal{H}'' = \{\mathcal{H}^k\}_{k=0}^{k''}$. Take any behavior in $b'' \in \mathcal{L}(\mathcal{H}'')$; by definition, this means that there is at least one hierarchical execution $\chi'' = \{\chi^k\}_{k=0}^{k''}$ that generates it (by no means unique). For the level 0 execution χ^0, observe that well-posedness (iii) (disjoint guards) implies that for all $i = 0, \ldots, n^0 - 1$, the events $s^0(i) = (0, \kappa^i) \in \Sigma^0$ across the discrete transitions are uniquely determined. Thus $b'' = \kappa^0\kappa^1 \cdots$. Ignoring the levels $k > k'$ from χ'', it is easy to see that the collection $\chi' = \{\chi^k\}_{k=0}^{k'}$ is also a hierarchical execution. By definition, χ' generates a behavior $b' \in \mathcal{L}(\mathcal{H}')$. Since χ' and χ'' share the same χ^0, we must have that $b'' = b' \in \mathcal{L}(\mathcal{H}')$. □

5.5 Main Results

In this section we show that if one has a HMA that satisfies certain conditions, then we can construct initial conditions and a hierarchical feedback controller that solve Problem 5.3.1. We begin with the notion of completeness, in which the highest level K consists of only one motion primitive, no external transitions, and has all states as valid initial conditions.

Definition 5.5.1. *Let $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$ be an HMA with $K \geq 1$. We say that $\mathcal{H}$ is* complete *if $M^K = \{m^K\}$, $E^K = \emptyset$, and $Q^{K,0} = \{(m, x) \in Q^K \mid x \in I^K(m)\}$.*

Completeness establishes a hierarchical maneuver automaton $\mathcal{H}$ whose top level $\mathcal{H}^K$ acts as a *control policy* dictating the assignment of motion primitives one level below, with those motion primitives dictating the assignment of motion primitives below that, and so on. From this principle we will derive the required set of initial conditions $\mathcal{X}_0 \subset \mathbb{R}^n$ and feedback control strategy $u(x)$. One of our main tasks is to relate the trajectories $x(\cdot, x_0)$ of (5.1) for $x_0 \in \mathcal{X}_0$ to hierarchical executions χ of the given HMA $\mathcal{H}$.

Suppose $\mathcal{H}$ is complete and well-posed. First, the reference frame of the top level motion primitive m^K is chosen to coincide with the reference frame in which the feasible boxes $L_f \subset \mathbb{Z}^p$ and goal boxes $L_g \subset L_f$ are defined. Then each hierarchically compatible state $q \in \mathcal{Q}(\mathcal{H})$ can be matched to a unique continuous state $x \in \mathbb{R}^n$ as follows. The continuous level 0 state $x^0 \in I^0(m^0)$ projected to the output space lies in $Y^\star$. This box is located at the discrete location $l^1 = o_{m^0}^{m^1} \in \mathbb{Z}^p$ with respect to the frame of

the motion primitive m^1. Continuing in this way, since $l^k = o^{m^k}_{m^{k-1}}$ for $k = 1, \dots, K$, we have that the sum of the reference frame shifts $\ell := \sum_{k=1}^{K} l^k \in \mathbb{Z}^p$ represents the location of $Y^\star$ in the reference frame of o_{m^K}. As such, we have the output space relationship

$$h(x) = h(x^0) + d \circ \ell, \tag{5.3}$$

where d is the grid vector of the lengths of $Y^\star$ and $\circ$ is the component-wise product. Lifting this relationship to the state space, the proposed set of initial conditions for the main problem is

$$\mathcal{X}_0 = \left\{ x^0 + h_o^{-1}\left(d \circ \ell\right) \mid q \in \mathcal{Q}_0(\mathcal{H}) \right\} \subset \mathbb{R}^n. \tag{5.4}$$

Next, the associated feedback control $u(x) = u(x, q)$ requires to record the current hierarchically consistent state $q \in \mathcal{Q}(\mathcal{H})$. To improve clarity and minimize duplicated information, it suffices to record only the internal states $(x^1, \dots, x^K)$. If the initial state $x_0 \in \mathcal{X}_0$ corresponds to $q_0 \in \mathcal{Q}_0(\mathcal{H})$ in the sense of (5.3), we initialize these internal states as $(x_0^1, \dots, x_0^K)$. The controller $u(x)$ consists of updating the internal states based on measured events and then applying the feedback controller for the current level 0 motion primitive. Algorithm 1 provides the update scheme, which is called at each time instant with the arguments

$$u(x - h_o^{-1}\left(d \circ \ell\right), x^1, \dots, x^K). \tag{5.5}$$

As will be shown, this algorithm is well-defined in that it provides an unambiguous update scheme (Lemma 5.5.4) and that the relationship (5.3) is always maintained (Lemma 5.5.6 (ii)).

We now elaborate on Algorithm 1. The input arguments are the level 0 continuous state $x^0 \in \mathcal{X}^0$ and the internal states $x^k = (l^k, \mu^k) \in \mathcal{X}^k$ from levels 1 to K. Lines 2 and 3 record the current motion primitives $m^{k-1} = \mu^k \in M^{k-1}$ for $k = 1, \dots, K$ for later use (on line 8). On line 4, it is checked whether the continuous state x^0 lies on a guard set $g^0_{e^0}$ for some edge $e^0 = (\mu^1, s^0, \cdot) \in E^0$. If it does not, none of the internal states need to be updated and the control value $u_{\mu^1}(x^0)$ is returned along with the internal states. If it does, then line 5 proceeds by determining all the levels at which the level 0 event s^0 is interpreted as an event $\sigma^k \in \Sigma^{k-1}$ from level 0 up to some level $K' \leq K$. Internal states above level K' are unchanged. Line 6 applies an internal transition at level K' to determine the next level K' internal state (notice that if $K' = K$, then $\mu^{K+1} := m^K$). Then lines 7 to 9 successively apply the reset maps down the hierarchy to update the internal states at each level below. Lines 10 and 11 reset the continuous state at level 0, which is then used to calculate the control value. The control value and updated internal states are returned.

Algorithm 1 Hierarchical Feedback Controller

function $u(x^0, (l^1, \mu^1), \ldots, (l^K, \mu^K))$ ▷ Input internal states in $\mathcal{X}^k$ for $k = 0, \ldots, K$
  for $k = 1, \ldots, K$ **do**
    $\mu^k_{\text{old}} \leftarrow \mu^k$ ▷ Save current motion primitives $\mu^k \in M^{k-1}$
  if GuardSet(x^0, μ^1) **then** ▷ Determine if $x^0 \in g^0_{e^0}$ for some $e^0 = (\mu^1, \cdot, \cdot) \in E^0$
    $\{\sigma^1, \ldots, \sigma^{K'}\} \leftarrow$ GetEvents$(x^0, (l^1, \mu^1), \ldots, (l^K, \mu^K))$ ▷ Extract events $\sigma^k \in \Sigma^{k-1}$ up to some $K' \leq K$
    $(l^{K'}, \mu^{K'}) \leftarrow f^{K'}_{\mu^{K'+1}}((l^{K'}, \mu^{K'}), \sigma^{K'})$ ▷ Internal transition at level K'
    for $k = K' - 1, \ldots, 1$ **do**
      $e^k \leftarrow (\mu^{k+1}_{\text{old}}, \sigma^{k+1}, \mu^{k+1})$ ▷ Form the edge $e^k \in E^k$
      $(l^k, \mu^k) \leftarrow r^k_{e^k}((l^k, \mu^k), \sigma^k)$ ▷ External transition at level k
    $e^0 \leftarrow (\mu^1_{\text{old}}, \sigma^1, \mu^1)$ ▷ Form the edge $e^0 \in E^0$
    $x^0 \leftarrow r^0_{e^0}(x^0)$ ▷ Reset map at level 0
  return $u_{\mu^1}(x^0), (l^1, \mu^1), \ldots, (l^K, \mu^K)$ ▷ Return control value $u_{\mu^1}(x^0)$ and updated continuous states

Our main result gives three conditions to solve Problem 5.3.1. The avoid property is given by condition (i) and is formulated in terms of the top level envelope, which describes all the possible visited boxes in the frame of m^K. By applying the recursive definition of the envelopes, we have that the top level envelope is

$$L^K(m^K) = \{\ell \mid q \in \mathcal{Q}(\mathcal{H})\} \subset \mathbb{Z}^p. \tag{5.6}$$

The reach property is given by condition (ii) and is formulated in terms of the set of box indices in the frame of m^K corresponding to hierarchical states in which the motion primitive at each level stabilizes trajectories,

$$\mathcal{G} = \left\{\ell \mid q \in \mathcal{Q}(\mathcal{H}),\ \ \Sigma^{k-1}(\mu^k) = \emptyset,\ k = 1, \ldots, K\right\}. \tag{5.7}$$

The behavioral property is given by condition (iii) and is stated in terms of the language of the HMA.

Theorem 5.5.2. *Consider the system* (5.1) *with a gridded output space in terms of grid length vector* $d \in \mathbb{R}^p$. *We are given a feasible and goal set of boxes,* $L_g \subset L_f \subset \mathbb{Z}^p$ *and a behavior constraint* $\widehat{\mathcal{B}} \subset \mathcal{B}$. *Consider a HMA* $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$ *which is well-posed and complete. Suppose that*

(i) $L^K(m^K) \subset L_f$;

(ii) $\mathcal{G} \subset L_g$, *and for all* $k = 1, \ldots, K$ *and* $m \in M^k$ *such that* $\Sigma^k(m) = \emptyset$, *every maximal run within* m *is finite;*

(iii) $\mathcal{L}(\mathcal{H}) \subset \widehat{\mathcal{B}}$.

Then the set of initial conditions $\mathcal{X}_0$ (5.4) *and the feedback control law* $u(x)$ *given by* (5.5) *and Algorithm 1 solves Problem 5.3.1.*

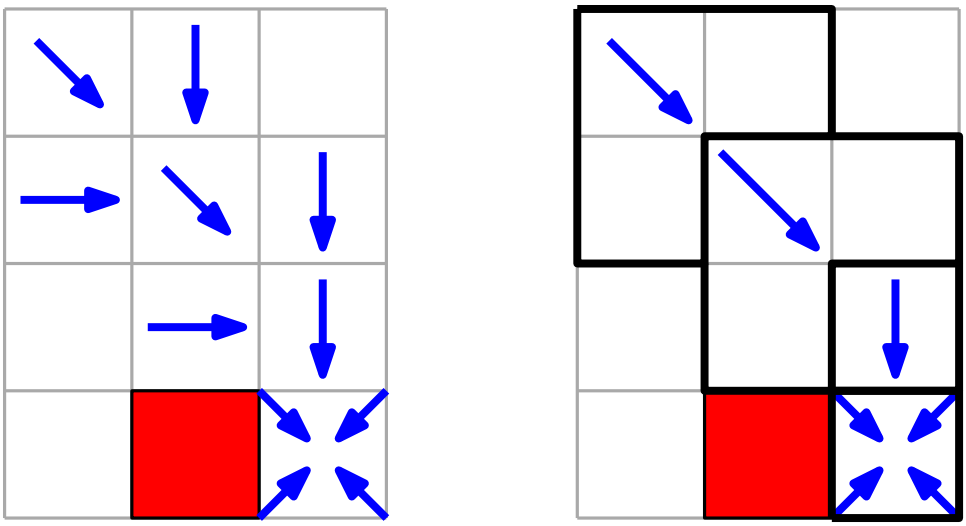

Figure 5.4: This figure compares a level 1 control policy generated using level 0 atomic motion primitives (left) and a level 2 control policy generated using level 1 motion primitives (right). In both cases, the policy reaches the goal box at the lower right while avoiding the obstacle (red), although the level 2 policy is simpler to encode. As the number of outputs grows (for example, as more vehicles are added), the effect of simplification becomes more dramatic.

Remark 5.5.3. *In practice, the control policy yielding a complete $\mathcal{H}$ is obtained by running a planning algorithm at level $K-1$ using only the discrete part $(M^{K-1}, \Sigma^{K-1}, E^{K-1})$ and knowledge of the feasible and goal boxes L_f and L_g. Notice that the control policy is fully specified by the invariant $I^K(m^K)$ and transition function $X^K(m^K)$. Also, if the lower levels are designed such that $\mathcal{L}(\mathcal{H}^{K-1}) \subset \widehat{\mathcal{B}}$, then Lemma 5.4.13 implies that any assignment of level $K-1$ motion primitives preserves language satisfaction, $\mathcal{L}(\mathcal{H}^K) \subset \widehat{\mathcal{B}}$. In this case, the reach-avoid aspect is decoupled from the behavior specification, which makes the planning step more computationally tractable. Since the discrete part $(M^{K-1}, \Sigma^{K-1}, E^{K-1})$ is essentially just a graph with labeled transitions, one may adapt algorithms used in Chapter 4 for planning at level $K-1$ or other graph search methods. The main advantage of planning at this higher level is that the desired behavior will be automatically enforced, and ideally there are significantly less transitions as compared to planning at level 0. Figure 5.4 illustrates a comparison between a control policy at $K=1$ and $K=2$.*

In the remainder of this section, we prove the main result. To this end, several lemmas are stated, which are mainly generalizations of results presented in Section 4.5. Before proceeding, we introduce a few more definitions and notation.

Consider a level 0 MA $\mathcal{H}^0$. The *execution time* of a 0-execution χ^0 is defined as $\mathscr{T}(\chi^0) := \sum_{i=0}^{n^0}(\tilde{\tau}_i^0 - \tau_i^0) = \lim_{i \to n^0} \tilde{\tau}_i^0$. A 0-execution is called *finite* if τ^0 is a finite sequence ending with a compact time interval. A 0-execution is called *infinite* if either τ^0 is an infinite sequence or if $\mathscr{T}(\chi^0) = \infty$. Finally,

a 0-execution is called *Zeno* if it is infinite but $\mathscr{T}(\chi^0) < \infty$. Since Zeno executions imply that motion primitives change infinitely many times in a finite time interval, one should ensure that they do not occur.

For an HMA $\mathcal{H}$, a hierarchical execution χ is called *maximal* if χ^0 is infinite and for all $k = 1, \dots, K$, $n^k < \infty$ implies that $\Sigma^k(m^k(\tau^k_{n^k})) = \emptyset$ and the run within $m^k(\tau^k_{n^k}) \in M^k$, $r^k = x^k(\tau^k_{n^k}) \cdots x^k(\tilde{\tau}^k_{n^k})$, is maximal. Let χ be a hierarchical execution. We may equivalently write it as a collection of hierarchically compatible states $q(t) = (q^0(t), \dots, q^K(t)) \in \mathcal{Q}(\mathcal{H})$ over the level 0 time domain τ^0, that is $t \in \tau^0$ (which will be an abbreviation for $t \in \mathcal{I}^0_j$ for some $j = 0, \dots, n^0$). For each level $k = 1, \dots, K$, $j = 0, \dots, n^0$, and $t \in \mathcal{I}^0_j$, the discrete states in $q^k(t) = (m^k(t), (l^k(t), \mu^k(t))$ are constant. For each $t \in \tau^0$, let $\ell(t) := \sum_{k=1}^K l^k(t)$ denote the current box index in the reference frame of o_{m^K}. For each level $k = 0, \dots, K-1$, the level k external events $s^k(\cdot) \in \Sigma^k$, are interpreted as occurring only at the endpoint times $\tilde{\tau}^0_j$ for some $j \in \{0, \dots, n^0 - 1\}$ depending on when they occur at their coarser discrete time $\tilde{\tau}^k_{j'}$ for some $j' \in \{0, \dots, n^k - 1\}$. To ease notation, we do not explicitly relate the intervals j at level 0 and j' and level k. By definition of a hierarchical execution, the $s^k(\cdot)$ are equal to the level $k+1$ internal events $\sigma^{k+1}(\cdot)$. When $\mathcal{H}$ is well-posed, Algorithm 1 provides a unique and maximal execution to each initial hierarchical state. The following lemma gives the construction.

Lemma 5.5.4. *Consider a HMA $\mathcal{H}$ which is well-posed and complete. Then for all $q_0 \in \mathcal{Q}_0(\mathcal{H})$ there exists a unique maximal hierarchical execution with initial condition q_0 and initial time $\tau^0_0 = 0$.*

Proof. Let $q_0 \in \mathcal{Q}_0(\mathcal{H})$ and write $q_0 = (q^0_0, \dots, q^K_0) \in \prod_{k=0}^K Q^{k,0}$, $q^k_0 = (m^k_0, x^k_0)$ for $k = 0, \dots, K$, and $x^k_0 = (l^k_0, \mu^k_0)$ for $k = 1, \dots, K$. To construct a hierarchical execution $\chi = \{\chi^k\}_{k=0}^K$, let the level 0 initial time to be $\tau^0_0 = 0$. By definition, $\tau^k_0 = 0$ for all $k = 1, \dots, K$. Thus for all $k = 0, \dots, K$, $m^k(0) = m^k_0$ and $x^k(0) = x^k_0$. So far we have that for all $k = 0, \dots, K$, $\tilde{\tau}^k_0 = 0$, $\tau^k = \{\mathcal{I}^k_0\}$, and $\mathcal{I}^k_0 = \{\tau^k_0\}$. With χ initialized, we construct the remainder of χ inductively. This is done by extending the solution x^0 over the current interval $\mathcal{I}^0_{n^0}$ and initializing it in the next one $\mathcal{I}^0_{n^0+1}$, updating the discrete evolutions at higher levels $k > 0$ as necessary; Algorithm 1 gives the update logic. We have established the base case, with $n^k = 0$ for all $k = 0, \dots, K$.

Suppose that χ has been defined over the hybrid time domains $\tau^k = \{\mathcal{I}^k_0, \dots, \mathcal{I}^k_{n^k}\}$ for $k = 0, \dots, K$, with $\tau^k_{n^k} = \tilde{\tau}^k_{n^k}$. Write $j = n^0$. We extend χ by one step to $j+1$.

If $\Sigma^0(m^0(\tau^0_j)) = \emptyset$, we extend $\tilde{\tau}^0_j$ to ∞ using well-posedness (vi) (invariance); the solution x^0 remains in the current motion primitive for all future time and the induction terminates with $j = n^0$.

Otherwise if $\Sigma^0(m^0(\tau^0_j)) \neq \emptyset$, well-posedness (vii) allows us to extend x^0 to a new $\tilde{\tau}^0_j$. That is, there exist unique $s^0 \in \Sigma^0(m^0(\tau^0_j))$ and $T \geq 0$ such that for all $t \in [\tau^0_j, \tau^0_j + T]$, $m^0(t) = m^0(\tau^0_j)$, and

$x^0(t) \in I^0(m^0(t))$. Also, for each $e^0 = (m^0(\tau_j^0), s^0, \tilde{m}^0) \in E^0$, there exists a guard set g_{e^0} such that $x^0(\tau_j^0 + T) \in g_{e^0}$. By well-posedness (ii), for all such $\tilde{m}^0$, the guard set is the same, while (iii) establishes the uniqueness of s^0. Overall, the interval $\mathcal{I}_j^0$ is extended to $[\tau_j^0, \tilde{\tau}_j^0]$ with $\tilde{\tau}_j^0 = \tau_j^0 + T$ and we initialize the next interval as $\mathcal{I}_{j+1}^0 = \{\tilde{\tau}_j^0\}$, and record $s^0(\tilde{\tau}_j^0) = s^0 \in \Sigma^0$. Also since $s^0 = (0, \kappa)$, write $\kappa(\tilde{\tau}_j^0) = \kappa$.

Before we can determine the new specific $\tilde{m}^0$, we must go up the hierarchy at the transition time $t' := \tilde{\tau}_j^0$ to see whether the level 0 event $s^0(t')$ can be interpreted as an external event at level k, $0 < k < K$, within the frame $o_{m^k(t')}$. Let K' be the smallest level at which there is no corresponding event $s^{K'}(t') \in \Sigma^{K'}$ that is registered at t'; by the existence of $s^0(t')$ and completeness, we have that $K' \in \{1, \ldots, K\}$.

This procedure is well defined in that if there is such an s^k at level k, then it is uniquely determined. We proceed inductively. Using the hierarchical relationship, write the internal level 1 event as $\sigma^1 = (l_{\sigma^1}, \kappa_{\sigma^1}) = (0, \kappa) = s^0$. By definition of the enabling conditions, we require the level 1 external event to be $s^1 = (l^1(t') + l_{\sigma^1}, \kappa_{\sigma^1})$. If $K = 1$ or $s^1 \notin \Sigma^1(m^1(t'))$, then we set $K' = 1$ and terminate the induction. Otherwise we have established the base case. Assuming we have the event $s^k \in \Sigma^k$ for $1 \leq k < K$ with $s^k = (l^k(t') + l_{\sigma^k}, \kappa_{\sigma^k})$ and $\sigma^k = (l_{\sigma^k}, \kappa_{\sigma^k}) \in \Sigma^{k-1}$, we attempt to construct $s^{k+1} \in \Sigma^{k+1}$. Using the hierarchical relationship, write the level $k+1$ internal event as $\sigma^{k+1} = (l_{\sigma^{k+1}}, \kappa_{\sigma^{k+1}}) = s^k$. By definition of the enabling conditions at level $k+1$, we require the level $k+1$ external event to be $s^{k+1} = (l^{k+1}(t') + l_{\sigma^{k+1}}, \kappa_{\sigma^{k+1}})$. If $K = k+1$ or $s^{k+1} \notin \Sigma^{k+1}(m^{k+1}(t'))$, then we set $K' = k+1$ and terminate the induction. Otherwise the induction continues. In summary, we see that at each level $0 < k < K'$, the external event $s^k \in \Sigma^k(m^k(t'))$ is uniquely determined. We record $\sigma^k(t') = \sigma^k$ for $k = 1, \ldots, K'$ and $s^k(t') = s^k$ for $k = 1, \ldots, K' - 1$.

Now that K' is determined, we must go down the hierarchy to determine the new states at $t'' := \tau_{j+1}^0$. At levels $K' < k \leq K$, there is no internal event $\sigma^k(t')$ and therefore no possibility for internal transitions. At level $k = K'$, since we have an only an internal event $\sigma^k(t')$, we employ an internal transition and n^k remains unchanged. At levels $0 \leq k < K'$, since we have an external event $s^k(t')$ and its corresponding internal event $\sigma^k(t')$, we employ an external transition, adding a new interval to τ^k. We have already introduced $\mathcal{I}_{j+1}^0$ to τ^0, so for $k = 1, \ldots, K' - 1$, we begin the next interval in τ^k as $\mathcal{I}_{n^k+1}^k = \{\tilde{\tau}_{n^k}^k + 1\}$.

Starting with $k = K' \geq 1$, we apply the transition function to get that $m^k(t'') = m^k(t')$ and $x^k(t'') = X^k(m^k(t'))(x^k(t'), \sigma^k(t'))$. We check the two properties (i) and (ii) of X^k to be able to conclude (i'), (ii'), and (iii'). First, since $x^k(t') \in I^k(m^k(t'))$ due to the previous step of the induction, (i) holds. Second, we need to show that $\sigma^k(t') \in \Sigma^{k-1}(\mu^k(t'))$ and $l^k(t') + l_{\sigma^k} + \kappa_{\sigma^k} \in L^k(m^k(t'))$. By definition of a hierarchical execution we have $s^{k-1}(t') = \sigma^k(t')$ and $\mu^k(t') = m^{k-1}(t')$, and by construction we have $s^{k-1}(t') \in \Sigma^{k-1}(m^{k-1}(t'))$. Thus we have $\sigma^k(t') \in \Sigma^{k-1}(\mu^k(t'))$. Next, since by construction

$(l^k(t') + l_{\sigma^k}, \kappa_{\sigma^k}) \notin \Sigma^k(m^k(t'))$ (for otherwise s^k would have been defined), we can apply well-posedness (viii) to get that $l^k(t') + l_{\sigma^k} + \kappa_{\sigma^k} \in L^k(m^k(t'))$. Thus we can conclude that (i') $x^k(t'') \in I^k(m^k(t''))$, (ii') $e^{k-1} := (\mu^k(t'), \sigma^k(t'), \mu^k(t'')) \in E^{k-1}$, and (iii') $l^k(t'') - l^k(t') = o^{\mu^k(t')}_{\mu^k(t'')}$.

Next, we apply the reset functions from $k = K' - 1$ down to $k = 0$, using the edge

$$e^k = (m^k(t'), s^k(t'), m^k(t'')) = (\mu^{k+1}(t'), \sigma^{k+1}(t'), \mu^{k+1}(t'')) \in E^k$$

defined the iteration above, where we have used the hierarchical relationship. The base case edge $e^{K'-1}$ was established above when $k = K'$. These edges determine the next motion primitives $m^k(t'')$ for $k = 0, \ldots, K' - 1$. Applying the reset function for each $k = K' - 1, \ldots, 1$, we get $x^k(t'') = R^k(e^k)(x^k(t'), \sigma^k(t'))$. Similarly, we check (i) and (ii) of R^k to conclude (i'), (ii'), and (iii'). By the same reasoning as for the internal transition at level K', $x^k(t') \in I^k(m^k(t'))$ and $\sigma^k(t') \in \Sigma^{k-1}(\mu^k(t'))$. On the other hand, since now by construction $(l^k(t') + l_{\sigma^k}, \kappa_{\sigma^k}) \in \Sigma^k(m^k(t'))$, we can use well-posedness (viii) to get that $l^k(t') + l_{\sigma^k} + \kappa_{\sigma^k} \notin L^k(m^k(t'))$. So $x^k(t') \in G^k(e^k)$ and we conclude that (i') $x^k(t'') \in I^k(m^k(t''))$, (ii') $e^{k-1} := (\mu^k(t'), \sigma^k(t'), \mu^k(t'')) \in E^{k-1}$, and (iii') $l^k(t'') + o^{m^k(t')}_{m^k(t'')} - l^k(t') = o^{\mu^k(t')}_{\mu^k(t'')}$.

Finally, when $k = 0$, we apply the reset $R^0(e^0)$ to get $x^0(t'') = R^0(e^0)(x^0(t'))$ and $m^0(t'') = \mu^1(t'')$ using the hierarchical relationship. By well-posedness (v), we have $x^0(t'') \in I^0(m^0(t''))$. We have completed the induction step from j to $j + 1$.

The inductive procedure above constructed a unique execution χ. It remains to show that it is maximal. If the induction above terminated, then $n^0 < \infty$ with $\mathcal{I}^0_{n^0} = [\tau^0_{n^0}, \infty)$. Thus $\mathscr{T}(\chi^0) = \infty$ and so χ^0 is infinite. If the induction does not terminate, then $n^0 = \infty$ and χ^0 is also infinite.

Next, we identify the smallest level $\tilde{K} \in \{1, \ldots, K\}$ such that $n^k < \infty$. If the induction above terminated, then $\tilde{K} = 1$, otherwise at most $\tilde{K} = K$, since $n^K = 0$ by completeness. By definition of a hierarchical execution, $n^K \leq \cdots \leq n^0$, and so we have $n^k < \infty$ for all $k = \tilde{K}, \ldots, K$. When $k = 1, \ldots, \tilde{K} - 1$, there is nothing to check for the maximality of χ. For $k \in \{\tilde{K}, \ldots, K\}$, we must show that $\Sigma^k(m^k(\tau^k_{n^k})) = \emptyset$ and the run $r^k := x^k(\tau^k_{n^k}) \cdots x^k(\tilde{\tau}^k_{n^k})$ within $m^k(\tau^k_{n^k})$ is maximal, that is, either $\tilde{\tau}^k_{n^k} = \infty$ or $\Sigma^{k-1}(\mu^k(\tilde{\tau}^k_{n^k})) = \emptyset$. We proceed by induction. For the base case $k = \tilde{K}$, by the hierarchical relationship we have $\tilde{\tau}^k_{n^k} = n^{k-1} = \infty$, so the run r^k is infinite, and therefore maximal. The contrapositive of well-posedness (ix) says that: for all $k = 1, \ldots, K$ and $m \in M^k$, if there exists a run within m that is infinite or there exists $x = (l, \mu) \in I^k(m)$ with $\Sigma^{k-1}(\mu) = \emptyset$, then $\Sigma^k(m) = \emptyset$. Since r^k is infinite, we must have that $\Sigma^k(m^k(\tau^k_{n^k})) = \emptyset$. Now assume $\Sigma^k(m^k(\tau^k_{n^k})) = \emptyset$ and r^k is maximal for $\tilde{K} \leq k < K$ and show $\Sigma^{k+1}(m^{k+1}(\tau^{k+1}_{n^{k+1}})) = \emptyset$ and r^{k+1} is maximal. By the hierarchical relationship, $\tilde{\tau}^{k+1}_{n^{k+1}} = n^k < \infty$, and so r^{k+1} is finite. By the hierarchical relationship again, $\mu^{k+1}(n^k) = m^k(\tau^k_{n^k})$

and thus by assumption, $\Sigma^k(\mu^{k+1}(n^k)) = \emptyset$. Thus r^{k+1} is maximal. Using the contrapositive of well-posedness (ix) again, we have that $\Sigma^{k+1}(m^{k+1}(n^k)) = \emptyset$, as desired. □

During the duration of a fixed level 0 motion primitive, if the physical continuous states of (5.1) and level 0 MA continuous states initially satisfy the output space relationship (5.3), they remain shifted by the same amount. Its proof follows easily from the definition of a 0-execution and Assumption 4.3.2.

Lemma 5.5.5. *Consider any level 0 execution $\chi^0 = (\tau^0, m^0(\cdot), x^0(\cdot))$ with $\tau^0 = \{\mathcal{I}_0^0\}$, and let $y \in \mathbb{R}^p$. If $x_0 = x^0(\tau_0^0) + h_o^{-1}(y)$, then the trajectory $x(\cdot, x_0)$ of (5.1) with the feedback control $u(x) = u_{m^0(\tau_0^0)}(x - h_o^{-1}(y))$ satisfies*

$$x(t, x_0) = x^0(t) + h_o^{-1}(y),\ \forall t \in \mathcal{I}_0^0.$$

Proof. Since by definition of a 0-execution $m^0(t)$ is constant for all $t \in I_0^0$, write $m = m^0(\tau_0^0)$. By definition of χ^0, $\frac{d}{dt}(x^0(t)) = f(x^0(t), u_m(x^0(t)))$ for all $t \in \mathcal{I}_0^0$. Using this relationship for $t \in \mathcal{I}_0^0$, Assumption 4.3.2, and the fact that y is constant gives

$$\begin{aligned}
\frac{d}{dt}x(t, x_0) &= \frac{d}{dt}(x^0(t) + h_o^{-1}(y)) = \frac{d}{dt}(x^0(t)) \\
&= f(x^0(t), u_m(x^0(t))) \\
&= f(x^0(t) + h_o^{-1}(y), u_m(x^0(t))), \\
&= f(x(t, x_0), u_m((x^0(t) + h_o^{-1}(y)) - h_o^{-1}(y))) \\
&= f(x(t, x_0), u_m(x(t, x_0) - h_o^{-1}(y))).
\end{aligned}$$

This shows that $x(t, x_0) = x^0(t) + h_o^{-1}(y)$ is a solution to (5.1) with the feedback control $u(x) = u_m(x - h_o^{-1}(y))$ over $\mathcal{I}_0^0$. Since $f(\cdot, u_m(\cdot))$ is globally Lipschitz, this must be the unique solution. □

We can apply Lemma 5.5.5 repeatedly to show that (5.3) is preserved throughout a hierarchical execution χ, see (5.9) below. To prove this, we first show (5.8), which says that the box indices differ by an amount $\kappa(\cdot) \in \{-1, 0, 1\}^p$ across the level 0 discrete transitions. Once we have these results, the proof of the main result follows easily.

Lemma 5.5.6. *Consider a HMA $\mathcal{H}$ which is well-posed and complete. Let χ be a hierarchical execution.*

(i) Writing the level 0 events as $s^0(\cdot) = (0, \kappa(\cdot))$, we have

$$\kappa(\tilde{\tau}_j^0) + \ell(\tilde{\tau}_j^0) = \ell(\tau_{j+1}^0),\ j = 0, \ldots, n^0 - 1. \tag{5.8}$$

(ii) If $x_0 = x^0(\tau_0^0) + h_o^{-1}\left(d \circ \ell(\tau_0^0)\right)$*, then the trajectory* $x(\cdot, x_0)$ *of* (5.1) *with the feedback control* (5.5) *satisfies: for all* $j \in \{0, \dots, n^0\}$

$$x(t, x_0) = x^0(t) + h_o^{-1}\left(d \circ \ell(t)\right), \quad t \in \mathcal{I}_j^0. \tag{5.9}$$

Proof. Let χ be a hierarchical execution, and we use the usual notation for all its components.

For (i), we show that (5.8) holds. Let $j = 0, \dots, n^0 - 1$, $t' := \tilde{\tau}_j^0$, and $t'' := \tau_{j+1}^0$. As in the proof of Lemma 5.5.4, let K' be the smallest level at which there is no corresponding event $s^{K'}(t') \in \Sigma^{K'}$ that is registered at t'. We have the following transition relationships

$$\begin{aligned}
l^k(t') + o^{\mu^k(t')}_{\mu^k(t'')} &= l^k(t'') + o^{m^k(t')}_{m^k(t'')}, && 1 \le k < K', \\
l^k(t') + o^{\mu^k(t')}_{\mu^k(t'')} &= l^k(t''), && k = K', \\
l^k(t') &= l^k(t''), && K' < k \le K.
\end{aligned}$$

That is, at levels $k < K'$ we apply the resets R^k; at level $k = K'$ we apply only the transition function X^k; and at levels $k > K'$ there is no update. This is the same logic in Algorithm 1. Observe that by definition of the transformations and hierarchical execution, $o^{\mu^1(t')}_{\mu^1(t'')} = \kappa(t')$ for all $j = 0, \dots, n^0 - 1$. Further, we have for all $k = 1, \dots, K-1$ that $o^{m^k(t')}_{m^k(t'')} = o^{\mu^{k+1}(t')}_{\mu^{k+1}(t'')}$. Using these two these relationships, the definition of $\ell(\cdot)$, and summing all the transition relationships from $k = 1, \dots, K$, we have that

$$\ell(t') + \kappa(t') + \sum_{k=2}^{K'} o^{\mu^k(t')}_{\mu^k(t'')} = \ell(t'') + \sum_{k=1}^{K'-1} o^{\mu^{k+1}(t')}_{\mu^{k+1}(t'')}.$$

Observing the cancellations among the transformations then gives (5.8).

For (ii), we must show that for all $j \in \{0, \dots, n^0\}$ that (5.9) holds. We establish this by induction. For $j = 0$, (5.9) follows from Lemma 5.5.5 with the execution χ^0 over just $\mathcal{I}_0^0$ and $y = d \circ \ell(\tau_0^0)$.

Next assume that (5.9) is true for $0 \le j < n^0$ and show that it is true for $j+1$. Using (5.9) for j at $t = \tilde{\tau}_j^0$ gives

$$\begin{aligned}
x(\tilde{\tau}_j^0, x_0) &= x^0(\tilde{\tau}_j^0) + h_o^{-1}\left(d \circ \ell(\tilde{\tau}_j^0)\right) \\
&= x^0(\tau_{j+1}^0) + h_o^{-1}\left(d \circ \kappa(\tilde{\tau}_j^0)\right) + h_o^{-1}\left(d \circ \ell(\tilde{\tau}_j^0)\right) \\
&= x^0(\tau_{j+1}^0) + h_o^{-1}\left(d \circ \ell(\tau_{j+1}^0)\right) \\
&= x(\tau_{j+1}^0, x_0),
\end{aligned}$$

where the second line follows from the definition of the reset at level 0, the third line follows from applying the component-wise multiplication with d and insertion map h_o^{-1} to (5.8), and the last line follows from the first line since $\tilde{\tau}_j^0 = \tau_{j+1}^0$. Applying Lemma 5.5.5 with the execution χ^0 over just $\mathcal{I}_{j+1}^0$ and $y = d \circ \ell(\tau_{j+1}^0)$, we have that (5.9) is true for $j+1$, as desired. □

Proof of Theorem 5.5.2. Let $x_0 \in \mathcal{X}_0$. We must show that the output trajectory $y(\cdot, x_0)$ under the feedback $u(x)$ given in Algorithm 1 satisfies the (i) avoid, (ii) reach, and (ii) behavior specifications. The three assumed properties (i), (ii), (iii) address each of these specifications respectively.

Since $x_0 \in \mathcal{X}_0$, let $q_0 \in \mathcal{Q}_0(\mathcal{H})$ be any associated HMA initial state such that $x_0 = x_0^0 + h_o^{-1}(d \circ \ell_0)$. By Lemma 5.5.4, there exists a unique maximal execution χ with initial condition q_0 and initial time $\tau_0^0 = 0$. We write the corresponding hierarchical states as $q(t)$ for $t \in \tau^0$.

First we show that for all $k = 0, \ldots, K$, $n^k < \infty$. By completeness, $n^K = 0 < \infty$ since we can never have an external transition at level K. This establishes the base case. Next we proceed by induction. Supposing that $n^k < \infty$ for $1 \leq k \leq K$, we show that $n^{k-1} < \infty$. Consider the sequence $r^k = x^k(\tau_{n^k}^k) \cdots x^k(\tilde{\tau}_{n^k}^k)$. By definition of χ^k, it is a run within $m^k(\tau_{n^k}^k) \in M^k$. Since $n^k < \infty$, the maximality of χ implies that $\Sigma^k(m^k(\tau_{n^k}^k)) = \emptyset$ and r^k is maximal. Then $\Sigma^k(m^k(\tau_{n^k}^k)) = \emptyset$ and assumption (ii) imply that r^k is finite. Finally the definition of a hierarchical execution implies that $\tilde{\tau}_{n^k}^k = n^{k-1} < \infty$, as desired.

Also observe that χ^0 is non-Zeno. Since the maximality of χ implies that χ^0 is infinite and since $n^0 < \infty$, we must have that $\mathscr{T}(\chi^0) = \infty$.

Now we apply Lemma 5.5.6 (ii) to χ and x_0. Projecting (5.9) to the output space, we obtain that for all $j \in \{0, \ldots, n^0\}$ and $t \in \mathcal{I}_j^0$, $y(t, x_0) \in Y_{\ell(t)}$.

We show the avoid specification is satisfied. For all $t \in \tau^0$, since $q(t) \in \mathcal{Q}(\mathcal{H})$, we have $\ell(t) \in L^K(m^K) \subset L_f$. Since $\mathscr{T}(\chi^0) = \infty$, $y(t, x_0) \in \bigcup_{j=0}^{n^0} Y_{\ell(t)} \subset \bigcup_{l \in L_f} Y_l$ for all $t \in [0, \infty)$.

Next we show the reach specification is satisfied. Since for all $k = 0, \ldots, K$, $n^k < \infty$, the maximality of χ implies that $\Sigma^k(m^k(\tau_{n^k}^k)) = \emptyset$. Thus we have that $\ell(\tau_{n^k}^k) \in \mathcal{G} \subset L_g$. Since $n^0 < \infty$, the maximality of χ^0 implies that $\mathcal{I}_{n^0}^0 = [\tau_{n^0}^0, \infty)$. Hence $y(t, x_0) \in Y_{\ell(t)} \subset \bigcup_{l \in L_g} Y_l$ for $t \in [\tau_{n^0}^0, \infty)$.

Finally, we show the behavior specification is satisfied. Recalling that the discrete transitions of level 0 correspond to some event $(0, \kappa) \in \Sigma^0$ (except $(0, 0)$, which is ruled out by well-posedness (i)), (5.8) shows that the current box is incremented by an element in $\{-1, 0, 1\}^p \setminus \{0\}$. Since for all $j \in \{0, \ldots, n^0\}$ and $t \in \mathcal{I}_j^0$, $y(t, x_0) \in Y_{\ell(t)}$, we have that the behavior induced by $y(t, x_0)$ is $\kappa(\tilde{\tau}_0^0)\kappa(\tilde{\tau}_1^0) \cdots \in \mathcal{L}(\mathcal{H}) \subset \widehat{\mathcal{B}}$. □

Remark 5.5.7. *The reach condition (ii) in Theorem 5.5.2 is conservative in order to eliminate any possibility of Zeno behavior. When dealing with more complex task specifications such as linear temporal*

logic (LTL), the reach condition would need to be relaxed. Following the idea that many interesting tasks can be decomposed into a sequence of reach-avoid problems [116], we present a method of addressing these problems in Section 5.8.4. In this case, Zeno behavior may result if the motion primitives are not well designed. Providing general but easily checkable conditions at level 0 and higher levels to ensure non-Zenoness remains an open problem. Fortunately, the lemmas provided in this section hold true independently of any reach condition, and so only a modification to the relatively short proof of Theorem 5.5.2 would be needed.

5.6 Composing Hierarchical Maneuver Automata

In practice, real systems can be very complex and the design of motion primitives can be a challenging task. To aid in this design process, system structure can often be exploited so that simpler motion primitives can be designed on smaller subsystems and then automatically combined to yield motion primitives for the entire system.

In this section, we present two methods of composing HMA, which are based on the cartesian product and union of sets, respectively. The first is *parallel composition*, which is a procedure for obtaining more complex motion primitives by stacking independent subsystems with existing motion primitives. The second is *union*, which aggregates different sets of motion primitives for the same underlying system.

It turns out that beyond level 0, there is generally no unique way to define these composition procedures. As such, we only formulate the essential guidelines that any concrete implementation ought to follow. Moreover, these composition procedures are also straightforward to implement in our applications. Formal details can be investigated in future work.

5.6.1 Parallel Composition

Given two independent 0-MA's $\mathcal{H}_1^0$ and $\mathcal{H}_2^0$, we defined the parallel composition $\mathcal{H}^0 = \mathcal{H}_1^0 \parallel \mathcal{H}_2^0$ in Chapter 4. As it was the case that the parallel composition of 0-MAs preserved well-posedness, the main requirement here is again to ensure that well-posedness is preserved.

Let $\mathcal{H}_i^0$ for $i = 1, 2$ be two 0-MAs, each with p_i outputs. In our previous work, the parallel composition $\mathcal{H}^0 = \mathcal{H}_1^0 \parallel \mathcal{H}_2^0$ was a 0-MA with $p = p_1 + p_2$ outputs and $M^0 = M_1^0 \times M_2^0$. Central to the parallel composition was a set of augmented edges $\overline{E}_i^0 = E_i^0 \cup F_i^0$ for $i = 1, 2$, where F_i^0 was disjoint from E_i^0 and included feasible edges involving the only possible internal event, $(0, 0) \in \Sigma_i^0$. These extra edges F_i^0 are necessary to correctly describe the asynchronicity between the two 0-MAs when they are composed. These notions are now extended to the parallel composition of HMAs.

Definition 5.6.1. *Consider two well-posed K-HMAs $\mathcal{H}_i = \{\mathcal{H}_i^k\}_{k=0}^K$, $i = 1, 2$, each with p_i outputs. A* parallel composition of HMAs, *denoted $\mathcal{H} = \mathcal{H}_1 \parallel \mathcal{H}_2$, is a K-HMA $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$ with $p = p_1 + p_2$ outputs, where*

- *for all $k = 0, \ldots, K$, $\mathcal{H}^k = \mathcal{H}_1^k \parallel \mathcal{H}_2^k$ is a k-MA with $M^k = M_1^k \times M_2^k$,*
- *$\mathcal{H}$ is well-posed.*

The extension to the parallel composition of a finite number of HMAs, $\mathcal{H}_j$, $j = 1, \ldots, J$, follows similarly and is denoted $\|_{j=1}^J \mathcal{H}_j$. At higher levels $k > 0$, one needs to define a set of augmented edges $\overline{E}_i^k$ for $i = 1, 2$, see Remark 5.6.2 below.

Remark 5.6.2. *The main difficulty in defining a generic parallel composition of HMAs lies in the fact that there can be multiple feasible choices for the reset conditions of the composed motion primitives. For example, suppose we have two level 1 motion primitives $m_1 \in M_1^1$ and $m_2 \in M_2^1$, where there exists an edge $e_1 = (m_1, s_1, m_1') \in E_1^1$ but $\Sigma_2^1(m_2) = \emptyset$. Then for the composed motion primitive $m = (m_1, m_2) \in M^1$, well-posedness (viii) ultimately requires us to define at least one edge of the form $e = (m, s, m') \in E^1$. The composed edge $e \in E^1$ will consist of $e_1 \in E_1^1$ and some suitably defined augmented edge $e_2 = (m_2, s_2, m_2') \in \overline{E}_2^1$. Similarly, the composed reset condition $R^1(e)$ will consist of $R_1^1(e_1)$ and some suitable reset condition for the augmented edge e_2, which cannot invoke R_2^1 since $\Sigma_2^1(m_2) = \emptyset$. When $m_2 = m_2'$, we may invoke the internal transitions defined in $X_2^k(m_2)$. It also may be feasible and desirable to define a reset condition for the case $m_2 \neq m_2'$ if the envelope of the m_2 is contained in m_2', although there may be multiple feasible choices if $|I_2^1(m_2')| > 1$.*

5.6.2 Union

Suppose that we have two 0-MAs $\mathcal{H}_1^0$ and $\mathcal{H}_2^0$, both over the same grid and system with p outputs. Without loss of generality, the motion primitive sets M_1^0 and M_2^0 can be reindexed so that they are disjoint. Informally, the union $\mathcal{H}^0 = \mathcal{H}_1^0 \cup \mathcal{H}_2^0$ pools together the two sets of motion primitives and forms additional feasible edges between the two sets of motion primitives. These notions are now extended to HMAs.

Definition 5.6.3. *Consider two well-posed K-HMAs $\mathcal{H}_i = \{\mathcal{H}_i^k\}_{k=0}^K$, $i = 1, 2$, each with p outputs. A* union of HMAs, *denoted $\mathcal{H} = \mathcal{H}_1 \cup \mathcal{H}_2$, is a K-HMA $\mathcal{H} = \{\mathcal{H}^k\}_{k=0}^K$ with p outputs, where*

- *for all $k = 0, \ldots, K$, $\mathcal{H}^k = \mathcal{H}_1^k \cup \mathcal{H}_2^k$ is a k-MA with $M^k = M_1^k \cup M_2^k$,*
- *$\mathcal{H}$ is well-posed.*

Remark 5.6.4. *If no additional edges between M_1^k and M_2^k are defined, their union is rather trivial but not very useful. The main purpose of union is to essentially automate the construction of edges between individually designed sets of motion primitives. As with the parallel composition, for levels $k > 0$, there may be multiple feasible ways to form a new reset condition $R^k(e)$ for an additional edge $e = (m_1, s, m_2) \in E^k$, where $m_i \in M_i^k$, $i = 1, 2$, and $s \in \Sigma_1^k(m_1)$. The k-MA requires a concrete choice to be made for each edge in order to have a well-defined implementation.*

5.7 A Library of Hierarchical Motion Primitives

In this section we present the design of a 1-HMA. It builds on the level 0 motion primitives for a system composed of double integrators that we presented in Chapter 4, which we proved were also well-posed. Our presentation of the 1-HMA is still quite abstract, so that it can be used as a building block for more complex maneuvers. It is instrumental towards addressing a multi-agent formation control problem, and it also serves as a model for other motion primitives that we use when morphing between different formations. This design is applied to quadrocopters in Section 5.8, showing that our method scales up well relative to the number of vehicles.

5.7.1 Base Level 0 Motion Primitives

Suppose that the system (5.1) has $p \geq 1$ outputs, with each modelled as a double integrator, so that the number of states is $n = 2p$. Expressing all the position states followed by the velocity states, we have $\dot{x}_i = x_{i+p}$ and $\dot{x}_{i+p} = u_i$ for $i = 1, \ldots, p$.

Consider first $p = 1$ for a double integrator system, where the box $Y^\star$ is just a segment of fixed length. Borrowing our previous design from Section 4.7.2 we have a single output level 0 MA, denoted as $\mathcal{H}^0_{\mathscr{H}\mathscr{F}\mathscr{B}}$. It consists of the three motion primitives $M^0_{\mathscr{H}\mathscr{F}\mathscr{B}} = \{\mathscr{H}, \mathscr{F}, \mathscr{B}\}$, which are *Hold*, *Forward*, and *Backward*, respectively. *Hold* stabilizes all output trajectories over a box, *Forward* causes them to move forward to the next box, and *Backward* causes them to move backward one box. The top of Figure 5.5 summarizes the discrete part of the design. While the continuous part is not salient here, we recall that the feedback control is parameterized by the segment length of $Y^\star$ and a desired maximum acceleration u^*. In contrast to Section 4.7.2, here we have also allowed the (feasible) edges between *Forward* and *Backward*.

Now consider a collection of $p \geq 1$ double integrator systems. Performing a p-fold parallel composition, we represent the overall system capabilities with the p output 0-MA $\mathcal{H}^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^p} := \|_{j=1}^p \mathcal{H}^0_{\mathscr{H}\mathscr{F}\mathscr{B}}$. The motion primitives are all the combinations $M^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^p} = (M^0_{\mathscr{H}\mathscr{F}\mathscr{B}})^p$, while the edges $E^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^p}$

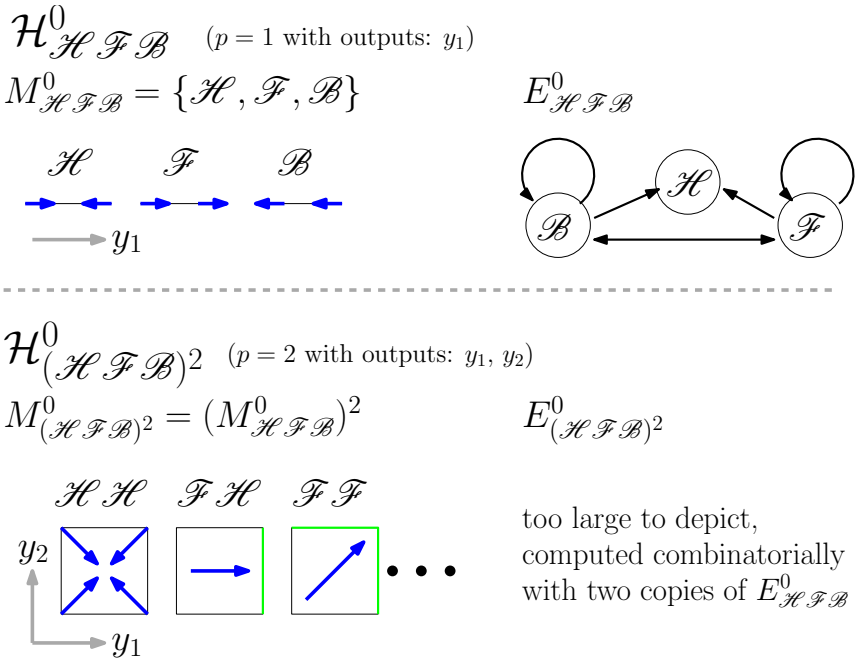

Figure 5.5: This figure illustrates level 0 motion primitives for a system of p double integrators. The top row shows the atomic motion primitives $M^0_{\mathcal{H}\mathcal{F}\mathcal{B}}$ and their edges $E^0_{\mathcal{H}\mathcal{F}\mathcal{B}}$ for one double integrator ($p = 1$), which are *Hold* ($\mathcal{H}$), *Forward* ($\mathcal{F}$), and *Backward* ($\mathcal{B}$). The bottom row shows that two copies of the $p = 1$ design may be composed to yield motion primitives for $p = 2$. This procedure extends easily to an arbitrary number of p outputs.

correspond to feasible combinations of edges in $E^0_{\mathcal{H}\mathcal{F}\mathcal{B}}$. For example, when there are two outputs, $p = 2$, *Right* corresponds to *Forward* in the first output and *Hold* in the second output. Notice that these edges are non-trivial in order to maintain correctness at the continuous level; for $p = 2$, the edge $((\mathcal{F}, \mathcal{F}), ((0, 0), (1, 0)), (\mathcal{F}, \mathcal{H})) \notin E^0_{(\mathcal{H}\mathcal{F}\mathcal{B})^2}$ because the second component cannot switch motion primitives without crossing to the next box, $(\mathcal{F}, (0, 0), \mathcal{H}) \notin E^0_{\mathcal{H}\mathcal{F}\mathcal{B}}$, for otherwise safety cannot be guaranteed. The bottom of Figure 5.5 shows the composed design for $p = 2$.

5.7.2 Formation Constrained Motion Primitives

We begin by expressing a behavioral constraint that captures the notion of multiple agents maintaining a certain relative spacing as they move over the gridded output space. As an example, consider two agents moving in the same physical direction, each modeled as a double integrator with a scalar output. The behavioral constraint to maintain a formation of two agents, each with one output, is that if one agent incurs an increment in its output value due to the application of a motion primitive, then the other agent experiences the same increment in the successive motion primitive.

Now we define the general p output formation behavioral constraint, $\mathcal{B}^\star \subset \mathcal{B}$. Consider a behavior

$y_b = \kappa^1\kappa^2\cdots \in \mathcal{B}$. Then $y_b \in \mathcal{B}^\star$ if for all $i = 1, 2, \cdots$, and for all $j, j' \in \{1, \ldots, p\}$,

$$\left| \sum_{l=1}^{i} \left(\kappa_j^l - \kappa_{j'}^l\right) \right| \leq 1. \tag{5.10}$$

This constraint ensures that over a run and every pair of outputs, the accumulation of offsets differs by at most one box index.

Using the 0-MA $\mathcal{H}^0_{(\mathscr{H}\,\mathscr{F}\,\mathscr{B})^p}$ given in the previous section for a fixed $p \geq 1$, we design an abstraction $\mathcal{H}^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star} \succeq \mathcal{H}^0_{(\mathscr{H}\,\mathscr{F}\,\mathscr{B})^p}$ such that $\mathcal{L}(\mathcal{H}^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}) \subset \mathcal{B}^\star$. It is easy to see that the language of the 0-MA is not contained in the desired behavior, $\mathcal{L}(\mathcal{H}^0_{(\mathscr{H}\,\mathscr{F}\,\mathscr{B})^p}) \not\subset \mathcal{B}^\star$, except when $p = 1$. For example, when $p = 2$, the assignment of motion primitives $(\mathscr{F}, \mathscr{H})(\mathscr{F}, \mathscr{H})$ produces the behavior $(1, 0)(1, 0) \notin \mathcal{B}^\star$. Our approach consists of coupling the allowed choices of motion primitives in each output. As such, we design three level 1 motion primitives, which we call *Formation Hold* ($\mathscr{H}^\star$), *Formation Forward* ($\mathscr{F}^\star$), and *Formation Backward* ($\mathscr{B}^\star$). *Formation Hold* causes no change in all outputs, *Formation Forward* causes all outputs to move forward exactly one box, while *Formation Backward* causes all outputs to move backward one box.

We now describe the implementation details when there are two outputs, $p = 2$; see Figure 5.6. First, the state space is $Q^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star} = M^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star} \times \mathcal{X}^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}$, where $M^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star} = \{\mathscr{B}^\star, \mathscr{H}^\star, \mathscr{F}^\star\}$ and $\mathcal{X}^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star} = \mathbb{Z}^2 \times M^0_{(\mathscr{H}\,\mathscr{F}\,\mathscr{B})^2}$. We denote the origin of each motion primitive as $o_{\mathscr{H}^\star}, o_{\mathscr{F}^\star}, o_{\mathscr{B}^\star} = 0 \in \mathbb{Z}^2$ and define for convenience the three starting states $x_{\mathscr{H}^\star} = (0, (\mathscr{H}, \mathscr{H}))$, $x_{\mathscr{F}^\star} = (0, (\mathscr{F}, \mathscr{F}))$, and $x_{\mathscr{B}^\star} = (0, (\mathscr{B}, \mathscr{B}))$ in $\mathcal{X}^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}$, which consist respectively of *Hold*, *Forward*, and *Backward* in both output components at the grid origin.

For $\mathscr{H}^\star$, the invariant is the singleton $I^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}(\mathscr{H}^\star) = \{x_{\mathscr{H}^\star}\}$ and $(\mathscr{H}^\star, x_{\mathscr{H}^\star}) \in Q^{1,0}_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}$ is an initial state. Since $\Sigma^0_{(\mathscr{H}\,\mathscr{F}\,\mathscr{B})^2}((\mathscr{H}, \mathscr{H})) = \emptyset$, there are no level 1 edges associated with $\mathscr{H}^\star$, and no need to specify specific values for the transition function $X^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}(\mathscr{H}^\star)$.

For $\mathscr{F}^\star$, the invariant $I^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}(\mathscr{F}^\star)$ is $\{x_{\mathscr{F}^\star}, ((1, 0), (\mathscr{H}, \mathscr{F})), ((0, 1), (\mathscr{F}, \mathscr{H}))\}$ and $(\mathscr{F}^\star, x_{\mathscr{F}^\star}) \in Q^{1,0}_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}$ is an initial state. Identifying $\mathscr{H} = 0$, $\mathscr{F} = 1$, and $\mathscr{B} = -1$, the transition function is defined formulaically as

$$X^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}(\mathscr{F}^\star)((l, \mu), \sigma) = (l + \kappa, \mu - \kappa) \tag{5.11}$$

for all $(l, \mu) \in I^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}(\mathscr{F}^\star)$ and $\sigma = (0, \kappa) \in \Sigma^0_{(\mathscr{H}\,\mathscr{F}\,\mathscr{B})^2}(\mu)$. An external transition occurs from a state $x = (l, \mu) \in I^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}(\mathscr{F}^\star)$ when $\sigma = (0, \kappa)$ is such that $X^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}(\mathscr{F}^\star)((l, \mu), \sigma) = ((1, 1), (0, 0))$. The associated edge $e = (\mathscr{F}^\star, s, m') \in E^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}$ has the corresponding level 1 event $s = (l, \kappa)$ and may allow concatenation to any level 1 primitive $m' \in M^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star}$, with enabling condition $g^1_e = \{x\}$, reset

condition $r_e^1(x, \sigma) = x_{m'}$ (a starting state), and transformation $O^1(e) = (1, 1)$. A similar construction applies to $\mathscr{B}^\star$.

The generalization to p outputs is straightforward. Note that although the size of the invariants of $\mathscr{F}^\star$ and $\mathscr{B}^\star$ and the number of edges from them increases exponentially with p, $M^1_{\mathscr{H}^\star \mathscr{F}^\star \mathscr{B}^\star}$ always consists of just three formation motion primitives. Moreover, a practical implementation does not require to explicitly write all the states of the invariants and edges, since the formula (5.11) can be generalized.

Remark 5.7.1. *By construction provided above, it is easy to see that $\mathcal{H}^1_{\mathscr{H}^\star \mathscr{F}^\star \mathscr{B}^\star}$ is in fact a 1-MA that abstracts $\mathcal{H}^0_{(\mathscr{H} \mathscr{F} \mathscr{B})^p}$ and that it satisfies conditions (viii) and (ix) of well-posedness. Intuitively, the behavior requirement $\mathcal{L}(\mathcal{H}^1_{\mathscr{H}^\star \mathscr{F}^\star \mathscr{B}^\star}) \subset \mathcal{B}^\star$ is also satisfied. A formal verification is facilitated by the observation that all reset conditions enter an initial state in $Q^{1,0}_{\mathscr{H}^\star \mathscr{F}^\star \mathscr{B}^\star}$ with all pairwise offsets equal to zero in* (5.10).

Remark 5.7.2. *Based on the discrete nature of the design, there is no immediate reason to expect that the associated output trajectories would exhibit a high degree of coordination at the continuous level. For example, if the outputs represent two vehicles moving in the same physical direction, assigning $\mathscr{F}^\star$ would command both vehicles to move forward exactly one box. If we continue to assign $\mathscr{F}^\star$, the vehicles would continue move exactly forward one box with each assignment of $\mathscr{F}^\star$, but not necessarily make their instantaneous transitions into the next box at the same time instants. Interestingly, it was found that the supplied design* does *lead to a coordination at the continuous level as well; this phenomenon is studied in Chapter 6.*

5.8 Quadrocopter Applications

In this section we illustrate how our hierarchical motion planning framework can be applied to a collection of quadrocopters. We consider two different centralized objectives, formation flight and formation morphing, and formulate them as behavior-constrained reach-avoid problems. We apply the hierarchical motion primitive designs from Section 5.7 and show how standard planning algorithms can be used to efficiently generate control policies at the highest level. The results are then experimentally demonstrated in Section 5.9. We begin with showing how our designed motion primitives from the previous section apply to quadrocopters over a gridded output space.

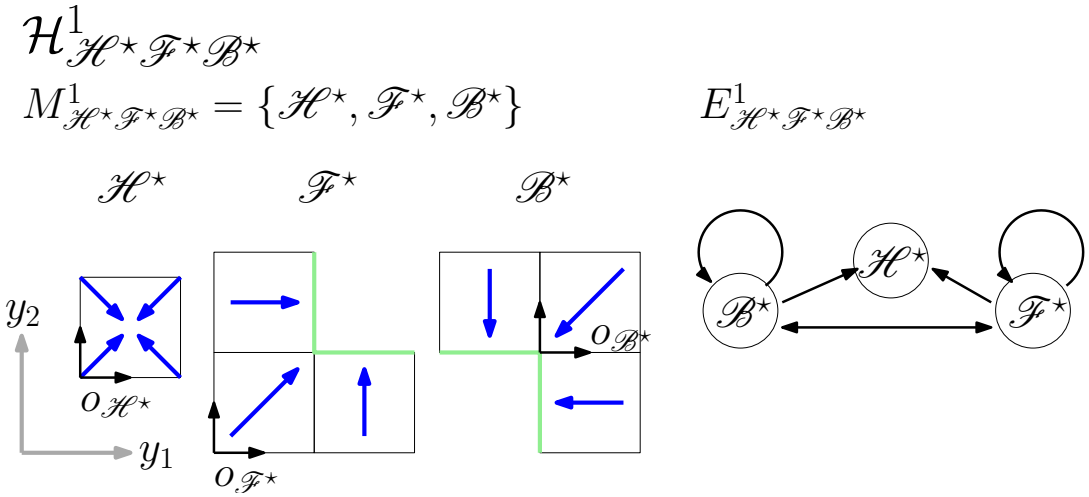

Figure 5.6: This figure illustrates a level 1 MA design $\mathcal{H}^1_{\mathcal{H}^\star\mathcal{F}^\star\mathcal{B}^\star}$ to achieve the formation behavior constraint. There are three level 1 motion primitives $M^1_{\mathcal{H}^\star\mathcal{F}^\star\mathcal{B}^\star}$, which are *Formation Hold* ($\mathcal{H}^\star$), *Formation Forward* ($\mathcal{F}^\star$), and *Formation Backward* ($\mathcal{B}^\star$). The implementation of these motion primitives uses the level 0 MA shown in Figure 5.5. Although this figure depicts the implementation for $p = 2$, this design is easily scalable to p outputs. The edges $E^1_{\mathcal{H}^\star\mathcal{F}^\star\mathcal{B}^\star}$ are independent of p.

5.8.1 Modeling

As in Section 4.8, we can apply the atomic motion primitives for the (x_w, y_w, z_w) directions on all the quadrocopters. Suppose that we have a collection of N quadrocopters. As also described previously, the output space grid is induced by gridding the physical 3D space. In what follows, for simplicity we ignore the z_w direction so that $p = 2N$. The extension to 3D is straightforward.

5.8.2 Policy Generation for Formation Flight

For the formation flight objective, a collection of quadrocopters must reach a goal location in a cluttered but known environment while maintaining a fixed formation. We encode the formation by a collection of a relative offsets, which describe the desired box separation in each physical direction from a reference vehicle to the i-th vehicle. Choosing the first vehicle as the reference, let $F^i = (F^i_1, F^i_2) \in \mathbb{Z}^2$ for $i = 1, \ldots, N$ denote the relative offsets. Referring to the first step shown in Figure 5.8, we have the reference vehicle has $F^1 = 0$ and the other vehicle has $F^2 = (0, 2)$.

Now we specify the objective as a behavior-constrained reach-avoid problem. Let $L_{f,w} \subset \mathbb{Z}^2$ be a finite collection of feasible boxes in the world frame and $L_{g,w} = \{l_g\} \in L_{f,w}$ be the goal box for the first vehicle. Then the feasible boxes $L_f \subset \mathbb{Z}^p$ correspond to joint boxes where each vehicle is in $L_{f,w}$ and each pair of vehicles has at least a one-box separation in some physical direction for safety. The goal box $L_g \subset L_f$ corresponds to the joint box where the first vehicle is in $L_{g,w}$ and the other vehicles are

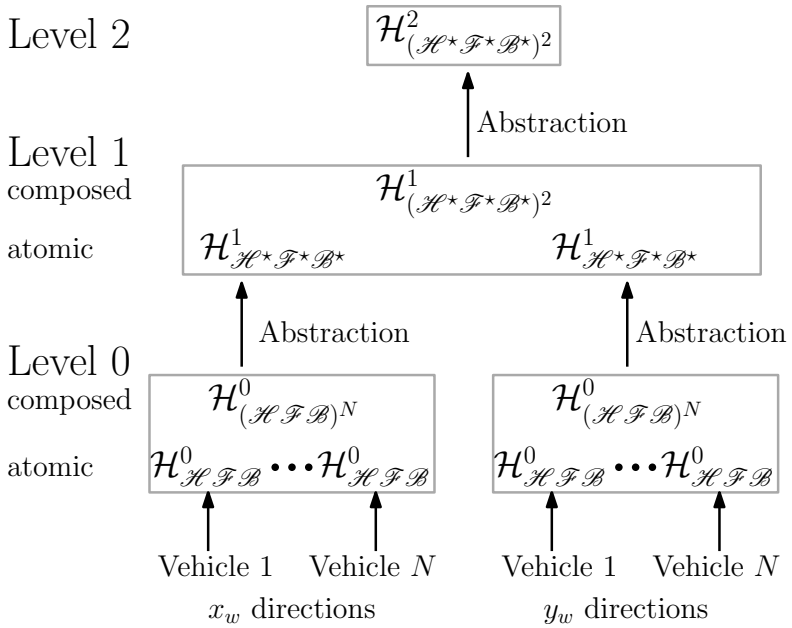

Figure 5.7: This figure illustrates the various maneuver automata at levels 0, 1, and 2 used to solve the formation control problem. At the bottom, each vehicle's x_w and y_w positional degrees of freedom are equipped with the atomic *Hold*, *Forward*, and *Backward* motion primitives. These motion primitives are parallel composed at level 0, abstracted to level 1 to ensure the correct formation behavior, composed at level 1, and finally abstracted to level 2 to yield a control policy that solves the overall behavior constrained reach-avoid problem.

translated by F^i. The behavior constraint $\widehat{\mathcal{B}} \subset \mathcal{B}$ consists of (5.10) applied to each physical direction independently among all the vehicles.

To solve the given problem, we use the formation MA designed in Section 5.7.2 to enforce the behavioral constraint. Then we construct a control policy to achieve the reach-avoid objective for the entire system. We assume that the vehicles are initially in formation. To ensure that the vehicles remain in L_f while executing the formation motion primitives, we also assume that for all $i, i' \in \{1, \ldots, N\}$, $i \neq i'$,

$$\max\{\ |F^i_j - F^{i'}_j|\ \mid\ j = 1, 2\ \} \geq 2. \tag{5.12}$$

First we describe the hierarchical maneuver automaton, see Figure 5.7. The N outputs corresponding to the x_w directions of the N vehicles are governed by the N-times parallel composed 0-MA $\mathcal{H}^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^N}$. We assign the 1-MA $\mathcal{H}^1_{\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star} \succeq \mathcal{H}^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^N}$ to enforce the formation constraint among these outputs. This process is repeated for the y_w direction. Next we parallel compose these two 1-MAs to obtain the 1-MA $\mathcal{H}^1_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2} = \|^2_{j=1}\mathcal{H}^1_{\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star}$. Finally, we design a 2-MA $\mathcal{H}^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2} \succeq \mathcal{H}^1_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}$ to serve as a control policy. The overall 2-HMA is $\mathcal{H}_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2} = \{\mathcal{H}^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^{2N}}, \mathcal{H}^1_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}, \mathcal{H}^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}\}$.

Now we describe how to construct the control policy, which only requires specifying the invariant

$I^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}(m^2)$ and transition function $X^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}(m^2)$. Observe that $M^1_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2} = \{\mathscr{H}^\star, \mathscr{F}^\star, \mathscr{B}^\star\}^2$ always consists of nine composed level 1 motion primitives. In contrast, the gridded output space $\mathbb{Z}^p$ and the underlying edges $E^1_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}$ for the transitions in $X^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}(m^2)$ grow exponentially with N. To make the computation tractable, we exploit knowledge of the formation to construct a policy based on a single reference vehicle, see Figure 5.8.

The second step shown in Figure 5.8 constructs a reduced grid

$$L_r = \{l \in \mathbb{Z}^2 \mid l + F^i \in L_{f,w},\ i = 1, \ldots, N\} \subset L_{f,w},$$

which creates virtual obstacles for the reference vehicle based on the other vehicle locations F^i and the physical obstacles. The reference vehicle is equipped with the level 0 *Hold*, *Forward*, and *Backward* motion primitives composed in each physical direction, $M^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^2} = \{\mathscr{H}, \mathscr{F}, \mathscr{B}\}^2$, which is essentially equivalent as planning for the overall system with the motion primitives $M^1_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}$. The third step shown in Figure 5.8 assigns these motion primitives over the reduced grid such that all paths lead to the reference vehicle goal box $l_g \in L_r$. Since motion primitives such as $(\mathscr{F}, \mathscr{F})$ may result in multiple possible next boxes, we use a non-deterministic Dijkstra algorithm to assign these motion primitives, as in Chapter 4. Also, since the algorithm expands outwards from the goal box, each box with a path to the goal is a valid initial condition. Other algorithms can also be implemented.

The above steps are carried out offline. The computation is generally very efficient for any number of vehicles N, as the generation of the reduced grid (step 2) scales linearly with N and the assignment of motion primitives of the reference vehicle using Dijkstra (step 3) scales only with the number of boxes in $L_r \subset \mathbb{Z}^2$ and not on N.

The control policy $\mathcal{H}^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}$ is implicitly described by the above steps. Each location on the reduced grid equipped with a motion primitive gives rise to a number of states in the invariant $I^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}(m^2)$. Referring to Step 3 of Figure 5.8, suppose the lower-left box is the origin of the reduced grid L_r and the outputs are ordered as shown. For example, since $(\mathscr{F}, \mathscr{H})$ is assigned at the lower-left box $(0, 0)$, we obtain the corresponding states $((0, 0, 0, 2), (\mathscr{F}^\star, \mathscr{H}^\star))$, $((1, 0, 0, 2), (\mathscr{F}^\star, \mathscr{H}^\star))$, and $((0, 1, 0, 2), (\mathscr{F}^\star, \mathscr{H}^\star))$ in $I^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}(m^2)$, which arise from all the combinations of the first and second vehicles having made progress in the x_w direction due to $\mathscr{F}^\star$. Similarly, each transition between locations on the reduced grid gives rise to multiple transitions in $X^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}(m^2)$ to account for the combinations in which a level 1 motion primitive can reach an enabling condition.

It is possible to verify that the behavior-constrained reach-avoid problem is solved by checking that $\mathcal{H}_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}$ is complete and well-posed, and checking conditions (i), (ii), and (iii) of Theorem 5.5.2.

A formal verification is lengthy, so we give an intuitive sketch. Completeness follows by construction and well-posedness follows from well-posedness of levels 0 and 1. Condition (i) follows because the reduced grid ensures that all vehicles are safe and (5.12) holds. Condition (ii) follows because all paths on the reduced grid reach the reference vehicle goal box. Condition (iii) follows by construction of the formation 1-MAs and since Lemma 5.4.13 ensures that behaviors are preserved when adding more hierarchical levels, $\mathcal{L}(\mathcal{H}^2_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2}) \subset \mathcal{L}(\mathcal{H}^1_{(\mathscr{H}^\star\mathscr{F}^\star\mathscr{B}^\star)^2})$.

5.8.3 Policy Generation for Formation Morphing

In the formation morphing objective, a collection of quadrocopters must reach a goal configuration in a known but cluttered environment. In this problem, there is no formation requirement. Instead, knowledge of the starting configuration is assumed to be known.

Once again, let $L_{f,w} \subset \mathbb{Z}^2$ be the feasible boxes in the world frame. For $i = 1, \ldots, N$, let $l_s^i \in L_{f,w}$ and $l_g^i \in L_{f,w}$ be the start and goal box for each vehicle, respectively. The joint feasible boxes L_f are obtained as before and the joint goal box L_g is obtained by stacking individual goals l_g^i. We do not specify a behavioral constraint, so $\widehat{\mathcal{B}} = \mathcal{B}$.

To solve this problem, we employ a variation of the formation MA designed in Section 5.7.2. Rather than coupling all the outputs to increment the same amount, we can similarly design additional level 1 motion primitives to cause each output to move a desired box increment of 0, 1 or -1. In this setting, we do not need to distinguish between the x_w and y_w directions specifically, so each level 1 motion primitive commands the desired level 0 motion primitives for each output among all the vehicles. When all outputs have completed their increment, the next level 1 motion primitive is chosen. Denoting this new HMA as $\mathcal{H}^1_{\text{cmd}}$, the level 1 motion primitives M^1_{cmd} command each output to increment 0, 1, or -1, so that there are 3^p such motion primitives. The hierarchical structure is shown in Figure 5.9, which may be contrasted to Figure 5.7. The $p = 2N$ outputs are governed by the parallel composed 0-MA $\mathcal{H}^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^{2N}}$. We assign the 1-MA $\mathcal{H}^1_{\text{cmd}} \succeq \mathcal{H}^0_{(\mathscr{H}\mathscr{F}\mathscr{B})^{2N}}$, and finally construct a control policy $\mathcal{H}^2_{\text{cmd}} \succeq \mathcal{H}^1_{\text{cmd}}$.

To construct the control policy, we use a variation of greedy search with the Manhattan distance as the heuristic, which is the sum of the distances in each direction to the goal box, see Figure 5.10 with $N = 2$ vehicles. The current box for each vehicle is initialized as the start box. For each vehicle, a neighboring direction from the current box that is both collision-free and decreases the Manhattan distance to the goal box is selected. Once a vehicle has chosen a neighboring direction, the other vehicles must take into account the selected neighboring direction in order to avoid collisions. Then the next current box is determined from the selected neighboring direction for each vehicle and the process continues until each

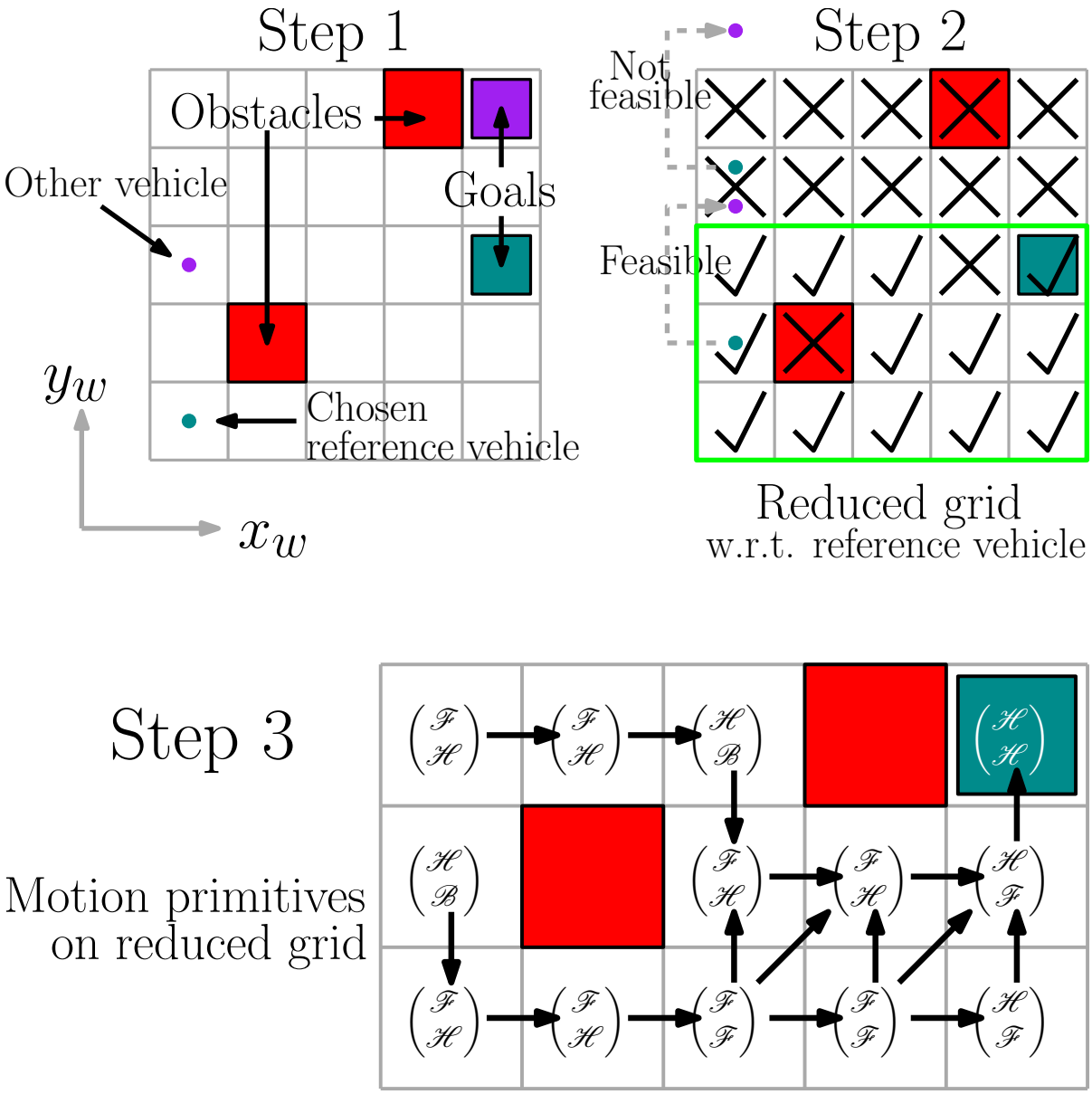

Figure 5.8: The procedure for computing a control policy is illustrated for the case of two vehicles ($N = 2$) over a physical 2D grid.

vehicle reaches the goal. The output of the algorithm is a sequence of motion primitives $m_j^1 \in M_{\text{cmd}}^1$ generated from each step. Referring to Step 1 in Figure 5.10, the vehicle on the left selects a positive increment in x_w and y_w, while the vehicle on the right selects negative increments. Following the order of outputs shown in Figure 5.7, the corresponding level 1 motion primitive is $(1, -1, 1, -1) \in M_{\text{cmd}}^1$. Similarly, Step 2 selects $(1, -1, 0, 0) \in M_{\text{cmd}}^1$.

The control policy's invariant $I_{\text{cmd}}^2(m^2)$ and transitions $X_{\text{cmd}}^2(m^2)$ are induced from the sequence $m_j^1 \in M_{\text{cmd}}^1$ and starting boxes of the vehicles, similar to the previous section. For example, if the origin is the lower-left box, Step 2 with the selected motion primitive $m_2^1 = (1, -1, 0, 0) \in M_{\text{cmd}}^1$ gives $((1, 3, 3, 1), m_2^1)$, $((2, 3, 3, 1), m_2^1)$, and $((1, 2, 3, 1), m_2^1)$ in $I_{\text{cmd}}^2(m^2)$, arising from the combinations in which the first and second vehicle make progress moving right and left respectively.

Since the heuristic guides the vehicles to the goals and collisions are easily accounted for at each step, the control policy can be generated quickly offline using greedy search. Its complexity scales linearly with the number of the vehicles N. On the downside, it may fail to generate a control policy if the physical obstacles have long or non-convex shapes, or if the goals l_g^i are not spaced out sufficiently. Our goal here was to provide a simple baseline algorithm to illustrate the applicability of planning with more abstract motion primitives. We envision that more sophisticated discrete planning algorithms can potentially be utilized in order to improve the ability to find a feasible solution, or even an optimal one.

When greedy search produces a control policy, it can be verified that the reach-avoid problem is solved. The control policy is complete by construction and well-posed because levels 0 and 1 are well-posed. Condition (i) of Theorem 5.5.2 follows because greedy search only selects collision-free neighboring directions, condition (ii) follows because greedy search found a path from the start to goal boxes for all the vehicles, and condition (iii) follows trivially.

5.8.4 Sequence of Reach-Avoid Objectives

As an extension to more complex task specifications, we consider a sequence of reach-avoid objectives. First we construct a level 2 control policy as in the previous sections for each individual reach-avoid objective. Each policy is then considered as an individual level 2 motion primitive and feasible concatenations among these motion primitives are constructed. Finally, a level 3 control policy is created that assigns the sequencing of the reach-avoid tasks. In this way, we may intermingle formation flight and formation morphing seamlessly within a single framework.

Several technical points are now discussed. Observe that both control policy methods from the previous sections generate level 2 motion primitives with no enabling conditions; that is, all vehicles

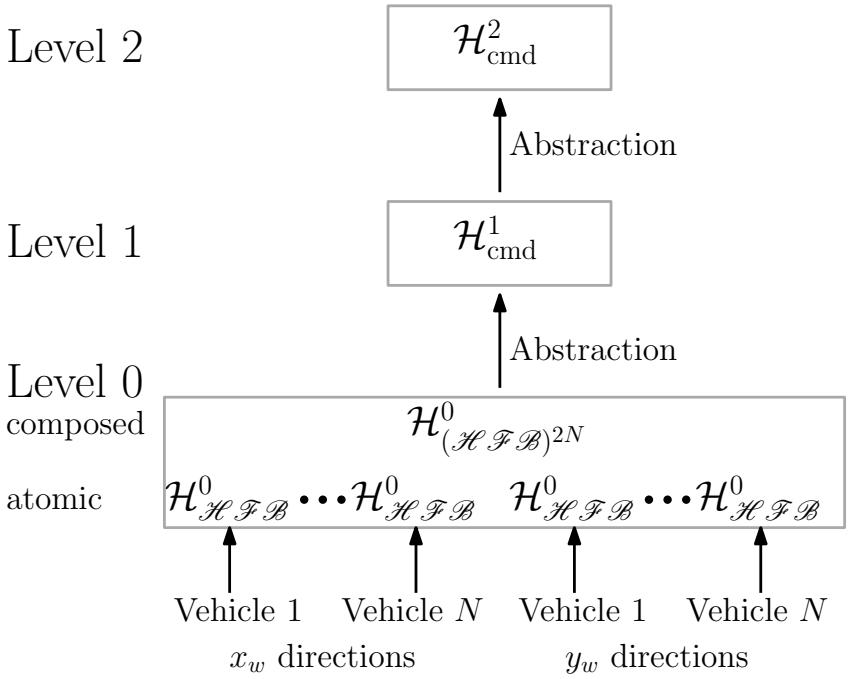

Figure 5.9: This figure illustrates the various maneuver automata at levels 0, 1, and 2 used to address the morphing control problem. At the bottom, each vehicle's positional degrees of freedom is equipped with the atomic *Hold*, *Forward*, and *Backward* motion primitives. These motion primitives are parallel composed at level 0, abstracted to level 1, and finally abstracted to level 2 to yield a control policy that morphs the system from one formation to another.

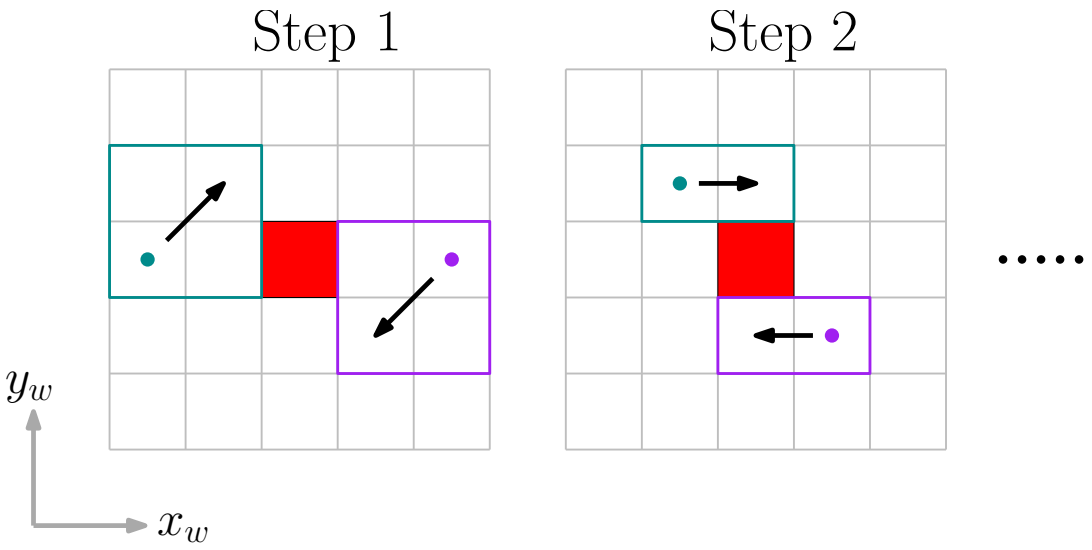

Figure 5.10: Two agents ($p = 4$) must swap places in a constrained planar space with an obstacle. The policy encoding a solution constitutes a level 2 motion primitive and is composed of level 1 motion primitives M^1_{cmd} that command each output an increment of 0, 1, or -1. As each step commands a level 1 motion primitive, it is easy to enforce safety.

reach and stabilize within their goal boxes in accordance with the reach condition (ii) of Problem 5.3.1. Thus the reach condition (ii) is relaxed so that trajectories only need to reach the goal once: there exists $T \geq 0$ such that $y(T, x_0) \in \bigcup_{l \in L_g} Y_l$. Next, in order to concatenate two level 2 motion primitives, the goal box is removed from the first motion primitive and appears instead at the beginning of the second motion primitive. To satisfy well-posedness (viii), internal transitions of a level 2 motion primitive that previously led to a removed goal box are now required to be made into external transitions to other level 2 motion primitives. These edges between level 2 motion primitives are formally added using the union of HMA from Section 5.6.2 to ensure a feasible construction.

5.9 Experimental Results

Our experimental platform is the same as described in Section 4.8.3. We showcase three different scenarios on up to eight quadrocopters, using a combination of the control policy strategies discussed earlier. These scenarios illustrate the ease, intuitiveness, and effectiveness with which our multi-level hierarchical approach can handle complex specifications involving many vehicles. A video showing the results is found at `http://tiny.cc/hier-moprim`. In our code implementation, the user can select the number of vehicles, the grid parameters, the obstacle locations, the goal locations, and other parameters such as the formation offsets or whether to morph instead. Mixing the two objectives in a general way was not implemented.

The control policy strategies from the previous section are computed offline. In runtime, Algorithm 1 is used to correctly implement the motion primitives at each level. Since in practice the state estimates $x \in \mathbb{R}^n$ are finitely sampled, the level 0 events s^0 are calculated by measuring the current box $l \in \mathbb{Z}^p$ such that $h(x) \in Y_l$ rather than checking guard sets. The determination of higher level events on line 6 of Algorithm 1 can be efficiently implemented for our designed motion primitives using a generalization of (5.11).

5.9.1 Room Transition in Line Formation

In this scenario, six vehicles are placed in a line formation, see Figure 5.11. The physical space is partitioned into a $7 \times 13 \times 1$ grid, as shown in Figure 5.12. In the first trial, the vehicles must fly from the left side of the room to the right side in the (x_w, y_w) plane. An obstacle lies on the direct path between the start and goal boxes only for the bottom vehicle. To maintain the formation, consequently all of the vehicles must navigate in the y_w direction so that the bottom vehicle avoids collision. The level 2 control policy is generated using the method in Section 5.8.2. The computation time using the

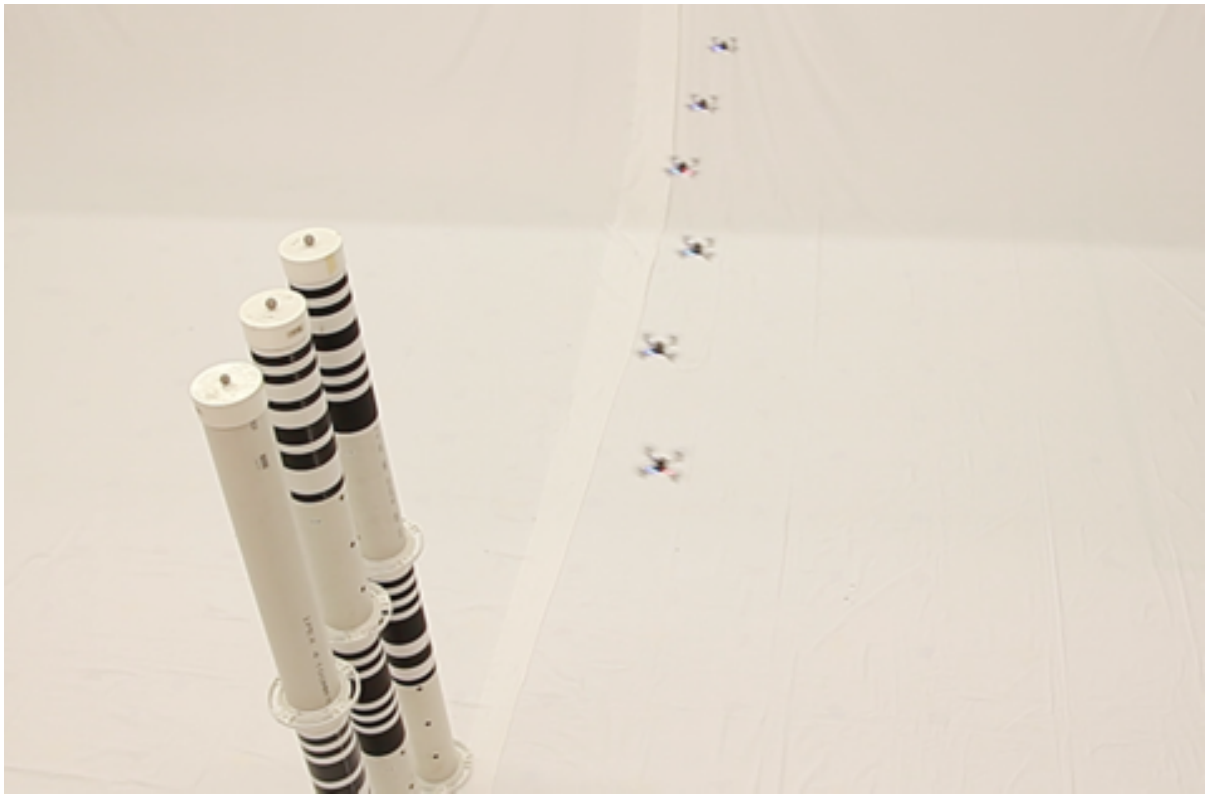

Figure 5.11: Snapshot of the room transition in line formation scenario.

Dijkstra algorithm is around 0.08 seconds. Figure 5.12 shows the paths followed in the (x_w, y_w) plane and the trajectories as a function of time. The alternating grey lines on the trajectories as a function of time indicate the size of the boxes on the grid.

The second trial illustrates several notable features compared to the first trial. Here, the vehicles must fly repeatedly from one side of the room to the other. We formulate this as a sequence of two reach-avoid objectives, following Section 5.8.4. The first level 2 motion primitive, denoted $m^2_{\rightarrow}$, causes all vehicles to reach the right side, as in the first trial. The second level 2 motion primitive, $m^2_{\leftarrow}$, causes all vehicles to reach the left side. The motion primitive $m^2_{\rightarrow}$ concatenates to $m^2_{\leftarrow}$ and $m^2_{\leftarrow}$ concatenates back to $m^2_{\rightarrow}$. Figure 5.13 shows the resulting motion and annotates the duration of each motion primitive $m^2_{\rightarrow}$ and $m^2_{\leftarrow}$. Also the vehicles started from a different location compared to the first trial. The motion primitives $m^2_{\rightarrow}$ and $m^2_{\leftarrow}$ provide paths to the goal from a wide range of starting locations since they were computed using a Dijkstra algorithm.

The second trial can be considered a case of a heterogenous collection of vehicles, which may arise, for example, when one vehicle is defective. The bottom vehicle is equipped with a 0-MA in the x_w direction that causes a slower speed during *Forward* and *Backward* relative to the other vehicles. We achieve this effect in this output by changing the maximum acceleration design parameter u^* in the 0-MA $\mathcal{H}^0_{\mathscr{H}\mathscr{F}\mathscr{B}}$

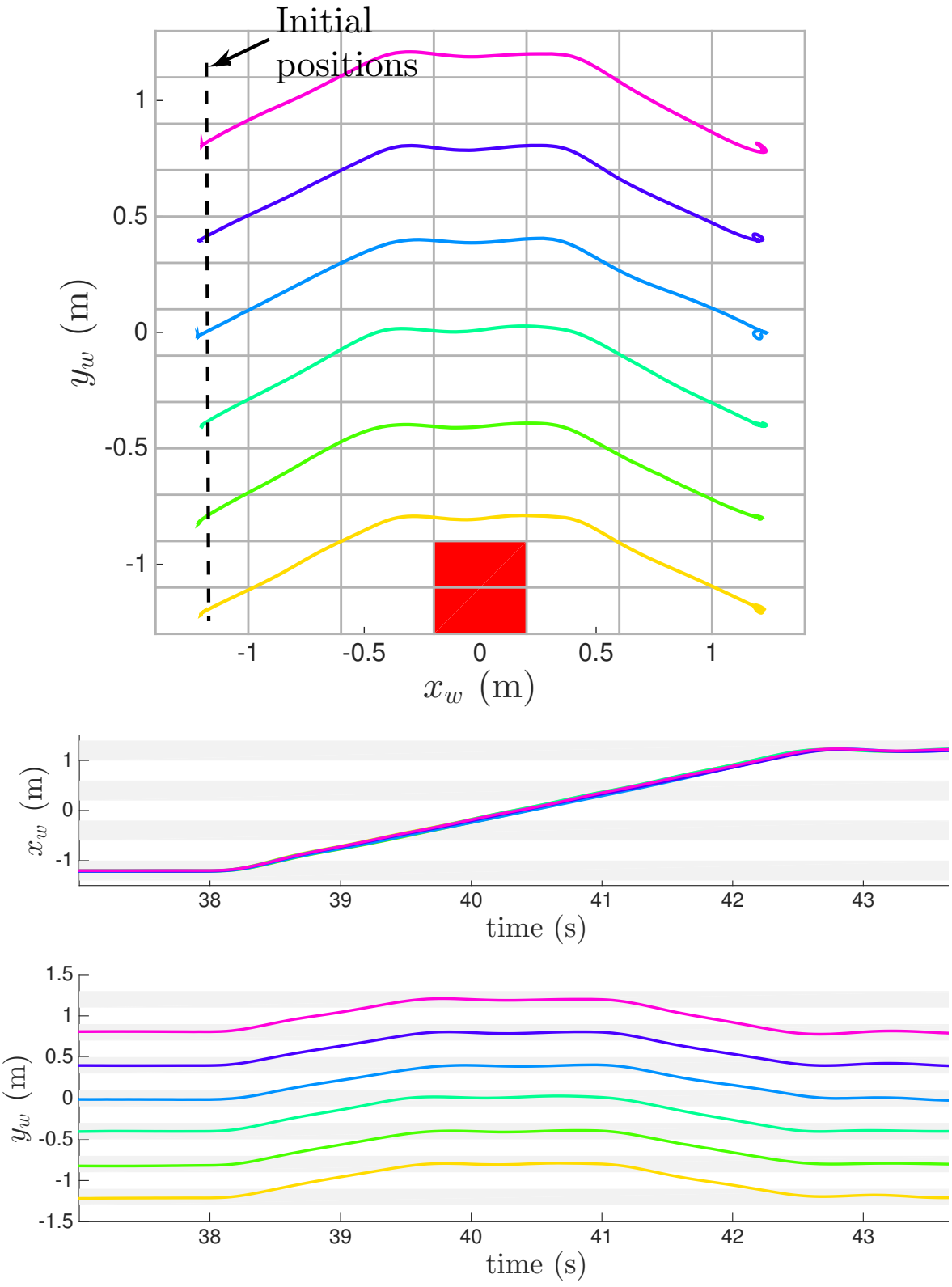

Figure 5.12: Experimental results for line formation in the first trial. The vehicles transition from the left to the right side of the room.

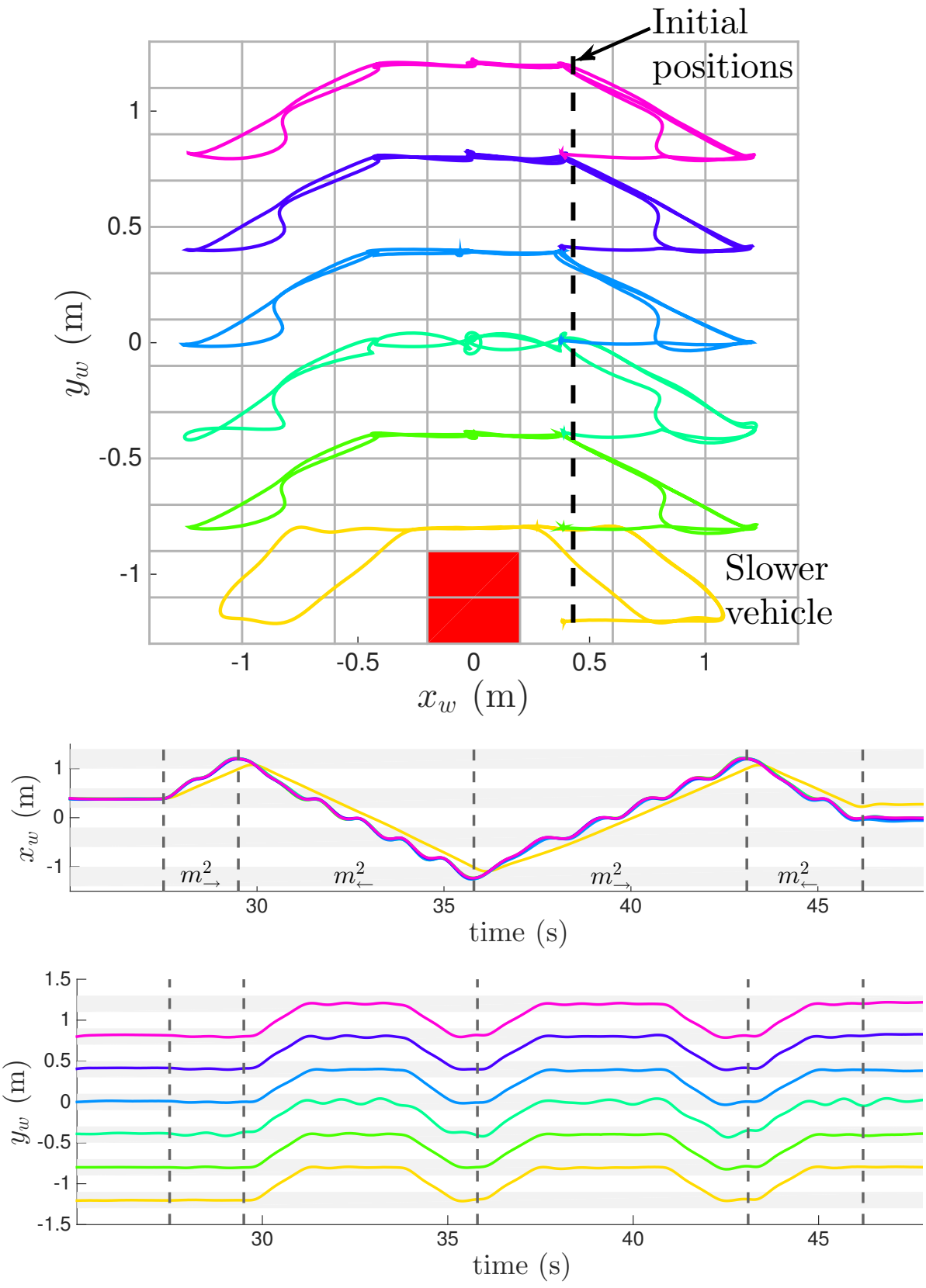

Figure 5.13: Experimental results for line formation in the second trial. The vehicles transition back and forth continuously with the bottom vehicle moving slower in x_w.

from Section 4.7.2. The resulting behavior is shown in Figure 5.13. In particular, the x_w trajectories as a function of time show that the bottom vehicle (yellow) moves at its slower nominal speed, while the other vehicles move at their faster speed and then stop as necessary in order to maintain the formation. This behavior emerges automatically from the design of the level 1 formation motion primitives. This feature is extremely robust, as the computation of the motion primitives $m^2_{\rightarrow}$ and $m^2_{\leftarrow}$ does not require knowledge that the system is heterogenous, nor does it require any information on the implementation of the underlying level 0 motion primitives, such as timing estimates to transition through boxes. Our entire framework of motion primitives is based upon feedback, both at the continuous and discrete levels, and does not require specifying any timed reference trajectories.

5.9.2 Morphing

In this scenario, eight vehicles morph from one configuration to another on a $9 \times 5 \times 8$ grid. Figure 5.14 shows a snapshot of the initial and goal configurations in the (x_w, z_w) plane, while Figure 5.15 shows the trajectories projected onto the (x_w, y_w) plane and as a function of time. The initial configuration is a spiral that forms a circular-shape in the (x_w, y_w) and an s-shape in the (x_w, z_w) plane. The goal configuration is a v-shape in the (x_w, z_w) plane which is situated in the middle row of boxes on the y_w axis. The greedy search algorithm described in Section 5.8.3 was used (generalized to 3D), resulting in a computation time of around 3.5 milliseconds. The computed control policy consists of only 8 steps m^1_j, $j = 1, \ldots, 8$, as all the vehicles can typically move into some neighboring direction towards their goal box at each step.

Notice that at each step, the underlying level 1 motion primitives M^1_{cmd} cause vehicles to wait once they have reached their neighboring box until all the other vehicles have reached their neighboring box before moving onto the next step. This feature accounts for the motion of all the vehicles so that collisions are easy to avoid. Although the individual waiting time for a given vehicle is typically short, the resulting motion is not the most efficient, especially for vehicles that are far away from all the other vehicles. Planning with these motion primitives affords a reasonable compromise between solution quality and efficient computation of a policy for a large number of vehicles, while ensuring strict safety guarantees.

5.9.3 Various Formations in a Cluttered Environment

In the final scenario, we combine both formation flight and formation morphing on a $14 \times 14 \times 5$ grid. Figure 5.16 demonstrates the flow of tasks, each of which is specified as a reach-avoid objective. Following

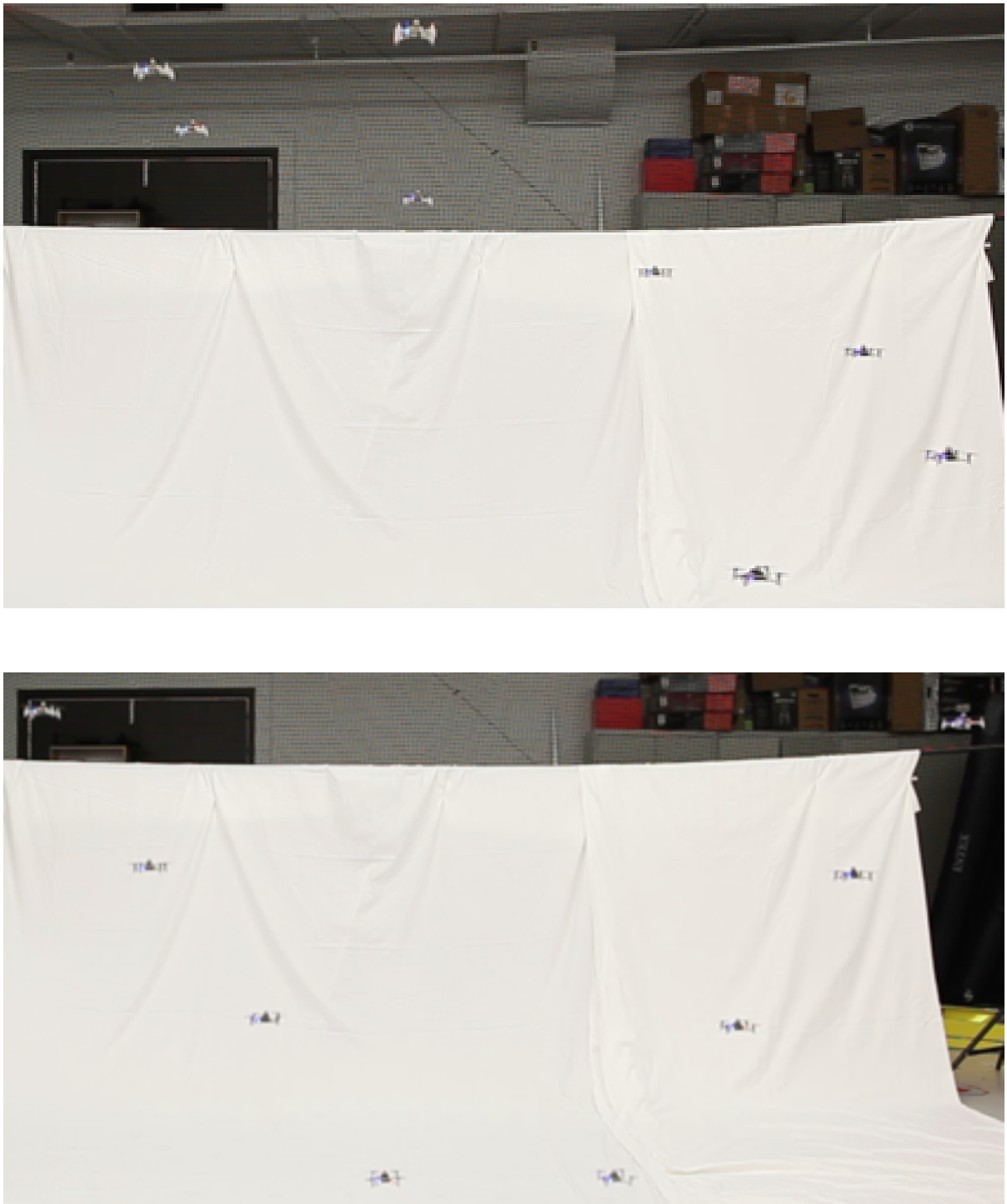

Figure 5.14: Snapshot of the morphing scenario. The top shows the initial configuration and the bottom shows the goal configuration.

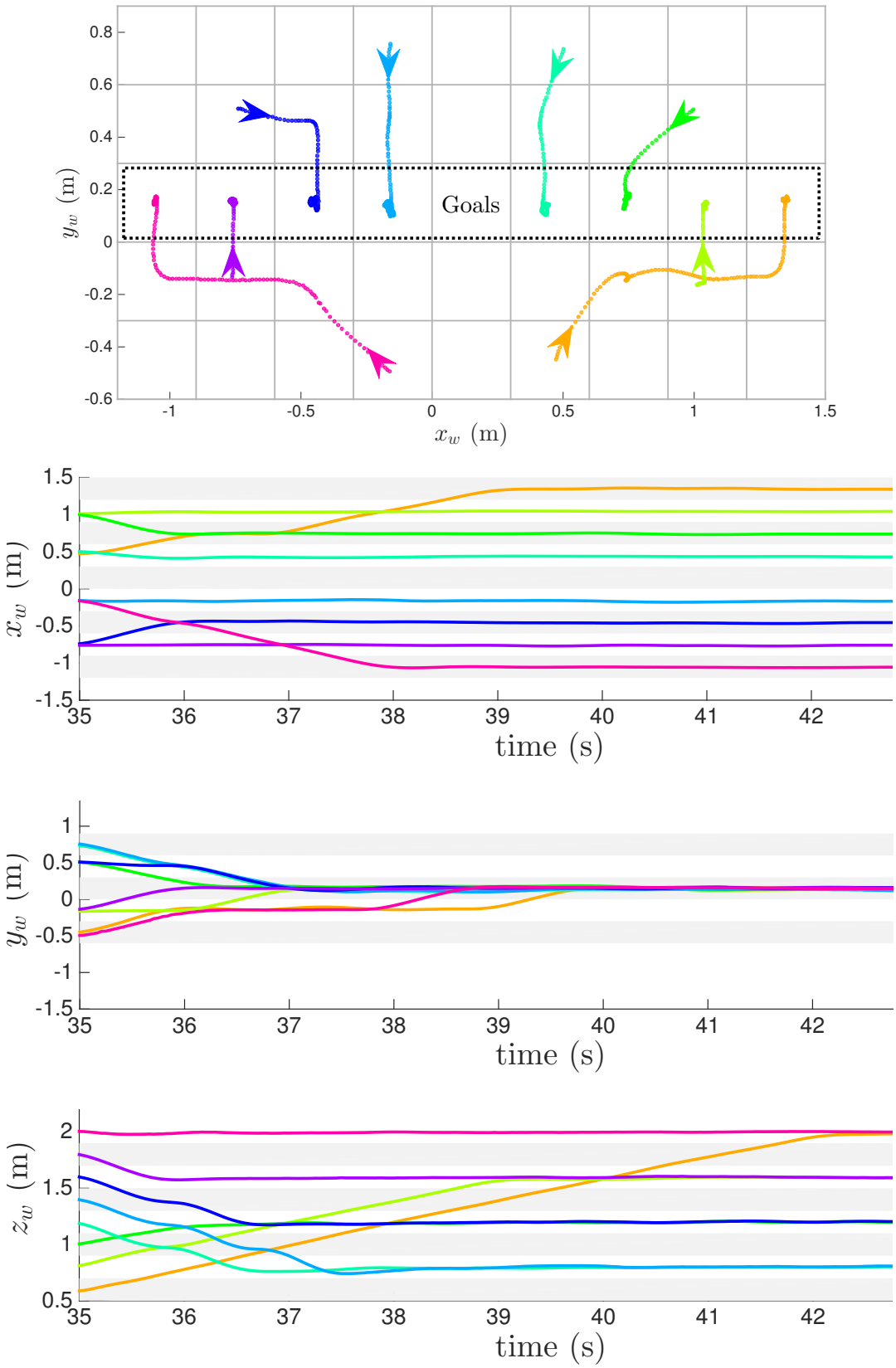

Figure 5.15: Experimental results for the morphing scenario. In the top figure, the arrowheads indicate the direction of motion.

Section 5.8.4, each objective is carried out by a level 2 motion primitive, denoted m_j^2, $j = 1, \ldots, 4$. The first objective is formation flight in a 2×4 arrangement, requiring to navigate past the three pillar obstacles. The second objective morphs into an octogonal formation. The third objective is formation flight, requiring to encircle a large obstacle. Finally, the fourth objective requires rotating about the encircled obstacle. Figure 5.17 shows a few snapshots, Figure 5.18 shows the trajectories in 3D, and Figure 5.19 shows the trajectories as a function of time. The execution of the level 2 motion primitives m_j^2 are also annotated on Figures 5.18 and 5.19.

The computation time for each of the level 2 primitives m_1^2 and m_3^2 for formation flight was about 1.2 seconds using the Dijkstra algorithm (generalized to 3D) from Section 5.8.2. The second objective is addressed using greedy search from Section 5.8.3 and is completed rather trivially in just two steps. The fourth objective is also formulated as a morphing problem; while greedy search could have been used, we chose to design the steps m_j^1 manually. The construction is an extension of the idea shown in Figure 5.10 from two to eight vehicles, which results in 16 steps for each vehicle to complete a full revolution.

This scenario illustrates the richness of the possible motions that can be encoded with our hierarchical motion primitives framework and also the ease and intuitiveness with which many vehicles can be controlled. The proposed level 1 motion primitives for motion planning introduced in Section 5.7 and the planning strategies for quadrocopters introduced in Section 5.8 offer a taste of what is potentially possible. It is not hard to see that other interesting behavioral constraints such as a wave pattern or contraction and dilation effect can also be tackled efficiently using hierarchical motion primitives. The use of hierarchy is advantageous as complex behaviors can be represented in a more economical and intuitive way.

The significance of our HMA framework is that it provides general but provably correct guidelines for well-designed hierarchical motion primitives in the context of motion planning. In our applications, we hand-designed the underlying level 0 and level 1 motion primitives using our intuition, created control policies at level 2 using standard planning algorithms, and even combined these at level 3. A different application may reuse some of the strategies we have proposed in this chapter. Alternatively, one may construct new lower level motion primitives, employ an arbitrary number of hierarchical levels, and adapt different planning algorithms at the top level or even plan among several levels. The strength of our approach lies in its modularity, as it affords the designer the ability to customize each of these components.

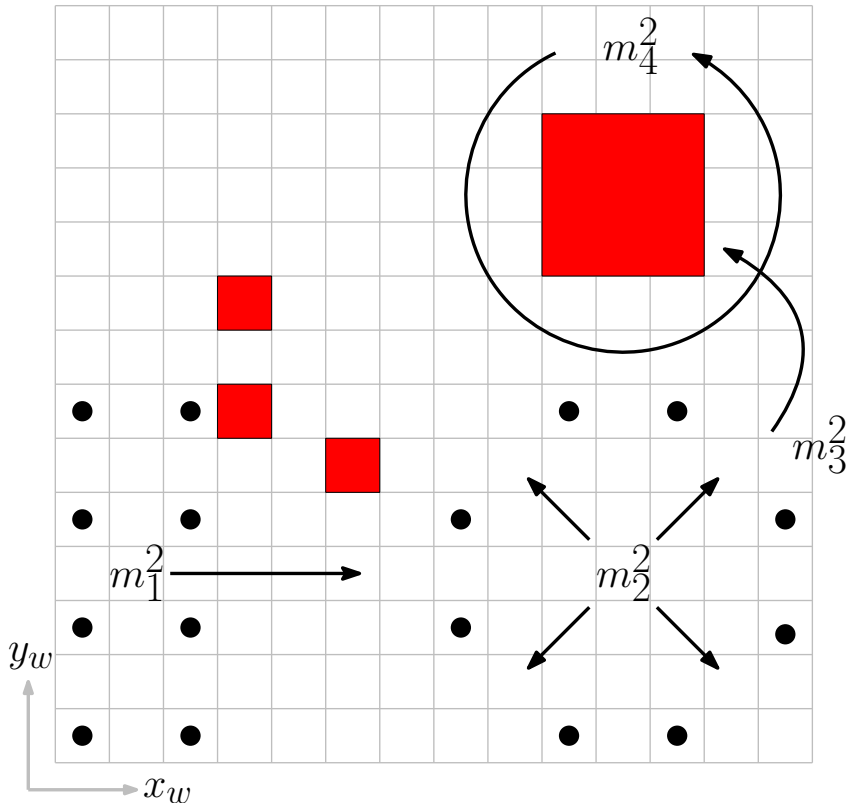

Figure 5.16: Objectives for experiment involving multiple formations.

5.10 Conclusion

We have presented a multi-level hierarchical framework for motion planning of a large collection of agents. Central to our methodology is the notion of a hierarchical maneuver automaton (HMA), which allows for a recursive construction of motion primitives, starting from a concrete implementation using low-level feedback controllers, to an increasing level of abstraction that specifies the rules for connecting lower-level motion primitives. We have characterized the solvability of reach-avoid problems with behavior constraints using our framework. To demonstrate practical applicability, we have designed a versatile library of hierarchical motion primitives to address formation flight and morphing. The effectiveness of the approach is validated experimentally on a collection of quadrocopters.

We provide a brief discussion on some of the limitations of the proposed hierarchical methodology and potential future work. As this work is an extension of Chapter 4, some of the same limitations are inherited, so we discuss some of the new complications that arise. First, just as we did not present an automated method of generating motion primitives (at level 0) in Chapter 4, we did not present an automated way of constructing motion primitives at higher levels. Even if we assume the existence of useful level 0 motions, such as those we have provided based on integrator systems, one of the main

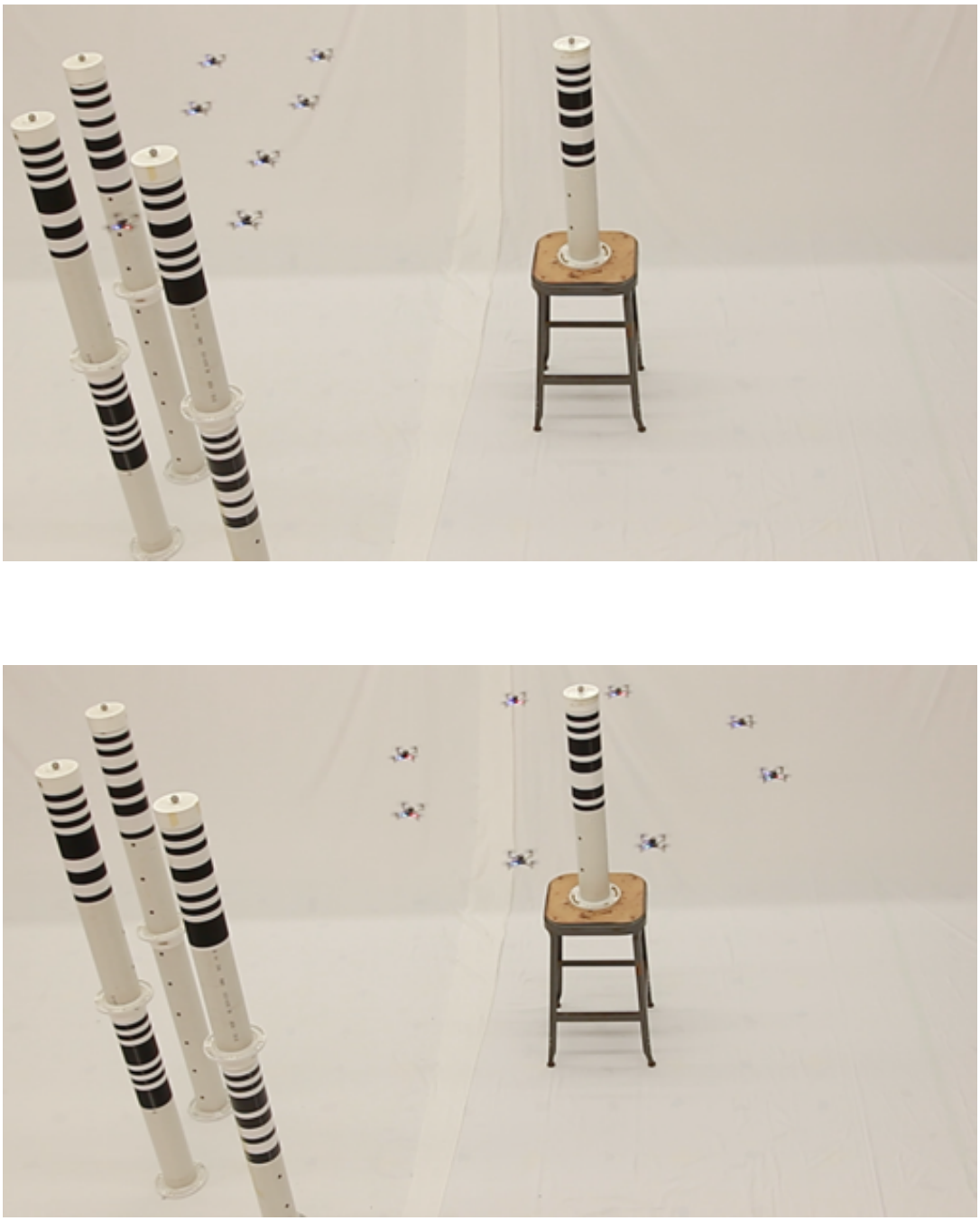

Figure 5.17: Snapshots of the scenario intermingling formation flight and morphing.

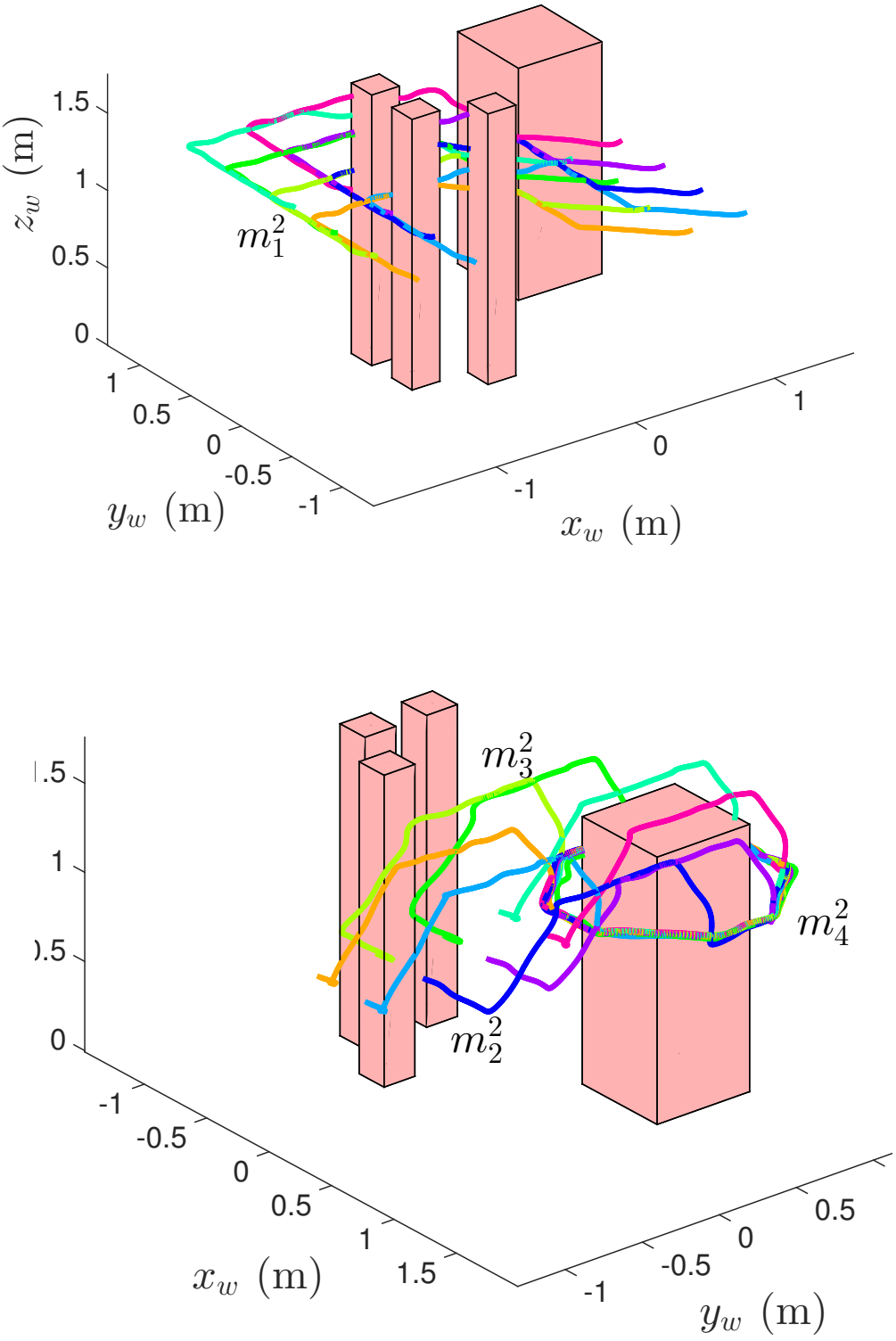

Figure 5.18: Experimental results involving multiple formations, showing 3D trajectories.

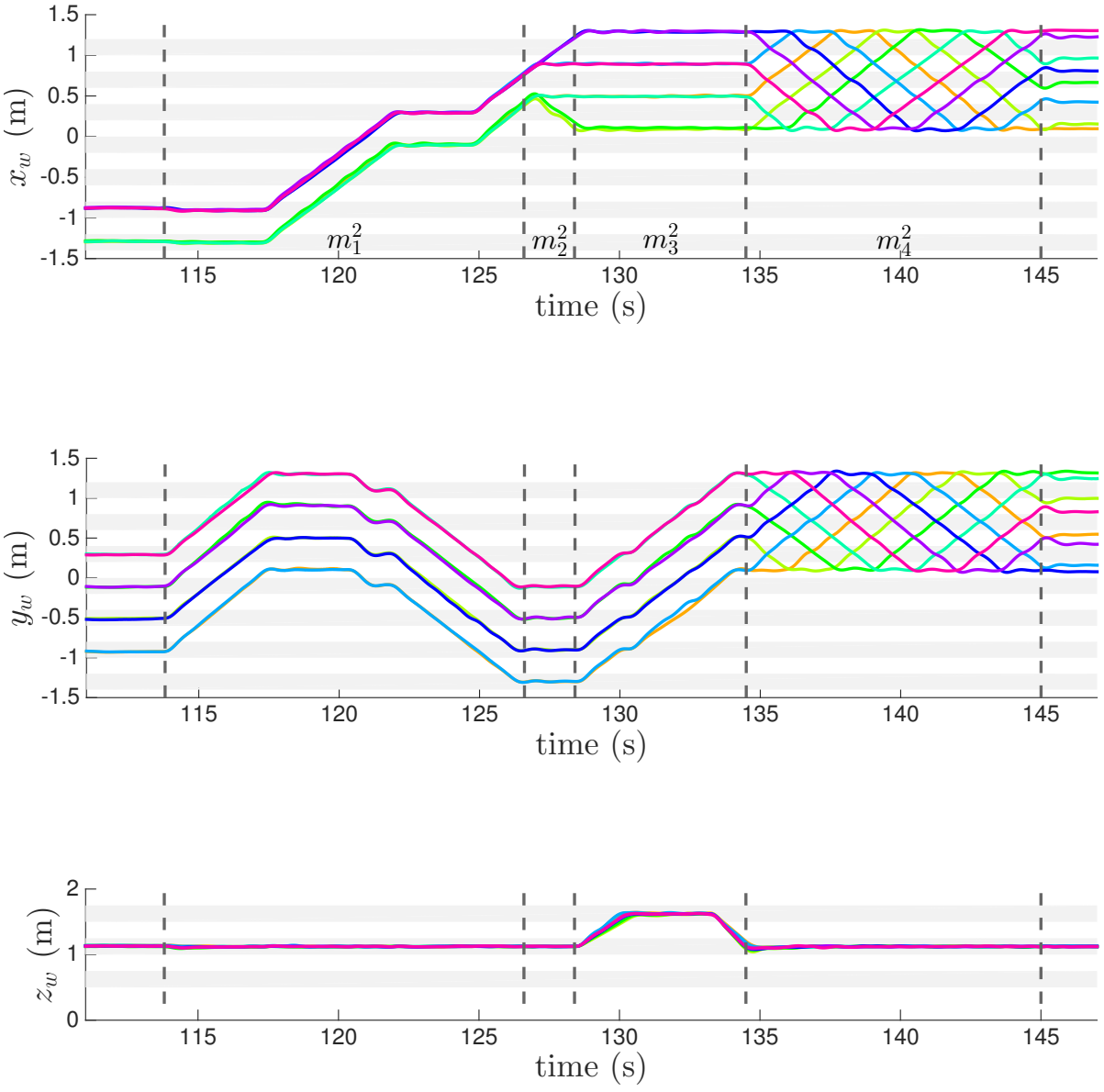

Figure 5.19: Experimental results involving multiple formations, showing trajectories vs. time.

complications is that there is an infinite number of possible feasible motion primitives that can be defined at higher levels (of course, only a small number should actually be designed). Moreover, the number of hierarchical levels is also a free parameter. While we have left these aspects to the discretion of the designer and have shown some examples of significant computational savings, planning in a free hierarchical fashion can generally be quite daunting. Clearly the use of intuition plays an important role, and so the development of some heuristics could be helpful. Overall, it may be interesting to investigate methods to automatically generate a hierarchy (or in some sense, an optimal hierarchy) that solves a given instance of the behavior-constrained reach-avoid problem.

Second, as in Chapter 4, we made an effort in this chapter to characterize the existence of Zeno executions in the hierarchical setting. Our constraint on the reach property in the main theorem is quite conservative in order to bypass the difficulty. For example, in the hierarchical setting it is quite possible to reach a collection of goal boxes and to cycle through these boxes. If the underlying level 0 motion primitives are not well designed, a Zeno execution may exist.

Finally, we note the overall complexity of employing a hierarchical approach. While we have provided sufficient conditions for the solvability of the behavior-constrained reach-avoid problem, it can be argued that an efficient verification of these conditions is in many cases quite prohibitive. Indeed, we did not supply formal verifications of well-posedness and the conditions of the main theorem in our applications; while these results are intuitive and verified experimentally, a complete proof is quite challenging to produce. Moreover, we did not provide a rigorous definition of the parallel composition and union procedures at higher levels because our efforts have suggested that it becomes a combinatorial nightmare. The complexity of the original problem is in some ways transferred to the complexity of encoding the hierarchy.

Chapter 6

Analysis of Formation Motion Primitives

6.1 Introduction

In this chapter, we examine in greater detail a characteristic of the formation motion primitives introduced in Chapter 5. The presentation of this analysis is motivated by the pleasant observation that a purely discrete design of higher level motion primitives could lead to remarkable properties at the continuous level, even though not explicitly designed for.

Consider the hierarchical maneuver automaton $(\mathcal{H}^0_{(\mathscr{H}\,\mathscr{F}\,\mathscr{B})^p}, \mathcal{H}^1_{\mathscr{H}^\star\,\mathscr{F}^\star\,\mathscr{B}^\star})$ for formation control from Section 5.7, where $p \geq 1$ is the number of outputs. We have observed that a "hybrid" limit cycle emerges in any hierarchical execution that continually assigns the level 1 motion primitive *Abstract Forward*, $\mathscr{F}^\star$, as first pointed out in Remark 5.7.2. By symmetry, the same would be observed for *Abstract Backward*, so it is not explicitly considered. Let us consider some examples and then motivate the implications.

Due to the modularity of our approach, we may use either the single integrator design from Section 4.7.1 or the double integrator design from Section 4.7.2 for the implementation of $\mathcal{H}^0_{\mathscr{H}\,\mathscr{F}\,\mathscr{B}}$. For simplicity, let us consider the case of single integrators with a unity canonical box, $Y^* = \prod_{i=1}^{p}[0,1]$. First suppose that $p = 1$. In this trivial case, we keep applying $\mathscr{F}^\star$ at level 1, which simply implements $\mathscr{F}$ at level 0. There is only one resulting (hybrid) trajectory, which consists of repeating the following: the position y increases with constant speed, and the position is reset to $y = 0$ when $y = 1$.

Next suppose that $p = 2$. Applying $\mathscr{F}^\star$ at level 1 consists of applying the level 0 motion primitives $(\mathscr{F}, \mathscr{F})$, $(\mathscr{F}, \mathscr{H})$, and $(\mathscr{H}, \mathscr{F})$, see Figure 6.1. Trajectories start in the lower left region with $(\mathscr{F}, \mathscr{F})$

and may progress in three ways to complete $\mathscr{F}^\star$. First, a trajectory may enter the lower right region with $(\mathscr{H}, \mathscr{F})$, eventually leading to upper right region, which is shown as blank to indicate that $\mathscr{F}^\star$ repeats by reseting the trajectory to the lower left region with $(\mathscr{F}, \mathscr{F})$. Second, a trajectory may enter the top left region with $(\mathscr{F}, \mathscr{H})$, eventually leading again to the upper right region. Third, a trajectory may exactly enter the upper right region. It should be clear that the reset action, which consists of subtracting 1 in each component at the completion of the level 1 motion primitive $\mathscr{F}^\star$, is employed as a convenience to analyse these hybrid trajectories in a compact domain without the need for re-expressing translated versions of the same equations. The hybrid nature of these trajectories refers to the switching among different closed-loop vector fields, corresponding to the different level 0 motion primitives.

Referring again to Figure 6.1, we see for this particular example that the hybrid trajectories seem to converge to the "diagonal" trajectory. Moreover, if we interpret the two outputs as corresponding to the motion of two vehicles along the same axis, this means that the individual output component trajectories "synchronize" as $\mathscr{F}^\star$ is repeatedly executed, in the sense that both vehicles take the same amount of time to complete their respective initial level 0 $\mathscr{F}$ motion primitive of $\mathscr{F}^\star$. In the context of formation flight, this synchronization property is highly desirable because the resulting motion would be smoother, as opposed to the two vehicles taking turns jerking forward. Our claim is that the presence of a hybrid limit cycle is a general phenomenon that occurs for any number of outputs for both the single and double integrator cases. This claim is proven for the single integrator case and is left as an open problem for the double integrator case.

Although we shall not provide a detailed survey of existing literature on limit cycles of hybrid systems, a notable example can be found in [48], which investigated limit cycles in bipedal locomotion by applying the concept of a Poincaré return map. We follow a fundamentally similar approach, employing a customized analysis based on applying the contraction principle from real analysis. Given that our design of the *Abstract Forward* motion primitive was first introduced by the author, the result stated in this chapter is a novel contribution.

This chapter is organized as follows. In Section 6.2 we recall the associated feedback controllers for the motion primitives and derive some useful expressions. Section 6.3 presents the main results. We conclude in Section 6.4.

6.2 Preliminaries

Next we recall the controller designs for *Hold* ($\mathscr{H}$) and *Forward* ($\mathscr{F}$) for both the single and double integrators (in both cases, the number of outputs is $p = 1$). The free parameters are the length of Y^*,

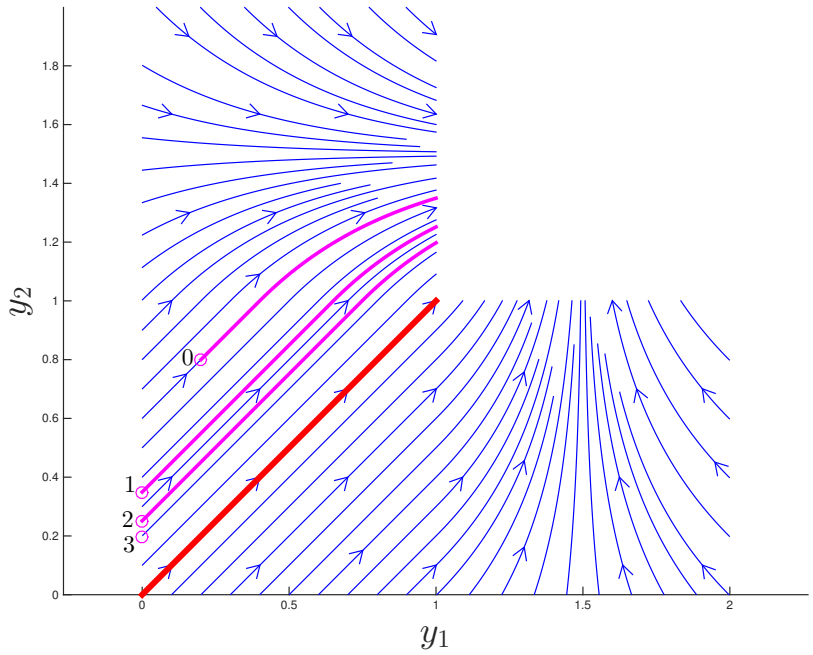

Figure 6.1: Hybrid trajectories of Abstract Forward ($\mathscr{F}^\star$) tend to a limit cycle (red). A hybrid trajectory (magenta) is shown over three applications of $\mathscr{F}^\star$.

$d > 0$, and the maximum control effort, $u^* > 0$. In particular, we shall write some expressions that will assist us in the stating and proving the main result.

6.2.1 Single Integrator

First consider the single integrator, $\dot{x} = u$. From Section 4.7.1, the controllers can be expressed as

$$u^1_{\mathscr{H}}(x) = (-2u^*/d)(x - d/2), \quad u^1_{\mathscr{F}}(x) = u^*.$$

It will be useful to normalize the dynamics for each controller so that the box length is unity. Define $\hat{x} = (1/d)x \in \mathbb{R}$ and $\rho = u^*/d$. Then the dynamics become $\dot{\hat{x}} = u$, with the new controllers

$$\hat{u}^1_{\mathscr{H}}(\hat{x}) = -2\rho(\hat{x} - 1/2), \quad \hat{u}^1_{\mathscr{F}}(\hat{x}) = \rho.$$

In this form, the explicit solutions to the ODEs $\dot{\hat{x}} = \hat{u}^1_{\mathscr{H}}(\hat{x})$ and $\dot{\hat{x}} = \hat{u}^1_{\mathscr{F}}(\hat{x})$ for *Hold* and *Forward* respectively, with initial condition $\hat{x}_0 \in \mathbb{R}$, are especially easy to compute:

$$\hat{x}_{\mathscr{H}}(t) = (\hat{x}_0 - 1/2)\exp(-2\rho t) + 1/2, \quad \hat{x}_{\mathscr{F}}(t) = \rho t + \hat{x}_0. \tag{6.1}$$

6.2.2 Double Integrator

Now consider the double integrator, $\dot{x_1} = x_2$ and $\dot{x_2} = u$. Let $v^* = \sqrt{du^*}$ be the derived maximum speed. From Section 4.7.2, the controllers can be expressed as

$$u^2_{\mathscr{H}}(x) = (-2u^*/d)(x_1 - d/2) + (-2u^*/v^*)x_2, \quad u^2_{\mathscr{F}}(x) = (-2u^*/v^*)(x_2 - v^*/2).$$

Once again, we normalize the equations, letting $\hat{x} = \begin{bmatrix} 1/d & 1/v^* \end{bmatrix} x \in \mathbb{R}^2$ and $\nu = \sqrt{\rho} = \sqrt{u^*/d}$. Then the dynamics become $\dot{\hat{x}}_1 = \nu\hat{x}_2$ and $\dot{\hat{x}}_2 = u$, with the new controllers

$$\hat{u}^2_{\mathscr{H}}(\hat{x}) = -2\nu(\hat{x}_1 - 1/2) - 2\nu\hat{x}_2, \quad \hat{u}^2_{\mathscr{F}}(\hat{x}) = -2\nu(\hat{x}_2 - 1/2).$$

In this form, the explicit solutions to the ODEs are still easy to compute. For *Hold*, the solution to $\dot{\hat{x}}_1 = \nu\hat{x}_2$ and $\dot{\hat{x}}_2 = \hat{u}^2_{\mathscr{H}}(\hat{x})$, with initial condition $\hat{x}_0 \in \mathbb{R}^2$, is

$$(\hat{x}_{\mathscr{H}})_1(t) = \exp(-\nu t)\left[c_1\cos(\nu t) + c_2\sin(\nu t)\right] + 1/2, \tag{6.2}$$

$$(\hat{x}_{\mathscr{H}})_2(t) = \exp(-\nu t)\left[(-c_1 + c_2)\cos(\nu t) + (-c_1 - c_2)\sin(\nu t)\right]. \tag{6.3}$$

where $c_1 = (\hat{x}_0)_1 - 1/2$ and $c_2 = (\hat{x}_0)_1 + (\hat{x}_0)_2 - 1/2$. For *Forward*, the solution to $\dot{\hat{x}}_1 = \nu\hat{x}_2$ and $\dot{\hat{x}}_2 = \hat{u}^2_{\mathscr{F}}(\hat{x})$, with initial condition $\hat{x}_0 \in \mathbb{R}^2$, is

$$(\hat{x}_{\mathscr{F}})_1(t) = 1/2\left[(1/2 - (\hat{x}_0)_2)(\exp(-2\nu t) - 1) + \nu t\right] + (\hat{x}_0)_1, \tag{6.4}$$

$$(\hat{x}_{\mathscr{F}})_2(t) = ((\hat{x}_0)_2 - 1/2)\exp(-2\nu t) + 1/2. \tag{6.5}$$

6.3 Main Results

6.3.1 Single Integrators

Consider now $p \geq 1$ single integrators stacked together. We consider each single integrator in normalized form (6.1), with their own $\rho_i > 0$, $i = 1, \ldots, p$, and drop the hats for convenience. Since the states are equivalent to the outputs, we may refer to these interchangeably.

Now we formulate our problem of a hybrid limit cycle. We construct a set $S \subset \mathbb{R}^p$ and a map $F : S \to \mathbb{R}^p$ that acts essentially as a Poincaré return map, in the following sense:

(1) S is defined as the product of the invariant sets of *Forward* in each output, and F may begin at any point $x \in S$;

(2) in each output $i = 1, \ldots, p$, F first applies the *Forward* closed-loop vector field to x_i until the guard set is reached (at position value 1), which defines the crossing time T_i through the equation $1 = \rho_i T_i + x_i$; let T be the largest of the T_i;

(3) in each output, once the guard set is reached, the state is reset (to position value 0) and then the *Hold* closed-loop vector field (from position 0) is applied for a time duration of $T - T_i$ (seconds), waiting until the "slowest" output completes *Forward*; the new point $F(x)$ occurs at time T (there is no need to reset again since we are still in Y^*).

Following the above recipe, we formally define the set S as $S = \prod_{i=1}^{p} I^0(\mathscr{F}) = [0,1]^p \subset \mathbb{R}^p$ and we define the map $F : S \to \mathbb{R}^p$ component-wise for $i = 1, \ldots, p$ as

$$F_i(x) = (0 - 1/2)\exp(-2\rho_i(T - T_i)) + 1/2 \tag{6.6}$$

$$= 1/2\left[1 - \exp(-2\rho_i(T - T_i))\right], \tag{6.7}$$

where $T_i = (1 - x_i)/\rho_i \geq 0$, and $T = \max\{T_i \mid i = 1, \ldots, p\}$. To show that the return map concept works, we will show that $F(S) \subset S$.

The main result is the following.

Theorem 6.3.1. *The map F defined by* (6.6) *has a unique fixed point x^*. Moreover, for all $x \in S$ the iterate $F^n(x) = F \circ \cdots \circ F(x)$ converges to x^* as $n \to \infty$ (in the standard topology of $\mathbb{R}^p$).*

Modulo the "resetting action", it should be clear that the associated hybrid limit cycle is generated by any state trajectory that goes through the fixed point. Moreover, the hybrid limit cycle is attractive. See Figure 6.2 for a generic example.

The contraction principle seems like it would be the ideal tool to use in proving this result. However, some preliminary attempts have indicated that this approach is complicated by the hybrid nature of F, which applies the two different vector fields of *Forward* and *Hold* with different timings in each output. Our strategy is to first show that for each point $x \in S$, applying F iteratively a finite number of times will reach a certain invariant subset of S, $\hat{S}$, which contains the fixed point. Then applying the contraction principle to the restriction of F to $\hat{S}$ gives the result.

We characterize the invariant subset by considering the outputs associated with the smallest or "slowest" parameters ρ_i. Let $\rho = \min\{\rho_i \mid i = 1, \ldots, p\}$, $I_{\rho=} = \{i = 1, \ldots, p \mid \rho_i = \rho\}$, and $I_{\rho<} = \{i = 1, \ldots, p \mid \rho < \rho_i\}$. We define the subset $\hat{S}$ as those states in S which have at least one slowest output starting at zero:

$$\hat{S} = \{x \in S \mid (\exists i \in I_{\rho=})\ x_i = 0\}. \tag{6.8}$$

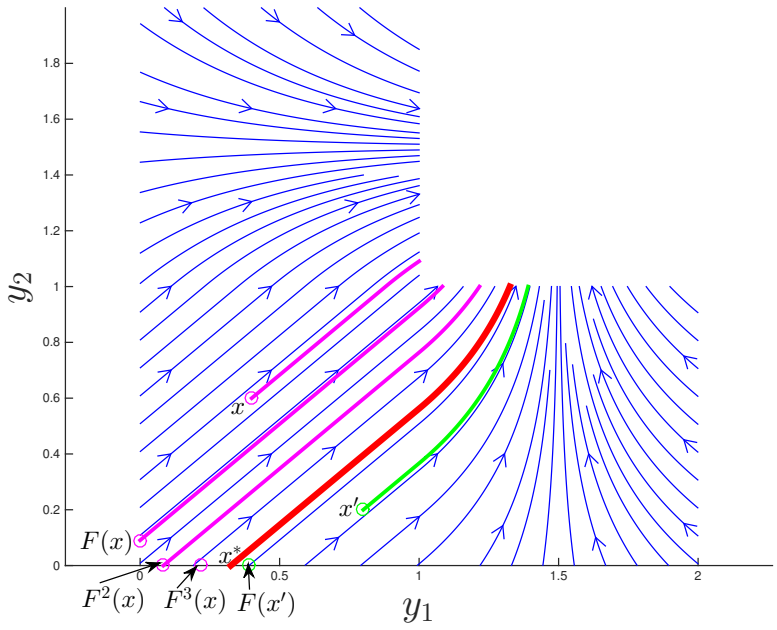

Figure 6.2: The limit cycle (red) going through x^*, and two hybrid trajectories (magenta and green) going through x and x' respectively. For ease of illustration, the resetting (subtracting 1 from each component) occurs only at the maximum crossing time T rather than after the individual crossing times T_i. The points mapped by F are shown, and they converge to the point x^* on the limit cycle. The point x^* is non-trivial, since $\rho_1 = 1.2$ and $\rho_2 = 1$.

We define $\hat{F} : \hat{S} \to \mathbb{R}^p$ as the restriction of F to $\hat{S}$. First we establish some basic properties.

Lemma 6.3.2. *We have that*

(i) $F(S) \subset S$,

(ii) $\hat{F}(\hat{S}) \subset \hat{S}$,

(iii) and for all $x \in \hat{S}$, *the maximum crossing time is* $T = 1/\rho$.

Proof. First we prove (i). Since for all $i = 1, \ldots, p$, $\rho_i > 0$ and $T - T_i \geq 0$, we must have that $0 < \exp(-2\rho_i(T - T_i)) \leq 1$, and thus $0 \leq F_i(x) < 1/2 < 1$.

Next we prove (ii) and (iii). Let $x \in \hat{S}$ and write $x_i = 0$ for some $i \in I_{\rho=}$. To show that $F(x) \in \hat{S}$, we will show that $F_i(x) = 0$; $T = 1/\rho$ will be deduced in the process. Since $\hat{S} \subset S$, for all $j = 1, \ldots, p$, $0 \leq x_j \leq 1$, and thus $0 \leq 1 - x_j \leq 1$. By definition of ρ, $\rho_j \geq \rho_i$ for all $j = 1, \ldots, p$. Altogether, we get for all $j = 1, \ldots, p$ that

$$(1 - x_i)\rho_j = \rho_j \geq \rho_i \geq \rho_i(1 - x_j) \quad \Rightarrow \quad T_i \geq T_j.$$

This shows that $T = T_i = 1/\rho$, which proves (iii). Then the formula for $F_i(x)$ gives that $F_i(x) = 0$, which proves (ii). □

The next results establish convergence to $\hat{S}$ in a finite number of iterations and that $\hat{F}$ has a unique fixed point. Referring to Figure 6.2 for an illustration, we see that $\hat{S} = \{x \in [0,1]^2 \mid x_2 = 0\}$ since $\rho_2 < \rho_1$. For the sample points $x, x' \in S$, they reach $\hat{S}$ in a finite number of iterations, since $F^2(x) \in \hat{S}$ and $F(x') \in \hat{S}$.

Lemma 6.3.3. *For all $x \in S$, there exists $N \geq 0$ such that the iterate $F^N(x) := F \circ \cdots \circ F(x) \in \hat{S}$.*

Proof. Let $x \in S$. First, observe that for all $n \geq 0$, the iterations $F^n(x)$ are well-defined, since by Lemma 6.3.2 (i), $F^n(x) \in S$.

If $x \in \hat{S} \subset S$, the result is trivial with $N = 0$ ($F^0(x) := x$). Thus suppose that $x \in S \setminus \hat{S} \neq \emptyset$ and write the iterates as $F^n(x)$ for $n \geq 0$. For each $n \geq 1$, let T_i^n be the individual output crossing times for $i = 1, \ldots, p$, and let $T^n = T_{i(n)}^n$ be the maximal crossing times (occurring at some output $i(n) \in \{1, \ldots, p\}$). It suffices to show that there exists some smallest $N \geq 1$ and $j \in I_{\rho=}$ such that $T^N = T_j^N$, assuming that $i(n) \notin I_{\rho=}$ for all $n < N$. Our strategy is the following. For each output $i \in I_{\rho=}$, we independently bound from above its iterated values by a suitably chosen function g_i. This function is essentially the worst-case mapping F_i assuming the maximal crossing time always occurs at some output i' with second slowest speed or parameter $\rho_{i'}$ from position 0. Moreover, this function has the property that it decreases to negative infinity. Then we choose N and j such that the iterate $g_j^N(x_j)$ first becomes negative; this will allow us to conclude that the output j gives the maximal crossing time.

Since $S \setminus \hat{S} \neq \emptyset$ implies that $I_{\rho<} \neq \emptyset$, let $\rho' = \min\{\rho_i \mid i \in I_{\rho<}\}$ denote the second smallest parameter. For each $i \in I_{\rho=}$, define the scalar mapping $g_i : \mathbb{R} \to \mathbb{R}$ as

$$g_i(z) = 1/2\,[1 - \exp(-2\rho(1/\rho' - t_i)]\,,$$

where $t_i = (1 - z)/\rho$. Since $\rho < \rho'$, it can be observed that for any fixed initial $z \in [0,1]$ that the iterate $g_i^n(z)$ tends to $-\infty$. Starting from x_i, write $g_i^n(x_i)$ as the iterates for $n \geq 0$, with crossing times t_i^n for $n \geq 1$. Since $g_i^n(x_i)$ tends to $-\infty$, let $N_i \geq 1$ be the earliest iteration where $g_i^{N_i}(x_i) < 0$, or equivalently $1/\rho' - t_i^{N_i} < 0$. Let $N = \min\{N_i \mid i \in I_{\rho=}\}$ and $j = \operatorname{argmin}\{N_i \mid i \in I_{\rho=}\}$. Thus $1/\rho' < t_j^N$.

Next, for all $0 \leq n < N$ we claim that $F_j^n(x) \leq g_j^n(x_j)$. This result is established by induction. The result is clearly true when $n = 0$. Supposing it is true for $0 \leq n - 1 < N - 1$ we show it is true for n. By direct manipulation, $F_j^n(x) \leq g_j^n(x_j)$ is equivalent to $T_{i(n)}^n - T_j^n \leq 1/\rho' - t_j^n$. First, from $F_j^{n-1}(x) \leq g_j^{n-1}(x_j)$, we establish that $-T_j^n \leq -t_j^n$ (a). Second, since $F(S) \subset S$ by Lemma 6.3.2 (i) and for all $n' < N$, $i(n') \notin I_{\rho=}$, $\rho_{i(n-1)} \geq \rho'$ implies $1/\rho' \geq T_{i(n)}^n = (1 - F_{i(n-1)}^{n-1}(x))/\rho_{i(n-1)}$ (b). Subtracting t_j^n from both sides of (b) and using (a) gives the desired result.

Since $F_j^{N-1}(x) \leq g_j^{N-1}(x_j)$ by the above result, direct manipulation gives $t_j^N = (1 - g_j^{N-1}(x_j))/\rho \leq (1 - F_j^{N-1}(x))/\rho = T_j^N$. Since $T_i^N \leq 1/\rho'$ for all $i \in I_{\rho<}$ and recalling that, $1/\rho' < t_j^N$, we have that $T^N = T_j^N$ is the maximum crossing time, completing the proof.

□

Lemma 6.3.4. *The map $\hat{F}$ has a unique fixed point $x^* \in \hat{S}$ and for all $x \in \hat{S}$, the iterate $\hat{F}^n(x)$ converges to x^* as $n \to \infty$ (in the standard topology of $\mathbb{R}^p$).*

Proof. We invoke the contraction principle. We equip $\hat{S}$ with the standard topology of $\mathbb{R}^p$, and use the infinity-norm metric, $d(x, x') = \max\{|x_i - x_i'| \mid i = 1, \ldots, p\}$, for convenience. First observe that $\hat{S}$ is compact (since $\hat{S} \subset \mathbb{R}^p$ is both closed and bounded) and $\hat{F}(\hat{S}) \subset \hat{S}$ by Lemma 6.3.2 (ii). It remains to show that $\hat{F}$ is a weak contraction.

Let $x, x' \in \hat{S}$, $x \neq x'$. We must show that $d(\hat{F}(x), \hat{F}(x')) < d(x, x')$. For each $i = 1, \ldots, p$, write the crossing times as T_i and T_i' for x and x' respectively. Moreover, $x, x' \in \hat{S}$ and Lemma 6.3.2(iii) implies that the maximum crossing times are $T = T' = 1/\rho$. Consider a fixed $i \in \{1, \ldots, p\}$ such that $x_i \neq x_i'$. Define $\beta_i = \exp(-2\rho_i/\rho + 1)$, $z_i = \min\{1 - 2x_i, 1 - 2x_i'\}$, and $z_i' = \max\{1 - 2x_i, 1 - 2x_i'\}$. Since $\rho \leq \rho_i$, it is easy to show that $\beta_i \leq 1/e$. Then we invoke the Mean Value Theorem for the function exp on the domain $[z_i, z_i']$ so that there exists $c \in (z_i, z_i')$ such that $\exp(z_i') - \exp(z_i) = \exp(c)(z_i' - z_i)$. Taking absolute values gives $|\exp(z_i') - \exp(z_i)| = |\exp(c)||z_i' - z_i|$. Since $0 \leq x_i, x_i' \leq 1$, we have that $-1 \leq z_i, z_i' \leq 1$, and so $|\exp(c)| < |\exp(1)| = e$. Combining, we have $|\exp(z_i') - \exp(z_i)| < e|z_i' - z_i|$. Using this (Lipschitz) condition with $\beta_i \leq 1/e$, direct computation then gives

$$\begin{aligned} |\hat{F}_i(x) - \hat{F}_i(x')| &= \left| \frac{1}{2}\left[1 - \exp\left(-2\rho_i\left(\frac{1}{\rho} - \frac{1 - x_i}{\rho_i}\right)\right)\right] - \frac{1}{2}\left[1 - \exp\left(-2\rho_i\left(\frac{1}{\rho} - \frac{1 - x_i'}{\rho_i}\right)\right)\right] \right| \\ &= \frac{\beta_i}{2}|\exp(z_i') - \exp(z_i)| < \frac{1}{2e}e|z_i' - z_i| = |x_i - x_i'| \leq d(x, x'). \end{aligned}$$

Finally, since $|\hat{F}_i(x) - \hat{F}_i(x')| < d(x, x')$ is true for all $i \in \{1, \ldots, p\}$ such that $x_i \neq x_i'$, we have $d(\hat{F}(x), \hat{F}(x')) < d(x, x')$. □

Now we can easily prove the main result.

Proof of Theorem 6.3.1. Let $x \in S$ and consider the iterations $F^n(x)$, $n \geq 0$. These iterations are well-defined, since by Lemma 6.3.2 (i), $F^n(x) \in S$ for $n \geq 0$. By Lemma 6.3.3, there exists $N \geq 0$ such that $F^N(x) \in \hat{S}$. The iterations for $n \geq N$ then converge to a unique fixed point $x^* \in \hat{S}$ by Lemma 6.3.4. The uniqueness of $x^* \in S$ is established by observing that another point $x' \neq x \in S$ also reaches $\hat{S}$ in a finite number of iterations. □

Remark 6.3.5. *Finally we make the connection to the synchronization property shown in Figure 6.1, which follows immediately as a special case of Theorem 6.3.1. Suppose that all the parameters are the same for all the outputs, that is, $I_{\rho<} = \emptyset$. Then after one iteration, any point $x \in S$ reaches $\hat{S}$. The fixed point is $x^* = 0$, and so the individual crossing times T_i all tend to $1/\rho$ over the iterations. If the outputs represent the motion of vehicles in the same physical direction, this means that eventually all the vehicles move with the same constant speed and cross into the next box at the same time instant, even if they start with at different relative positions in their box with different speeds.*

6.3.2 Double Integrators

Consider now $p \geq 1$ double integrators stacked together. We consider each double integrator in normalized form, using (6.2), (6.3), (6.4), and (6.5) with their own $\nu_i > 0$, $i = 1, \ldots, p$, and drop the hats for convenience. There are $2p$ states, and we identify the outputs as the states with an odd index.

We apply the same concept of a Poincaré map. Let $S = \prod_{i=1}^{p} I^0(\mathscr{F}) \subset \mathbb{R}^{2p}$, where in normalized coordinates the *Forward* invariant region $I^0(\mathscr{F})$ is the convex hull of the six vertices $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$, and $(1,-1)$ in $\mathbb{R}^2$. Define the map $F : S \to \mathbb{R}^{2p}$ component-wise for $i = 1, \ldots, p$ as

$$F_{2i-1}(x) = \exp(-\nu_i(T - T_i)) \left[c_{i,1} \cos(\nu_i(T - T_i)) + c_{i,2} \sin(\nu_i(T - T_i))\right] + 1/2, \tag{6.9}$$

$$F_{2i}(x) = \exp(-\nu_i(T - T_i)) \left[(-c_{i,1} + c_{i,2}) \cos(\nu_i(T - T_i)) + (-c_{i,1} - c_{i,2}) \sin(\nu_i(T - T_i))\right], \tag{6.10}$$

where T_i is the unique non-negative solution to

$$1 = 1/2 \left[(1/2 - x_{2i})(\exp(-2\nu_i T_i) - 1) + \nu_i T_i\right] + x_{2i-1}, \tag{6.11}$$

$T = \max\{T_i \mid i = 1, \ldots, p\}$, $c_{i,1} = 0 - 1/2 = -1/2$, and

$$c_{i,2} = 0 + \left[(x_{2i} - 1/2) \exp(-2\nu_i T_i) + 1/2\right] - 1/2 = (x_{2i} - 1/2) \exp(-2\nu_i T_i). \tag{6.12}$$

Unfortunately, the proofs given in the single integrator case do not easily generalize to the double integrator case, because the times T_i cannot be solved for analytically due to the complexity of more states. However, a similar procedure of identifying an invariant subset $\hat{S}$, showing convergence to it from S, and then applying the contraction principle on $\hat{S}$, seems promising.

We sketch out some ideas. Again, consider the smallest parameter $\nu = \min\{\nu_i \mid i = 1, \ldots, p\}$ and the indices $I_{\nu=} = \{i = 1, \ldots, p \mid \nu_i = \nu\}$. Our main observation, which follows by manipulating the above

equations, is: for $x \in S$ and for all $i \in I_{\nu=}$, if $x_{2i-1} = 0$, $x_{2i} = 0.5$, and $T_i = T$, then $F_{2i-1}(x) = 0$ and $F_{2i} = 0.5$. This result isn't so useful because we have to assume that $T_i = T$, unlike Lemma 6.3.2(iii) which proved this. Also, in contrast to how $\hat{S}$ was defined in the single integrator case, the candidate set given below for $\hat{S}$ does not work:

$$S^c = \{x \in S \mid (\exists i \in I_{\nu=})\ x_{2i-1} = 0, x_{2i} = 0.5\}.$$

A simple counterexample is obtained from considering $\nu_1 = 1$, $\nu_2 = 1.1$, and $x = (0, 0.5, 0, 0) \in S^c$; one can verify that $F(x) \notin S^c$, since $T = T_2 > T_1$. Thus a suitable invariant set $\hat{S}$ is non-trivial to find, but simulation results support its existence.

We conclude by stating our main conjecture. Our numerical simulations support its validity.

Conjecture 6.3.6. *The map $F : S \to \mathbb{R}^{2p}$ defined above has a unique fixed point x^*. Moreover, for all $x \in S$ the iterate $F^n(x) = F \circ \cdots \circ F(x)$ converges to x^* as $n \to \infty$ (in the standard topology of $\mathbb{R}^p$).*

6.4 Conclusion

In this chapter we analyzed the periodic motions associated with the *Abstract Forward* motion primitive introduced in Chapter 5. For the case of single integrators for each output, we proved that there always exists a unique, attractive limit cycle. Moreover, when the parameters are selected accordingly, this results in a synchronization of the outputs, which is useful in the context of formation control. We conjectured that a similar result exists for the case of double integrators.

The analysis in this chapter was specific to the application that motivated its study. Experience with designing reach controllers has also suggested the existence of limit cycles, see for example Figure 1.1. This motivates a more general study of limit cycles in the context of hybrid systems with affine dynamics.

Chapter 7

Conclusion

This thesis investigated the application of formal methods in control to obtain a systematic and scalable methodology for motion planning in robotics. In the first research direction, we formulated an optimal control problem as an alternative to the RCP for obtaining feedback controllers more easily on higher dimensional systems. Our formulation led to the study of an unresolved problem in the area of indefinite linear quadratic optimal control, which we solved. In the second research direction, we formulated a novel modular framework for motion planning by combining feedback-based motion primitives with planning techniques in a compatible and rigorous way so as to ensure performance and safety. The methodology was demonstrated experimentally on quadrocopters and led to additional study into hierarchical structures and limit cycle analysis.

With regards to the two approaches considered, the literature on both optimal control and motion planning is vast. Throughout our investigations and contributions to these fields, a number of areas for future work were observed and elaborated in the conclusions of each chapter, which we now highlight.

- Through the classical or "geometric" approach to analytical optimal control, there are still many cases in which the solution to the optimization problem is not completely resolved. For example, in the indefinite linear quadratic control problem that we considered, necessary and sufficient conditions for a finite value function is still an open question. Although not considered in this thesis, singular optimal control problems, where the weighting on control need not be strictly positive definite, are much more difficult and contain many more unresolved cases [37]. Finally, we recognized the difficulty in applying controllers derived from optimization problems in the context of hybrid control, where often there are stringent requirements on reachability and safety. To this end, we highlighted some directions for optimal control problems with constraints in Section 3.8.

- The proposed framework for motion planning, both hierarchical and non-hierarchical versions, placed great emphasis on the fundamental architectural requirements that establish a formal guarantee on safety and implementability. The framework applies to nonlinear systems with symmetries, although we have placed emphasis on multi-agent systems. It would be worthwhile to investigate the application of our framework to other vehicle classes, particularly unicycles and robotic manipulators. Also it would be of interest to identify alternative methods of construction for motion primitives and planning algorithms for better automation of the controller synthesis process. In the context of the multi-level hierarchy we proposed, there is also potential for investigating different hierarchical structures to improve scalability and achieve very complex motions.

- The proposed framework for motion planning focused exclusively on reach-avoid problems in known static environments in a centralized manner. While an essential starting point for any motion planning framework, this restrictive scenario is not representative of many practical situations in the real world. Nevertheless, the modularity and robustness of our approach suggests applicability to dynamic environments or decentralized operation. In such modalities, we expect that the majority of our framework can be reused, with the difference of employing online planning algorithms to generate control policies at the highest level only.

- A theoretical issue in the design of motion primitives and maneuver automata was that of avoiding Zeno behavior. We highlighted some of the associated difficulties in numerous remarks throughout Chapters 4 and 5. From this point of view, our theory on maneuver automata is not complete. It is generally recognized in the hybrid systems literature that the study of Zeno behavior is very technically challenging.

- Our analysis of a limit cycle in Chapter 6 opened many research questions. First, there is the immediate resolution of our conjecture for the double integrator case. Second, and more importantly, is a more thorough analysis on the stability of limit cycles for hybrid systems with affine dynamics.

In the author's view, this is an exciting time for robotics and control. There are still many open questions, both on the theoretical and practical side. By engaging both sides through a feedback process of inquiry, better integration can ultimately be achieved.

Bibliography

[1] A. Akhtar, S.L. Waslander, and C. Nielsen. Path following for a quadrotor using dynamic extension and transverse feedback linearization. *Conference on Decision and Control*, 2012.

[2] A. Albert. Conditions for Positive and Nonnegative Definiteness in Terms of Pseudoinverses. *SIAM J. Applied Mathematics*. vol. 17, no. 2, pp. 434-440, 1969.

[3] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, vol. 88, no 7, pp. 971-984, 2000.

[4] B. d'Andrea-Novel, G. Bastin, and G. Campion. Modelling and control of non-holonomic wheeled mobile robots. *International Conference on Robotics and Automation*, 1991.

[5] A. D. Ames, A. Abate, and S. Sastry. Sufficient conditions for the existence of Zeno behavior. *IEEE Conference on Decision and Control*, 2005.

[6] A.D. Ames, X. Xu, J.W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp 3861-3876, 2016.

[7] B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods*. Prentice-Hall International, Inc., 1989.

[8] G. Ashford, *A time-varying feedback approach to reach control on a simplex*, MASc thesis, University of Toronto, 2011.

[9] G. Ashford and M. E. Broucke. Design of reach controllers on simplices. *IEEE Conference on Decision and Control*, 2013.

[10] F. Augugliaro, A. P. Schoellig, and R. D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. *International Conference on Intelligent Robots and Systems*, 2012.

[11] N. Ayanian and V. Kumar. Decentralized feedback controllers for multiagent teams in environments with obstacles. *IEEE Transactions on Robotics*. vol. 26, no. 5, pp. 878-887, 2010.

[12] A. Berman, M. Neumann, and R. J. Stern, "Nonnegative Matrices in Dynamic Systems," John Wiley and Sons, Inc., 1989.

[13] S. Bornot and J. Sifakis. On the composition of hybrid systems. *Hybrid Systems: Computation and Control*, pp. 49-63, 1998.

[14] S. Boyd and L. Vandenberghe, "Convex Optimization," Cambridge University Press, first edition, 2004.

[15] R. W. Brockett. *Finite Dimensional Linear Systems*. 1970.

[16] M.E. Broucke. Reach control on simplices by continuous state feedback. *SIAM J. Control and Optimization*. vol. 48, no. 5, pp. 3482-3500, 2010.

[17] M. E. Broucke, S. Di Gennaro, M. Di Benedetto, and A. Sangiovanni-Vincentelli. Efficient solution of optimal control problems using hybrid systems. *SIAM Journal on Control and Optimization*, vol. 43, no. 6, pp. 1923-1952, 2005.

[18] M.E. Broucke and M. Ganness. Reach control on simplices by piecewise affine feedback. *SIAM J. Control and Optimization*. vol. 52, no. 5, pp. 3261-3286, 2014.

[19] F. Bullo, J. Cortés, and S. Martínez, Distributed Control of Robotic Networks, Princeton University Press, 2009.

[20] P.E. Caines and Y-J. Wei. Hierarchical hybrid control systems: a lattice theoretic formulation. *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 501-508, 1998.

[21] M. Čáp, P. Novák, A. Kleiner, and M. Selecký. Prioritized planning algorithms for trajectory coordination of robots. *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 835-849, 2015.

[22] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta. Formal approach to the deployment of distributed robotic teams. *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 158-171, 2012.

[23] D. Chmielewski and V. Manousiouthakis, "On constrained infinite-time linear quadratic optimal control," *Systems and Control Letters*, vol. 29, no. 3, pp. 121-129, 1996.

[24] M. Cirillo, T. Uras, and S. Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. *International Conference on Intelligent Robots and Systems*, 2014.

[25] D. C. Conner, A. A. Rizzi, and H. Choset. Composition of local potential functions for global robot control and navigation. *International Conference on Intelligent Robots and Systems*, 2003.

[26] R. V. Cowlagi and P. Tsiotras. Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles. *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 379-395, 2012.

[27] M. Deniša and A. Ude. Synthesis of new dynamic movement primitives through search in a hierarchical database of example movements. *International Journal of Advanced Robotic Systems*, vol. 12, no. 10, pp. 137-150, 2015.

[28] A. Dobson, K. Solovey, R. Shome, D. Halperin, and K. E. Bekris. Scalable asymptotically-optimal multi-robot motion planning. *International Symposium on Multi-Robot and Multi-Agent Systems*, 2017.

[29] M. Egerstedt M. X. Hu. Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 947-951, 2001.

[30] J. Engwerda. The Regular Convex Cooperative Linear Quadratic Control Problem. *Automatica*. vol. 44, no. 9, pp. 2453-2457, 2008.

[31] Fainekos G. E., Girard A., and G. J. Pappas. Hierarchical synthesis of hybrid controllers from temporal logic specifications. *International Workshop on Hybrid Systems: Computation and Control*, Springer, pp. 203-216, 2007.

[32] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*. vol. 45, no. 2, pp. 343-352, 2009.

[33] E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems With symmetries. *IEEE Transactions on Robotics*. vol. 21, no. 6, pp. 1077-1091, 2005.

[34] F. R. Gantmacher. *The Theory of Matrices*. vol. 1. Chelsea Publishing, 1959.

[35] C.R. Garrett, T. Lozano-Pérez, and L.P. Kaelbling. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research*, 2018

[36] T. Geerts. A Necessary and Sufficient Condition for Solvability of the Linear-Quadratic Control Problem without Stability. *Systems and Control Letters*. vol. 11, no. 1, pp. 47-51, 1988.

[37] T. Geerts. *Structure of Linear-Quadratic Control*. Ph. D. Thesis, Eindhoven University of Technology, Eindhoven 1989.

[38] T. Geerts. A Priori Results in Linear-Quadratic Optimal Control Theory. *Kybernetika*. vol. 27, no. 5, pp. 446-457, 1991.

[39] J. Gillula, G. Hoffmann, H. Huang, M. Vitus, and C. Tomlin, "Applications of hybrid reachability analysis to robotic aerial vehicles," *The International Journal of Robotics Research*, vol. 30, no. 3, pp. 335-354, 2011.

[40] R. Goebel and M. Subbotin, "Continuous time constrained linear quadratic regulator - convex duality approach," in *Proceedings of the American Control Conference*, 2005.

[41] I. Gohberg, P. Lancaster, and L. Rodman. On Hermitian Solutions of the Symmetric Algebraic Riccati Equation. *SIAM J. Control and Optimization*. vol. 24, no. 6, pp. 1323-1334, 1986.

[42] E.A. Gol, M. Lazar, and C. Belta. Language-Guided Controller Synthesis for Linear Systems. *IEEE Transactions on Automatic Control*. vol. 59, no. 5, pp. 1163-1176, 2014.

[43] D.J. Grymin, C.B. Neas, and M. Farhood. A hierarchical approach for primitive-based motion planning and control of autonomous vehicles. *Robotics and Autonomous Systems*, vol. 62, no. 2, pp. 214-228, 2014.

[44] M. Guo, K. H. Johansson, and D. V. Dimarogonas. Motion and action planning under LTL specifications using navigation functions and action description language. *International Conference on Intelligent Robots and Systems*, 2013.

[45] M. Guo and D. V. Dimarogonas. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218-235, 2015.

[46] L.C.G.J.M. Habets and J.H. van Schuppen, "A control problem for affine dynamical systems on a full-dimensional polytope," *Automatica*, vol. 40, no. 1, pp. 21-35, 2004.

[47] L.C.G.J.M. Habets, P.J. Collins, and J.H. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*. vol. 51, no. 6, pp. 938-948, 2006.

[48] K.A. Hamed, B.G. Buss, and J.W. Grizzle, "Exponentially stabilizing continuous-time controllers for periodic orbits of hybrid systems: Application to bipedeal locomotion with ground height variations", *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 977-999, 2016.

[49] D. Harabor and A. Botea. Hierarchical path planning for multi-size agents in heterogeneous environments. *IEEE Symposium on Computational Intelligence and Games*, 2008.

[50] W.P.M.H. Heemels, S.J.L. Van Eijndhoven, and A.A. Stoorvogel, "Linear quadratic regulator with positive controls," *International Journal of Control*, vol. 70, no. 4, pp. 551-578, 1998.

[51] M.K. Helwa, *Reach Control Problem on Polytopes*, PhD thesis, University of Toronto, 2013.

[52] D. Heß, M. Althoff, and T. Sattel. Formal verification of maneuver automata for parameterized motion primitives. *International Conference on Intelligent Robots and Systems*. 2014.

[53] P. Hubbard and P.E. Caines. Dynamical consistency in hierarchical supervisory control. *IEEE Transactions on Automatic Control*, vol. 47, no. 1, pp. 37-52, 2002.

[54] K. H. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. On the regularization of Zeno hybrid automata. *Systems and Control Letters*. vol. 38, no. 3, pp. 141-150, 1999.

[55] L.P. Kaelbling and T. Lozano-Pérez. Hierarchical task and motion planning in the now. *IEEE International Conference on Robotics and Automation*, 2011.

[56] C-T. Kim and J-J. Lee. Mobile robot navigation using multi-resolution electrostatic potential field. *31st Annual Conference of IEEE Industrial Electronics Society*, 2005.

[57] M. Kloetzer and C. Belta. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 320-330, 2007.

[58] M. Kloetzer and C. Belta. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Transactions on Automatic Control*. vol. 53, no. 1, pp. 287-297, 2008.

[59] M. Kloetzer and C. Belta, Dealing with non-determinism in symbolic control. *International Workshop on Hybrid Systems: Computation and Control* 2008.

[60] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360-375, 2012.

[61] Z. Kroeze, *Output Reach Control Problem with Applications to Motion Planning for Robotic Systems*, PhD thesis, University of Toronto, 2018.

[62] V. Kucera. A Contribution to Matrix Quadratic Equations. *IEEE Trans. Automatic Control*. vol. 17, no. 3, pp. 344-347, 1972.

[63] D. Kulic, C. Ott, D. Lee, J. Ishikawa, and Y. Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 330-345, 2012.

[64] P. Lancaster and L. Rodman. *Algebraic Riccati Equations*. Clarendon Press, Oxford, 1995.

[65] S. M. LaValle. *Planning Algorithms*. Cambridge University Press. 2006.

[66] T. Lee, M. Leok, and N.H. McClamroch. Geometric tracking control of a quadrotor UAV. *Conference on Decision and Control*, 2010.

[67] S.R. Lindemann and S.M. LaValle. Incremental low-discrepancy lattice methods for motion planning. *IEEE International Conference on Robotics and Automation*, 2003.

[68] S. R. Lindemann and S. M. LaValle. Simple and efficient algorithms for computing smooth, collision-free feedback laws over given cell decompositions. *International Journal of Robotics Research*, vol. 28, no. 5, pp. 600-621, 2009.

[69] S. Lupashin, M. Hehn, M. Mueller, A. P. Schoellig, M. Sherback, and R. D'Andrea, A platform for aerial robotics research and demonstration: the Flying Machine Area," *Mechatronics*, vol. 24, no. 1, pp. 41-54, 2014.

[70] J. Lygeros, K.H. Johansson, S. Sastry, and M. Egerstedt. On the existence of executions of hybrid automata. *IEEE Conference on Decision and Control*, 1999.

[71] A. Majumdar and R. Tedrake. Funnel libraries for real-time robust feedback motion planning. *International Journal of Robotics Research*, vol. 36, no. 8, pp. 947-982, 2017.

[72] D. Li, E. Mayar, and J. Raisch. A novel hierarchical control architecture for a class of discrete-event systems. *IFAC Proceedings Volumes*, vol. 37, no. 18, pp. 405-410, 2004.

[73] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. *International Conference on Robotics and Automation*, 2011.

[74] M. D. Mesarovic, D. Macko, and Y. Takahara. Theory of hierarchical, multilevel, systems. *Elsevier Science*, vol. 68, 1st edition, 2000.

[75] M. Mesbahi and M. Egerstedt. Graph theoretic methods in multiagent networks. Princeton University Press, 2010.

[76] P. Mihail, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, vol. 26, no. 3, pp. 308-333, 2009.

[77] M. Moarref, M. Ornik, and M.E. Broucke, "An obstruction to solvability of the reach control problem using affine feedback," *Automatica*, vol. 71, pp. 229-236, 2016.

[78] B. P. Molinari. The Time-Invariant Linear-Quadratic Optimal Control Problem. *Automatica*. vol. 13, no. 4, pp. 347-357, 1977.

[79] M. Montemerlo and S. Thrun. A multi-resolution pyramid for outdoor robot terrain perception. AAAI, 2004.

[80] S. Nagavalli, N. Chakraborty, and K. Sycara. Automated sequencing of swarm behaviors for supervisory control of robotic swarms. *IEEE International Conference on Robotics and Automation*, 2017.

[81] K-K. Oh, M-C. Park, and H-S. Ahn. A survey of multi-agent formation control. *Automatica*, vol 53, no. 3, pp. 424-440, 2015.

[82] M. Ornik, *New Mathematical Tools in Reach Control Theory*, PhD thesis, University of Toronto, 2017.

[83] M. Pachter. Revisit of Linear-Quadratic Optimal Control. *J. Optimization Theory and Applications*. vol. 140, no. 2, pp. 301-314, 2009.

[84] M. Pivtoraiko and A. Kelly. Kinodynamic motion planning with state lattice motion primitives. *IEEE International Conference on Intelligent Robots and Systems*, 2011.

[85] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme. Downwash-aware trajectory planning for large quadrotor teams. *International Conference on Intelligent Robots and Systems*, 2017.

[86] C. C. Pugh. Real Mathematical Analysis". Second edition, Springer, 2015.

[87] V. Raman and H. Kress-Gazit. Synthesis for multi-robot controllers with interleaved motion. *International Conference on Robotics and Automation*, 2014.

[88] J. B. Rawlings and D. Q. Mayne. Model Predictive Control: Theory and Design. Nob Hill Publishing, first edition, 2009.

[89] G. Rigatos. Nonlinear control and filtering using differential flatness approaches. Springer International Publishing, 2015.

[90] B. Roszak and M. E. Broucke. Necessary and sufficient conditions for reachability on a simplex. *Automatica*. vol. 42, no. 11, pp. 1913-1918, 2006.

[91] A. Roza and M. Maggiore. Path following controller for a quadrotor helicopter. *American Control Conference*, 2012.

[92] A. Roza and M. Maggiore. A class of position controllers for underactuated VTOL vehicles. *IEEE Transactions on Automatic Control*, vol. 59, no. 9, pp. 2580-2585, 2014.

[93] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas, and S. A. Seshia. Automated composition of motion primitives for multi-robot systems from safe LTL specifications. *International Conference on Intelligent Robots and Systems*, 2014.

[94] C. Scherer. The Solution Set of the Algebraic Riccati Equation and the Algebraic Riccati Inequality. *Linear Algebra and its Applications*. vol. 153, pp. 99-122, 1991.

[95] J. M. Schumacher. The Role of the Dissipation Matrix in Singular Optimal Control. *Systems and Control Letters*. vol. 2, no. 5, pp. 262-266, 1983.

[96] P. O. M. Scokaert and J. B. Rawlings, "Constrained Linear Quadratic Regulation," *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1163-1169, 1998.

[97] L. Sentis and O. Khatib, Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, vol. 2, no. 4, pp. 505-518, 2005.

[98] M. A. Shayman. Geometry of the Algebraic Riccati Equation-Part 1. *Siam J. Control and Optimization*. vol. 21, no. 3, pp. 375-393, 1983.

[99] J.M. Soethoudt and H. L. Trentelman. The Regular Indefinite Linear-Quadratic Problem with Linear Endpoint Constraints. *Systems and Control Letters*. vol. 12, no. 1, pp. 23-31, 1989.

[100] M.W. Spong, S. Hutchinson, and M. Vidyasagar. Robot Modeling and Control. New York: Wiley, 2005.

[101] N. Sturtevant M. and Buro. Partial pathfinding using map abstraction and refinement. *20th national conference on Artificial intelligence*, 2005.

[102] X. Sun and C.G. Cassandras. Optimal dynamic formation control of multi-agent systems in constrained environments. *Automatica*, vol. 73, no. 11, pp. 169-179, 2016.

[103] P. Tabuada, G. J. Pappas, and P. Lima, Compositional abstractions of hybrid control systems. *IEEE Conference on Decision and Control*, 2001.

[104] A. Tiwari. Abstractions for hybrid systems. *Formal Methods in System Design*, vol. 32, no. 1, pp. 57-83, 2008.

[105] H. L. Trentelman. The Regular Free-Endpoint Linear Quadratic Problem with Indefinite Cost. *SIAM J. Control and Optimization*. vol. 27, no. 1, pp. 27-42, 1989.

[106] H. Trentelman, A. A. Stoorvogel, and M. Hautus. *Control Theory for Linear Systems*. Springer, London, 2001.

[107] P. Tsiotras, D. Jung, and E. Bakolas. Multiresolution hierarchical path-planning for small UAVs using wavelet decompositions. *Journal of Intelligent and Robotic Systems*, vol. 66, no. 4, pp. 505-522, 2012.

[108] M. Vukosavljev, I. Jansen, M. E. Broucke, and A. P. Schoellig. Safe and Robust Robot Maneuvers Based on Reach Control. *International Conference on Robotics and Automation*. 2016.

[109] M. Vukosavljev, Z. Kroeze, M. E. Broucke, and A. P. Schoellig. A framework for multi-vehicle navigation using feedback-based motion primitives. *International Conference on Intelligent Robots and Systems*, 2017.

[110] M. Vukosavljev, A. P. Schoellig, and M. E. Broucke. The regular indefinite linear quadratic optimal control problem: stabilizable case. *SIAM Journal on Control and Optimization*, vol. 56, no. 1, pp. 496-516, 2018.

[111] M. Vukosavljev, Z. Kroeze, A.P. Schoellig, and M.E. Broucke. A modular framework for motion planning using safe-by-design motion primitives. To appear in *IEEE Transactions on Robotics* in 2019.

[112] M. Vukosavljev, A.P. Schoellig, and M.E. Broucke. Hierarchical motion primitives for motion planning To be submitted to *International Journal for Robotics Research* in 2019.

[113] G. Wagner and H. Choset, Subdimensional expansion for multirobot path planning. *Artificial Intelligence* vol. 219, no. 2, pp. 1-24, 2015.

[114] J. C. Willems. Least Squares Stationary Optimal Control and the Algebraic Riccati Equation. *IEEE Trans. Automatic Control*. vol. 16, no. 6, pp. 621-634, 1971.

[115] J. C. Willems, A. Kitapçi, and L. M. Silverman. Singular Optimal Control: A Geometric Approach. *SIAM J. Control and Optimization*. vol. 24, no. 3, pp. 323-337, 1986.

[116] E. M. Wolff, U. Topcu, and R. Murray. Optimal control of non-deterministic systems for a computationally efficient fragment of temporal logic. *IEEE Conference on Decision and Control*. December 2013.

[117] W. M. Wonham. On a Matrix Riccati Equation of Stochastic Control. *SIAM J. Control*. vol. 6, no. 4, pp. 681-697, 1968.

[118] W. M. Wonham. *Linear Multivariable Control: A Geometric Approach*. Springer-Verlag, New York, 3rd Edition, 1985.

[119] W.M. Wonham and K. Cai. *Supervisory Control of Discrete-Event Systems*. `http://www.control.utoronto.ca/~wonham/Research.html`.

[120] J. Yu and D. Rus. An Effective Algorithmic Framework for Near Optimal Multi-Robot Path Planning. Robotics Research, Springer Berlin/Heidelberg, vol. 1, pp. 495-511, 2018.

[121] J. Zhang, K. H. Johansson, J. Lygeros, and S. Sastry, Dynamical systems revisited: hybrid systems with Zeno executions. *Hybrid Systems: Computation and Control*, 2000.

[122] D. Zhou and M. Schwager. Vector field following for quadratures using differential flatness. *International Conference on Robotics and Automation*, 2014.

[123] D.J. Zhu and J-C. Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 9-20, 1991.

[124] W.M. Zuberek and I. Bluemke. Hierarchies of place/transition refinements in petri nets. *IEEE Conference on Emerging Technologies and Factory Automation*, 1996.