Correct-by-Construction Control Adaptation against Sensing, Model, and Information Uncertainty

by

Kwesi Joe Rutledge

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan
2022

Doctoral Committee:

Associate Professor Necmiye Özay, Chair
Associate Professor Dmitry Berenson
Professor Jessy Grizzle
Professor Ilya Kolmanovsky
Professor Hadas Kress-Gazit, Cornell University

Kwesi Joe Rutledge
krutledg@umich.edu
ORCID iD: 0000-0001-8231-1184

DEDICATION

This dissertation is dedicated to my grandmothers, Lois Gray and Jacqueline Rutledge. They were both teachers whose curiosity and love of learning travelled all the way down the family tree to their grandchildren's house in New Jersey.

ACKNOWLEDGMENTS

There are a great number of people that helped me get to this point and I apologize if I accidentally omit any one's name. Despite any omissions, I appreciate every one of you.

First and foremost, I must acknowledge my advisor Necmiye Özay for her steadfast support, honesty, and excitement over the years as my advisor during this PhD. She is a phenomenal example of a great problem solver and human being. I attempted to channel these and many other qualities as I struggled to solve difficult control problems or better communicate the importance of our research. I will never forget her humility despite her amazing depth of knowledge and the intense work ethic that has earned her so many great results.

Secondly, I would like to thank my other committee members, Profs. Berenson, Grizzle, Kress-Gazit and Kolmanovsky. Each of the committee members offered advice on matters related to my research, professional development, self-care or much more during the process of finishing this dissertation. I appreciate their helpful observations and honesty at all of the points along the way.

My colleagues in the Ozay group also deserve many thanks. This research group is special because, although we can seemingly work on very different areas of control, we are still willing to head into 'unfamiliar' territory and learn about another group-member's field. The willingness to go into unfamiliar territory was demonstrated by the eagerness that many of the participants in our lab reading group and I hope to find something similar in the near future. Additionally, some of my colleagues that have graduated (including Yunus Sahin and Petter Nilsson) helped teach me many of the tools that the group specialized in. Their helpful board conversations, GitHub repositories and research papers were invaluable. I'm also thankful for the many thought-provoking discussions with Glen, Zhe, Zexiang, Andrew, Daphna, Philippe, Yuhang, and Xingze in the group. Although those discussions did not always result in a paper, they broadened my understanding of the field of control and robotics in a way that I really appreciated.

I also must acknowledge the friends that I've made outside of the research lab. Through my time as an executive board member of the Graduate Society of Black Engineers and Scientists (formerly SMES-G) and my tenure helping out on the Electrical and Computer Engineering Graduate Student Council, I've met many fun and inspiring people that helped enable great things at the University of Michigan. I won't be able to list all of the individuals here, but some people that I was honored to have met were Eva Mungai, Gabrielle Dotson, Yves Nazon, Nosakhare Edoimioya, Nathan Louis, Uriah Israel, Jonathan Michaux, Raghav Muralidharan, and Daniel Spatcher.

Special thanks to my family and oldest friends for offering support whenever they called. To Gabriel Aguilera, thanks for dragging me out to a concert every once in a while to make sure I remembered why I love them so much. To John Brewer, thank you for remembering all of my random PhD terms, dates and goals despite your busy executive job. To Mr. Scott and Mrs. K, thank you for always making time to talk/eat with me when I needed it. To my parents, thank you for teaching me the value in deliberate practice and how to exceed my own expectations. I hope that I have made you proud.

Last, but not least, I would like to acknowledge my partner Nailah Seale. She bore with me during many trying deadline seasons and also my attempts to juggle work-life and personal-life throughout all of the years of my PhD. Her sacrifices did not go unnoticed and I would not have been able to accomplish all of what I did without her understanding and help.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

LIST OF APPENDICES

LIST OF ACRONYMS

Adaptive Control Lyapunov Function (aCLF)

Adaptive Control Barrier Function (aCBF)

Computation Tree Logic (CTL)

Cyber Physical System (CPS)

Inter-Triggering Hybrid Automaton (ITHA)

Linear Temporal Logic (LTL)

Linear Temporal Logic with Knowledge Operator (KLTL)

Mixed Integer Quadratic Program (MIQP)

Model Reference Adaptive Controller (MRAC)

Piece-wise Affine (PWA)

Proportional Integral and Derivative (PID)

Self-Tuning Regulator (STR)

Set-Membership Identification (SMID)

Signal Temporal Logic (STL)

ABSTRACT

The theory of formal methods had a profound effect on computer science. By providing tools that can verify a program's correctness or generate software that is "correct-by-construction," Formal Methods made it much easier to design safer, bug-free code for computer systems. Safe, bug-free code is not only helpful in pure software systems, but also in systems that interact with and manipulate the world. Unfortunately, these so-called Cyber-Physical Systems (CPSs) have limitations or considerations that prevent many of the formal methods tools from being immediately applied to them. Of the many limitations that prevent Formal Methods from being applied to Cyber-Physical Systems, this dissertation discusses three types: sensing, modeling, and information limitations.

First, this dissertation develops a method for overcoming a practical sensing limitation called "missing data." Missing data refers to a phenomena where some data that is important to the CPSs safety is lost either in transmission or during sensing. This work provides a method for guaranteeing safety of a CPS experiencing missing data during a reachability task. Specifically, we define a linear program that is feasible if and only if an adaptive controller exists that is guaranteed to achieve the reachability task subject to missing data. The adaptive controller is composed of prefix-constrained linear controllers with a simple switching rule between them and can be optimized with respect to a convex objective function.

Second, this dissertation develops a method for overcoming the limitations in our ability to model CPSs. Often, a single model of the CPS is not known a priori but a family of potential models is known. To guarantee that our system is safe while accomplishing a reachability task and with only a family of models known a priori, this work provides a bilinear optimization-based method which can guarantee task satisfaction despite this initial model uncertainty. Each model can be incorporated into an optimization using a consistency set, which is similar to a reachable set but contains additional model information. The consistency set is used to determine which actions will discriminate (or reveal information about the true, linear mode of the system) a priori and thus analyzes the trade-offs of discriminating actions and other tasks (such as reachability of a target region or minimization of a cost). The controller design problem is defined as a bilinear optimization. Rather than constructing this bilinear optimization problem from scratch, a method is presented for translating Linear Temporal Logic with the epistemic knowledge operator (KLTL) formulae into bilinear optimization problems.

Thirdly, this dissertation develops a method for overcoming the limited ability of a single CPS to make safe actions when part of a large collection of connected CPSs. The proposed method decomposes the problem of guaranteeing the safety of a collection of CPSs into a series of smaller problems for each individual agent using the framework of the inter-triggering hybrid automaton (ITHA). Under this framework, the problem can be solved by finding self-safe actions (actions that an agent knows should be safe under its own dynamics) along with responsible actions (actions that will not cause problems for others that this agent is "responsible for").

CHAPTER 1

Introduction

Adaptation to new environments has been an extremely useful skill for humans, allowing us to limit the number of skills we need to master to survive. For example, a person living in the tropical nation of Trinidad and Tobago can limit the number of skills they learn about keeping warm on the tundra because they will likely never need that knowledge to survive. This person can adapt to the tropical climate and avoid overly concerning themselves with non-tropical issues.

While such adaptation would be similarly useful in robots and other cyber-physical systems, transferring the adaptation to them has been difficult. For example, a robot that is fixed in a certain environment (e.g., a restaurant) might need a policy to achieve its tasks safely, while the same robot in different environment (e.g., a home) might require a somewhat different policy. The policy the robot must use to take actions depends on the environment and what we can sense, predict, or previously know about it. The task of using information collected about the environment to select a policy that achieves a given task is the focus of the field of adaptive control.

The field of adaptive control initially addressed the design of adaptive controllers for aircraft autopilot systems [179]. The designers of autopilot systems realized that, while a certain PID controller would safely guide a pilot at certain altitudes and aircraft speeds, it would not be safe at other altitudes and aircraft speeds. The adaptive controller was designed to safely switch between multiple PID structures depending on the measurements of the current altitude and speed. This switching adaptive structure was developed in the 1950s, and many other forms of adaptive control were developed in subsequent years. Unfortunately, the many types of adaptive controllers that were used to satisfy goals and specifications such as stability were not of use in some practical situations or were difficult to certify.

A richer set of goals is discussed in the area of formal methods (originally created in computer science). Formal methods is used in computer science as a set of tools that can for verify the correctness of or correctly design software. The success of such tools has caught the attention of control theorists and The area of formal methods in control seeks to expand the set of specifications that controllers can achieve. Specifications such as reachability and invariance are of interest

in this field, and practitioners attempt to design controllers that satisfy these specifications. The controllers developed by formal methods practitioners have been applied to linear [72, 122], nonlinear [2], and hybrid [76] cyber-physical systems including adaptive cruise control [111, 2] and air conditioning systems for public buildings [72]. While the number of places where formal methods have been used is inspiring, there are many limitations as to where it can be applied. Formal methods, for example, are typically used in systems where the state is directly observed or where the states of the system are finite.

Assuming that the state of the system is directly observed (state feedback) or that the number of states in the system are finite (finite state space) are prohibitive for many interesting cyber-physical systems such as robot manipulators or large fleets of autonomous vehicles. In such situations, the hidden dimensions, or the large dimensions of the system state, make using classical formal methods problematic. When these properties of the state are present, we propose methods that allow for the synthesis of controllers that correctly satisfy the goal or specification using adaptive controllers.

The form of dynamical system that will be discussed in this dissertation is the discrete-time hybrid system:

Definition 1 (Hybrid System with Unknown Parameters)**.** *A discrete-time hybrid system with hidden mode is a system whose hybrid state at time t, $s_t = (x_t, q_t)$, evolves according to the function:*

$$s(t+1) = f_\theta(s(t), u(t), w(t)), \quad w(t) \in \mathcal{W}_{q(t)} \tag{1.1}$$

and its state is measured according to the function

$$y(t) = \omega_\theta(t, s(t), v(t)), \quad v(t) \in \mathcal{V}_{q(t)}. \tag{1.2}$$

The hybrid state s_t is composed of the continuous state $x(t)$ and the discrete state $q(t)$. The parameter θ is not directly observed but can impact the dynamics or the measurement function. The input to the system is $u(t)$. The process disturbance to the hybrid system's dynamics $w(t)$ belongs to a bounded set $\mathcal{W}_{q(t)}$ that is determined by the discrete set $q(t)$. The measurement function ω_θ is a function of the current time, the state and the measurement disturbance $v(t)$ which belongs to a bounded set $\mathcal{V}_{q(t)}$ that is determined by the discrete state $q(t)$.

The dynamics f_θ is assumed to be a Lipschitz continuous function for a fixed mode $\theta \in \Theta$. The measurement function ω_θ may not be smooth.

Dissertation Organization. During the rest of this chapter, two of the fields (Adaptive Control and Formal Methods) that this dissertation seeks to combine are discussed. In Chapter 3 of this dissertation, an adaptive controller is developed which can be used to tackle the problem of con-

trolling a dynamical system in the presence of data or measurement loss during transmission from the sensor to the controller (i.e. $\omega_\theta(t, s)$ becomes an empty set for specific values of θ). In Chapter 4, an adaptive controller is developed which automatically trades off between mode discrimination and robust control to achieve a finite time horizon task (i.e. the synthesis algorithm automatically determines which regimes $\Theta_i \subseteq \Theta$ the final controller will be robust to depending on user preferences and the important constraint of achieving the task). In Chapter 5, we present a method which synthesizes an adaptive controller of the form in Chapter 4 that correctly satisfies a KLTL formula. In Chapter 6, we present a method developed in [136] for designing an adaptive controller which uses intention estimation techniques to decide which robust control invariant set to use to guide its decision-making. In Chapter 7, the conditions under which a very permissive or very conservative adaptive controller can be developed in several cyber-physical systems (e.g. an autonomous vehicle driving on a highway) are developed in terms of a modelling formalism which simplifies the controller synthesis problem for large multi-agent systems. Finally, this dissertation concludes with Chapter 8, which contains a review of the results presented here and an examination of its future implications.

1.1 Existing Gaps

This dissertation was written to address several important limitations in the field of formal methods. These limitations are that the field does not yet have standard tools for addressing sensing, model and information uncertainty. These three uncertainties will be described and then the field's current approaches to them will be outlined briefly in this section. A deeper analysis will be found in Section 1.2.

- *Sensing Uncertainty*: In this dissertation, sensing uncertainty refers to the uncertainty in when (if at all) measurements from sensors will arrive. This phenomena of delayed or dropped sensor measurements in control systems has been studied in the area of wireless control networks [173, 54], but has not been fully explored by the formal methods community. This is crucial because many important control systems in today's world (e.g., storm water drainage systems, power generation networks) rely on sensor readings being sent wirelessly across large distances. It is well-understood that such communications can be delayed or interrupted due to things like weather and adversarial attacks on the network [104].

- *Model Uncertainty*: In this dissertation, model uncertainty refers to parametric uncertainty in the dynamics of a system (i.e. how the current state evolves into the next state). The "parametric uncertainty" in the dynamics indicate that there are some parameters (typically that cannot be sensed) but govern the dynamics of the system. This is often the case when

a robotic system interacts with objects and people in the real world. Objects in the world have different masses, materials, and shapes which can dramatically impact how their state evolves over time [117] (e.g. a rubber ball will bounce in a very differently way than a glass ball). Learning or adapting to knowledge about the dynamics is necessary to understand these parameters in real-time and then to make guarantees about correctness and safety. Making such guarantees has only just begun to be studied through work such as [133, 91].

- *Information Uncertainty*: In this dissertation, information uncertainty refers to the uncertainty which occurs when a single agent (or robot) must make a decision as part of a team, but only has understanding about its individual state and its neighboring teammates. The problem of designing controllers or decision-making software for a single agent with limited knowledge about its team is the decentralized control problem and it has been approached recently by the formal methods community [114, 105, 119]. The approaches to the problem have many different flavors, but are sometimes limited by the definition of safety and other tasks. Specifying safety now often requires careful contract definitions [57, 25, 58], but a more flexible notion of safety (e.g., [141]) may be better suited for these problems.

1.2 Literature Review

The following collection of prior work will be useful in putting this work's claims into context.

1.2.1 Real-World Limitations of CPSs

The first part of this literature review will discuss the many ways that a real CPS differs from the idealized CPS considered in traditional formal methods research. These differences are described as "uncertainties" which are not present in traditional formal methods problem statements but must exist in order for correct-by-construction control to work on real CPSs.

1.2.1.1 Sensing Uncertainty (or Intermittent Sensing)

In this dissertation, sensing uncertainty refers to the uncertainty in when (if at all) measurements from sensors will arrive.

Sensor information may be delayed when reaching a decision maker by things such as communication queues in networks. For this reason, the subject of handling delayed measurements has been studied in the context of networked control systems for a long time [177]. In that context, delays in the transmission of sensor data to controllers and controller information to actuators

can be expected when transmitting across the network. We will primarily focus on the delay in transmission of sensor data to a controller/estimator in this dissertation.

The delay itself can be modeled as the output of a stochastic process. Several works model it as a Markov Decision Process with known [143] or partially known [176] transition probabilities. Nonetheless, the probabilistic nature of these approaches is not directly applicable for several safety-critical applications, where hard bounds on the estimation or tracking errors are often necessary. For example, in motion planning problems with long time horizons, the transition probabilities lead to the overall plan having a low probability of success.

Sensor information may also be completely lost when being transmitted to a decision maker. Control and estimation problems for systems subject to such losses are sometimes said to be subject to missing data or intermittent measurements. These systems have been extensively considered in the context of networked control systems [175, 145, 19] and more recently for security problems involving denial of service attacks [4]. For missing and intermittent observations modeled by probability distributions, extensions of the Kalman filter have been proposed (e.g., [145, 78, 16]) to estimate the system state. Set-membership estimators have also been proposed to estimate the state of the system subject to such events [56, 164]. Set-membership estimators produce an estimate which is a set of potential values that the state can currently have given the previous data. Similarly, in this setting of probabilistic data loss models, stabilizing controllers or controllers minimizing a quadratic cost have been studied (e.g., [140, 175, 55]). The probabilistic nature of many of these approaches makes it somewhat hard to use for safety-critical applications, but the set-membership approaches are much easier to apply for safety-critical systems. The set-membership estimators produce hard bounds on the estimation or tracking errors which can be used to guarantee that a state is safe during control.

Another approach for modeling missing data patterns is to use a characterization of the set of all plausible missing data patterns. A simple characterization of this sort is the so-called (m, k) firmness [77, 49] that indicates that for any k consecutive measurements, at least m are available. A more general set description can be obtained using automata [79, 171] or finite languages [131, 67, 132] to represent the set of all feasible missing data patterns. In continuous time, missing data patterns are effectively intervals during which measurements are not taken [84]. Jungers et al. [79] study observability and controllability like properties for discrete-time linear systems subject to data loss, and characterize conditions on the system and the automata representing the missing data pattern for these properties to hold. Laine et al. [84] define controllers for continuous time systems with invariant set analysis and the Hamilton Jacobi partial differential equation which is also known to struggle with scalability (in terms of the size of the state space). In this dissertation, we use the language-based representation as in [131, 67, 132] and focus on control and estimator designs, which guarantee constraints on the states or errors are satisfied, for systems subject to missing

measurements and various types of noise. Our approach is based on linear programming and scales much better than similar discrete-time methods. Another related, yet complementary line of work is on designing measurement schedules (i.e., when to measure and when not to measure if there is a budget on the number of measurements) together with controls [39, 7], which differs from our setting in that we assume the missing measurements are chosen adversarially from the set of feasible patterns.

1.2.1.2 Model Uncertainty

In this dissertation, model uncertainty refers to parametric uncertainty in how the system's state will evolve over time (i.e., parametric uncertainty in the dynamical system model).

Parametric uncertainty is a useful concept in the real world when a CPS must interact with many different objects or people. For example, a robotic arm brought into the home to organize a toy room will need to understand how to manipulate objects like Hot Wheels toy cars, Lego bricks, and much more. One could attempt to address this by saving an extremely rich data set into the robot's memory containing every object that can possibly find. Unfortunately, this is an impractical solution due to the amount of data collection necessary and the amount of storage required to keep the data set up-to-date. Instead, a less data-intensive approach is to develop algorithms that can work for classes of objects (e.g. algorithms that work for all toy cars) by incorporate parametric uncertainty into them. By incorporating parametric toy uncertainty into its algorithms, an algorithm does not need to know the exact toy that it has picked up but can instead discover the relevant parameters (e.g., mass, moment of inertia, dimensions) and then achieve the task at hand.

As illustrated in the toy example, parametric uncertainty often arises in situations where a CPS interacts with an object without knowledge of its physical parameters [148, 37] like mass or moment of inertia. Such parameters have physically plausible bounds which makes it easy to verify whether or not an estimator is correctly identifying them. Model uncertainty can also arise in situations where a fault occurs in a CPS. If the possible faults are known a priori, then all faulty systems and the correct system can be modeled a priori [169, 35]. It is up to the CPS's controller will thus have model uncertainty whenever it is running and must continuously make decisions that prove or disprove faultless operation.

Parametric uncertainty may appear in linear, nonlinear, and hybrid systems [106]. In some cases, model uncertainty is represented by a finite set of potential models [121, 128] (i.e. the parameters are the dynamics themselves). In other cases, the model uncertainty is represented by a parameter vector which appears linearly in the dynamics [92, 150]. For an example, consider the

following dynamical system:

$$\dot{x} = f(x) - \Delta(x)^\top \theta + B(x)u$$

which appears in [92]. When this is the case, set membership [92] or control-barrier based estimators [150] can be used to identify plausible values of the uncertain parameter θ. Occasionally, the parameter vector does not appear linearly in the dynamics but in a nonlinear, but Lipschitz function [106] in a nonlinear or a sector-bounded function [17]. When the sector-bound is a set like a conic hull and it is the only nonlinear component in an otherwise linear system [17], there are robust control approaches that can handle them.

1.2.1.3 Information Uncertainty

In this dissertation, information uncertainty refers to uncertainty about the effect of a single agent's action will be when it is part of a team but only has partial information about the team. This uncertainty arises in the decision-making processes of single agents in large collections (e.g. a single autonomous car on a highway of other cars) as they attempt to reason about what a safe action will be when they do not know how other agents will react to their own action.

This uncertainty is the reason why centralized controllers (or single controllers that control every agent in a collection) [81] are capable of achieving tasks that decentralized controllers (or controllers that run separately on each individual agent in the collection) [11, 15, 30] can not [157]. Unfortunately, centralized controllers can be impossible to implement in the real world due to communication bandwidth issues [15], security/competition concerns [123], and much more. So, decentralized controllers are needed, but come with the limitation that each controller only understands local information.

The information available to each agent in the collection can be described with a so-called information structure [95, 174]. An information structure can be static (i.e., the information is shared/observed by each agent does not change over time) or dynamic (i.e., the information that is shared can be changed by other agent's actions, states, etc.). One commonly used example of a static information structure is the partially nested information structure [174, 6]). In this setting, there is a tree structure that explains that for each agent (represented as a node on the graph) the information available to it is a superset of all of the information available to its child nodes. Another common static information structure is the one-step delayed information structure [107, 125] where information from one agent is sent to another after one step.

The number of information structures for which strong guarantees can be made are limited. Partially nested information structures can be very useful for making guarantees as they allow for an intuitive decomposition of the optimal control problem. Without such information structures,

one can analyze agents individually and assume that all other agents act adversarially (a very conservative approach)[40]. Another approach to making guarantees without the aid of helpful information structures but with an expectation of collaboration is to use contract-based reasoning [41, 149, 101, 163, 34, 138, 57, 88]. Another alternative viewpoint is presented by responsibility-sensitive safety [141, 158, 68, 42], in the context of autonomous driving, where some hard safety constraints are replaced by a notion of not causing a crash and avoiding one whenever possible. This is particularly well-suited for scenarios where some of the agents are human-controlled, which can lead to unpredictable behaviors. From this viewpoint, we expect autonomous agents to act in a well-behaved fashion as long as others reciprocate, and we should not punish the autonomous agents for failures that are out of their control.

1.2.2 Adaptive Control

Adaptive controllers are informally defined as any "controller with adjustable parameters and a mechanism for adjusting the parameters" [179]. This definition is broad enough to encapsulate many forms of feedback controllers, but several of the popular types of adaptive controllers are gain scheduling, model reference adaptive controllers (MRACs), self-tuning regulators (STRs) and dual controllers.

A gain scheduling controller changes its parameters depending on the operating conditions of the process in a pre-programmed way [86]. The pre-programmed gain schedule can change gains depending on the current time [80], the current state and its proximity to a nonlinearity/region of interest [50], or something else. It is frequently used in nonlinear control, where known non-linearities are compensated by a pre-computed gain schedule. Unfortunately, designing gain scheduling controllers is not straight-forward. For example, a constraint on how quickly the state changes [86, 36] may be needed for such methods to work on a given nonlinear systems and it is unclear how to define such a limit a priori. Also, the designer must select the scheduling variables and the switching function, a task which may not be straight-forward depending on how nonlinear the system dynamics are.

Model reference adaptive controllers (MRACs) or model reference adaptive systems are controllers in which "the desired performance is expressed in terms of a reference model, which gives the desired response to a command signal" [179]. In such a controller, there is a reference model which is placed in the closed-loop system with the controller. With both of these components in the control loop, the error (or a loss function written in terms of the error) can be minimized using a gradient of the cost with respect to the controller's parameters. This is the so-called MIT rule [113, 98]. The MIT rule is used to adapt adaptive controller parameters in a way that minimizes the cost or loss and has been implemented in DC motors [13], simple robotic systems [73] and more.

Unfortunately, in these applications and others, unconstrained gradient descent is often unsafe or undesirable. In order to decrease the value of the loss *while* achieving other tasks, Lyapunov theory has replaced [116, 115, 93] or in some cases combined with the MIT rule [124] to achieve stability while properly adapting to the target model. Other notions of stability can also be used to improve the classic MIT rule approach to MRAC design, but all of these methods suffer from some sort of restriction to sufficiently smooth dynamics or to systems with perfect state feedback. Only in those situations can derivatives be consistent and thus they are the most well-studied.

Self-tuning regulators (STRs) are controllers that attempt to automate controller design and parameter estimation during controller operation. The controller uses online estimates of the parameters to run a controller design algorithm which is then immediately used during a control task. Notably, self-tuning regulators were originally intended for systems with unknown but constant parameters or slowly-varying parameters [178]. The parameter value is converged upon by the parameter estimator, so that the controller (which treats the estimated parameters as the true parameters, a treatment which is called the certainty equivalence principle) eventually has a model close enough to the true system's. The estimation algorithm for self-tuning regulators is usually of a simple form like the recursive least squares estimator [179, 62] but a plethora of estimation algorithms have been proposed including support-vector based [156] algorithms. The controller synthesis algorithms are typically those controller design algorithms which can be computed quickly, including the Kalman Filter gain design algorithm [69], or a pole placement techniques [9, 161, 168, 137]. Self-tuning regulators, which mean to estimate and design controllers in real-time, are frequently limited to using algorithms which are simple enough to be run in very fast control loops and can not incorporate too many safety or task specifications into their design.

Dual controllers are controllers that incorporate the dual effects of a controller's actions into the design. The dual effects of a controller are that "control inputs to an uncertain system must have a probing effect for active learning of system uncertainty and a directing effect for controlling the system dynamics" [48]. Dual control design methods attempt to find optimal controllers that solve the stochastic version of the adaptive control problem. The principle of optimality is thus often used to design the controller [100], but model predictive control based methods can also be used [151, 152] to find solutions to the problem. Unfortunately, it is frequently difficult to solve the control design problem using Bellman's equation because "closed-form solutions for the Bellman equation can be obtained only for a handful of situations" [100], so approximate solutions are found instead [71, 162]. The model-predictive control approaches also suffer from computation difficulties and frequently resort to heuristics such as sampling trees or restrictive control functions [162] to reduce the complexity of the optimization problem.

The adaptive controllers in this work are similar to gain scheduled controllers, self-tuning regulators and dual controllers. The aims of this dissertation are to design such controllers in a way

that correctly satisfies a given specification for different classes of discrete-time systems.

1.2.2.1 "Explore, Then Exploit"

One common method for quickly designing adaptive controllers is to implement a controller that has two discrete modes. There is an "Explore" mode of the controller, which typically implements a simple system identification algorithm and then there is an "Exploit" mode of the controller, which typically implements a robust controller. This description of the adaptive controller's modes comes from the popular explore-exploit paradigm discussed in many forms of psychological research [165, 99, 32] (also examined in several related fields). Often, the "Explore" mode is run until parametric uncertainty is reduced enough in the controller. Then, the "Exploit" mode is run which has a guarantee that the task can be achieved when uncertainty is reduced to an acceptable level [136]. In this section, we discuss some of the algorithms implemented in the "Explore" or "Exploit" modes.

One form of system identification with formal guarantees is active mode or model discrimination [46]. The theory of active model discrimination designs controllers that can identify the true model of a system from a class of models for the system, with predominant application being fault detection [139, 120]. This controller, also known as a discriminating controller, produces a different trajectory for every system in the class when it is used to close the control loop. Designing discriminating controllers for linear [35], nonlinear [144], or discrete-state [53] systems is an active area of current research unto itself. For continuous-state linear or nonlinear systems, the main design approach is based on (convex) optimization; and for discrete-state systems, the problem is related to automata learning.

Another form of system identification with formal guarantees is set-membership identification (SMID) [92]. An estimator using SMID produces not a single estimate of the unknown parameter in an uncertain system, but instead produces a set of parameters which explain the observed data. This type of system identification algorithm is closely related to interval estimation, where intervals in n-dimensional space are produced as estimates of unmeasured parameters [75, 108] or states [44]. One advantage of producing interval estimates is that computing the volume or "size" of the estimate is easier than when the set-valued estimate is an arbitrary set.

In addition, another form of system identification uses small modifications to controllers to better identify parameters [3, 155]. In some cases, the modifications are small deviations applied to a trajectory-tracking controller that reveals more information about the parameters [3]. In [155], white Gaussian noise is applied to the system by the controller in order to characterize the system.

1.2.3 Formal Methods

Formal methods originated in computer science where designers of computer algorithms and network communication protocols sought to avoid using "informal design techniques" and to instead define techniques that "facilitate design of correct protocols" [29]. The three primary topics within formal methods were defined as specification, verification, and implementation or synthesis. While these three topics occur in many other fields as well (e.g., control theory), "formal methods" focused on applications in computer programming. These three topics were recently incorporated into the new areas of program synthesis in computer science [61, 51] as well as control theory. Formal methods have been applied to controller specification [146], synthesis [110], and verification [5] in a variety of ways but we will focus on controller synthesis for the rest of this section.

A specification in formal methods is typically written in the form of temporal logic (e.g. LTL [14], CTL [38], STL [96]), but can equivalently be described as an automaton [146] or via safe sets. This dissertation will consider specifications written in terms of temporal logics without much loss of generality. The synthesis problem is the problem of designing a controller which guarantees that the trajectory of the closed-loop system satisfies some temporal logic property correctly often by "[merging] concepts from formal methods, including formal specification languages and discrete protocol synthesis, and those from controls, including optimization-based control and receding horizon implementations" [167]. Discrete protocols can be developed to solve control problems by decomposing a control system into a finite, discrete-state space system using state-space abstraction [66, 110, 12] and then developing a strategy over the now discrete system. Alternatively, optimization-based protocols can be developed to achieve a specification by defining controllers via optimal control or model-predictive control approaches [18].

Until recently, most control synthesis problems in the literature used the setting where the state of the system was completely observed by the controller and the control system's dynamics were completely known a priori. This allows for the controller to determine the correctness or satisfaction of a formula during operation, but is frequently not the case in most control problems. The areas of output feedback is a large segment of control theory and adaptive control, as was just stated, are also of great interest to control theorists and yet very little has been said of applying formal methods in these settings. Some attempts at designing adaptive controllers using formal methods have recently been published, where state space abstraction [133] was used to simplify the problem or invariance specifications were considered which limited the scope of the guarantees that could be made [94]. Applying temporal logic to either of the control areas of output feedback or adaptive control is difficult because there is typically partial information available in each setting and thus it is difficult to reason about a formula being satisfied or not by the controller given the current output trajectory. Instead, one must reason about all possible state trajectories related to an output trajectory which requires temporal logics that can define hyperproperties. There are very

few papers discussing hyperproperties for control systems [5]. To address this scarcity, this dissertation aims to demonstrate how epistemic logics (which can be used to represent hyperproperties) can be used to express tasks that are important in robotics and can be used in controller synthesis.

1.2.3.1 Bounded Error Control

As we are interested in enforcing constraints on the control or estimation error, our results are related to disturbance rejection, set-valued observers, or ℓ_∞ filtering [21, 103, 142, 28, 85], though these approaches cannot readily handle missing data. Of particular interest to our approach is an intuitive property for state estimators called equalized performance that ensures that the estimation error does not increase at each step (e.g., [27, 28]). Instead, we allow a bounded increase in the error during missing data events as long as the error recovers back to its original level at the end.

From a computational standpoint, our controller and estimator synthesis approach builds upon Q-parametrization to reduce the non-convex measurement feedback controller/estimator design problem to a convex optimization form via a nonlinear change of variables [147]. In particular, we restrict the controllers and design variables in the estimators to be affine in the measurements with memory (i.e., we allow the current action to depend on past measurements). However, the main difference is that, instead of memory depending only on the output measurement history, the memory of the controllers and filters we design also depends on the prefix of the missing data pattern seen so far (i.e., on the discrete-state history), which results in significantly improved performance. As an alternative to measurement feedback one can use disturbance feedback [4, 67], which is equivalent to measurement feedback only in special cases [59]. Existing work using disturbance feedback with potentially missing data [4, 67] does not consider dependence on the discrete-state history and our prefix-based parametrization can be used in disturbance feedback controllers/filters as well to improve their performance. Moreover, as a minor difference from the literature, both disturbance feedback and measurement feedback with Q-parametrization are mostly used for optimal control while we use the latter to enforce constraints in a worst-case setting using tools from robust optimization, a problem stated as a future direction in [4]. Finally, the prefix dependence can be interpreted as essentially performing estimation at the discrete-level akin to estimators in hidden mode hybrid systems [45, 160], however in our case the mode is observed (we know whether the measurement is missing or not at each time) but we are trying to estimate the mode-sequence (the missing data pattern). The prefix-based parametrization we propose is also closely related to path-dependent controllers proposed in [85] in the context of disturbance attenuation and stability for Markov jump linear systems, with the difference being that the problems in [85] lead to linear matrix inequalities in the controller gains due to different control objectives, whereas in our setting the problems are non-convex in the filter/controller gains and a nonlinear transformation is needed.

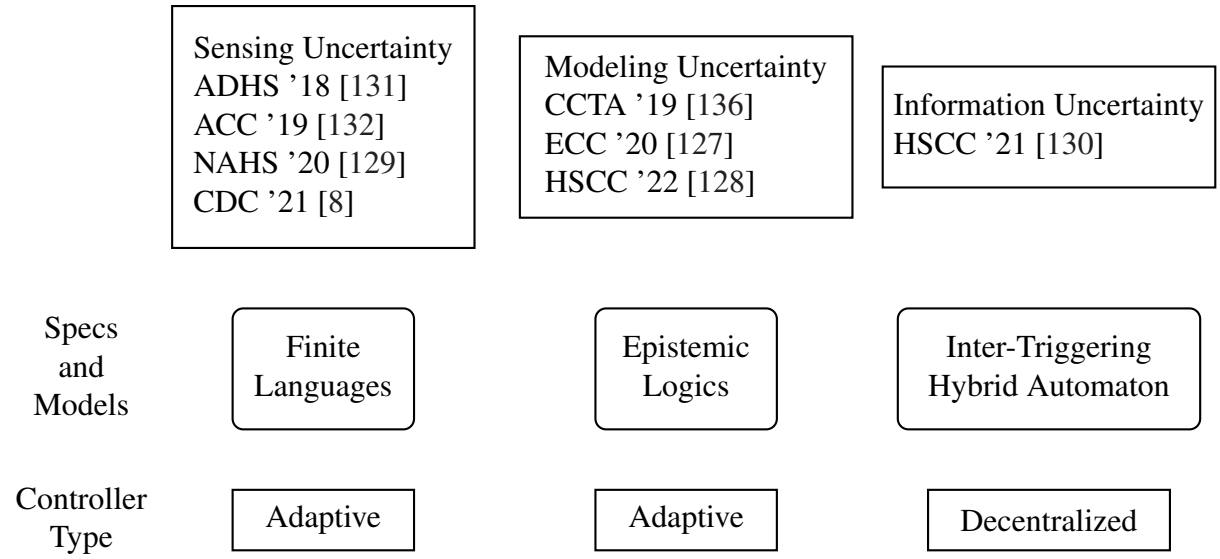


Figure 1.1: A map showing the contributions of the dissertation and categorizes them by the type of controller used, type of hidden mode system and specifications that the system has.

1.3 Summary of Contributions

This dissertation makes contributions in three different areas. It is divided into three different chapters by the different sources of uncertainty used in adaptation as illustrated in Figure 1.1. We contribute to each area in the following ways:

1. On adapting to sensing uncertainty:

    - For linear systems with output feedback and missing observations coming from a missing data pattern, we present a method for designing optimal linear controllers or estimators that satisfy finite horizon reachability objectives. (Chapter 3)
    - The design method is a linear program and it produces a prefix-based controller, which adapts to the missing data pattern chosen by the environment. The missing data is slowly revealed over time in a way that allows the controller to only identify the pattern's prefix at each point in time, thus making the controller prefix-based. (Chapter 3)
    - A finite language-based model is proposed for specifying potential missing data patterns. It is show that feedback controllers that can adapt their gains based on the prefixes of this language can still be synthesized using convex optimization. (Chapter 3)
    - These results are also extended to bounded-error estimation problems in the existence of missing data. In particular, a novel safety criteria, named equalized performance,

that allows the error bound to be relaxed when measurements are missing as long as there is a guarantee to recover the original bound is proposed. Estimators satisfying this criteria are shown to be synthesized using similar techniques. (Chapter 3)

2. On adapting to modeling uncertainty:

   - An estimation based controller architecture is proposed where the estimator does set-membership model identification, and the controller switches its linear, disturbance-feedback gains based on the estimator's output. Such controllers can be designed for appropriate linear systems using bilinear programming. (Chapter 4)
   - These switching adaptive controller strategies can be described by how they trade off between exploration and exploitation with an *exploration-exploitation profile*. When a profile is provided a priori, we present a method for encoding that desired amount of exploration and exploitation into the controller design problem while still producing a bilinear programming problem. (Chapter 4)
   - Instead of designing with respect to an exploration-exploitation profile, design of these adaptive controllers can be done with a temporal logic formula in mind. We show how to design adaptive controllers that satisfy a useful fragment of KLTL formulas using bilinear optimization. (Chapter 5)
   - An approach for designing adaptive controllers that solve safety problems over an infinite time horizon is also presented. This adaptive controller similarly uses a combination of affine mode-discrimination and robust control invariant set selection to achieve safety tasks. (Chapter 6)

3. On adapting to information uncertainty in large multi-agent systems:

   - A modeling paradigm named, inter-triggering hybrid automaton (ITHA), is proposed for modeling dynamically decoupled agents that interact through discrete-triggering actions. (Chapter 7)
   - Compositional safety and responsibility rules are proposed for ITHA agents. It is shown that if all agents follow these local rules, the overall system is guaranteed to be safe. Moreover, if one agent follows these rules, there exist strategies for the rest of the agents to ensure safety of the overall system. This enables designing decentralized controllers or individual controllers with these rules as constraints. (Chapter 7)
   - The effects of changing the information available to each agent at run-time is investigated and different information sharing methods are proposed. It is shown that the safety and responsibility rules adjusted to the given information structure allow local

controllers to automatically become more permissive with increasing informativeness. This allows a level of adaptation at design-time since information structures are assumed to be fixed at run-time. (Chapter 7)

CHAPTER 2

Mathematical Preliminaries

In this chapter, we will present some of the mathematical concepts and ideas that this work relies on. Specifically, we discuss several concepts including polytope manipulation.

We denote the set of real numbers by $\mathbb{R}$, the set of non-negative real numbers $\mathbb{R}_+$ and the set of binary numbers by $\mathbb{B}$. Besides these important exceptions, this dissertation will use calligraphic letters such as $\mathcal{S}$ to denote sets. Throughout this work, the norm $\|\cdot\|$ is assumed to be the infinity norm, i.e., for a vector $v \in \mathbb{R}^n$, $\|v\| \triangleq \max_{i=1,\ldots,n} |v_i|$.

The symbol $\otimes$ represents the Kronecker product, I_k represents the identity matrix of size k, $0_{k\times m}$ represents the $k \times m$ zero matrix, $\mathbb{1}_k$ denotes a k dimensional vector of ones. The subscripts are dropped when the dimension of the matrix is clear from the context. The operator diag : $\mathbb{R}^n \to \mathbb{R}^{n\times n}$ maps a vector $v \in \mathbb{R}^n$ to the $n \times n$ diagonal matrix with the elements of v on its main diagonal. For matrices and vectors, the inequalities $\geq$ are always taken element-wise. For a (block) vector v, v_k and $v_{i:j}$ denote its kth entry, and its sub-vector consisting of entries from ith to jth, respectively.

2.1 Alphabets and Languages

We call any finite set Σ an *alphabet*. In particular, we use $\Sigma = \mathbb{B}$ to represent missing data patterns. Any Σ-valued signal $q = q(t_0)q(t_0+1)\ldots q(t_0+N)$ is called a *word*. The symbol Σ^* denotes the set of all finite-length words (including the empty word), whereas Σ^T and $\Sigma^{[T]}$ denote the set of all words with length equal to T or with length up to T that are formed by elements in Σ, respectively. For a word $q \in \Sigma^*$, its length is denoted by $|q|$. For $i \leq |q|$, we use $q_{[1:i]}$ to denote the length i prefix of q. For example, if $q = q(t_0)q(t_0+1)\ldots q(t_0+N)$, then $q_{[1:i]} = q(t_0)q(t_0+1)\ldots q(t_0+i-1)$. Finally, the set of all non-empty prefixes of q is denoted by $\text{Pref}(q)$. As a concrete example, the word $q = 1001 \in \mathbb{B}^{[4]}$ has length $|q| = 4$, $q_{[1:3]} = 100$ is a prefix, $q_{[2:3]} = 00$ is a subword but not a prefix, and $\text{Pref}(q) = \{1, 10, 100, 1001\}$. An arbitrary set $\mathcal{L}$ of words formed from a given alphabet Σ is called a language over Σ. We overload the $\text{Pref}(\cdot)$ operator and use it for languages

as $\text{Pref}(\mathcal{L}) \doteq \cup_{q \in \mathcal{L}} \text{Pref}(q)$.

For a given set $\mathcal{S}$, $\mathcal{P}_{\geq n}(\mathcal{S})$ denotes the set of subsets of $\mathcal{S}$ with at least n elements.

One important set in this dissertation will be the polytope.

2.2 Polytopes

Throughout this work, there are many sets of continuous variables that are represented as polytopes. A *polytope* is a bounded, convex set of vectors which satisfy a set of half-space constraints (i.e. $\mathcal{P} = \{x \in \mathbb{R}^n \mid Hx \leq h\}$ where $H \in \mathbb{R}^{q \times n}$ and $h \in \mathbb{R}^q$). The representation of the polytope as a set of half-space constraints is called the H-Representation and always exists for each polytope.

The following results relate polytopes to one another or describe properties of polytopes. These results will be useful when reasoning about system trajectories and how to formulate optimal control constraints. The first lemma identifies the conditions under which a polytope is a subset of another polytope.

Lemma 1 (Polytope Containment [97])**.** *Consider the following two polytopes $\mathcal{X} = \{x \in \mathbb{R}^n \mid H_{\mathcal{X}} x \leq h_{\mathcal{X}}\}$ and $\mathcal{Y} = \{y \in \mathbb{R}^n \mid H_{\mathcal{Y}} y \leq h_{\mathcal{Y}}\}$ where $H_{\mathcal{X}} \in \mathbb{R}^{q_x \times n}$ and $H_{\mathcal{Y}} \in \mathbb{R}^{q_y \times n}$. Polytope $\mathcal{Y}$ contains $\mathcal{X}$ (i.e. $\mathcal{X} \subseteq \mathcal{Y}$) if and only if there exists a matrix $\Lambda \in \mathbb{R}_+^{q_x \times q_y}$ such that:*

$$\Lambda H_{\mathcal{X}} = H_{\mathcal{Y}} \tag{2.1a}$$

$$\Lambda h_{\mathcal{X}} \leq h_{\mathcal{Y}}. \tag{2.1b}$$

The second property is one version of the well-known Farkas Lemma.

Lemma 2 (Farkas Lemma [47])**.** *Consider the matrices $H \in \mathbb{R}^{q \times n}$, $h \in \mathbb{R}^{q_H}$. One and only one of the following two conditions is true:*

- *There exists an $x \in \mathbb{R}^n$ such that $Hx \leq h$, or*
- *There exists a $\lambda \in \mathbb{R}^q$ such that $H^\top \lambda = 0$, $h^\top \lambda < 0$, and $\lambda \geq 0$.*

This property allows us to identify constraints for a polytope $\mathcal{X} = \{x \in \mathbb{R}^n \mid H_{\mathcal{X}} x \leq h_{\mathcal{X}}\}$ being nonempty (first condition) or empty (second condition).

2.3 Block Triangular Matrices

In this section, we present some properties of block triangular matrices that are used to develop optimization results later in the dissertation. Several results in this section are presented without

proof. The proofs are located in Appendix A.1.

Recall the definition of a triangular matrix:

Definition 2 (Lower Triangular Matrix, [87])**.** *The matrix $A \in \mathbb{R}^{n\times n}$ is lower triangular if $A_{ij} = 0$ for all $i < j$.*

A block (lower) triangular matrix will then be defined as follows:

Definition 3 (Block (Lower) Triangular Matrix)**.** *The matrix $A \in \mathbb{R}^{nl\times mp}$ is $l \times p$ block lower triangular if each block $\tilde{A}_{ij} = 0$ for all $i < j$ where $\tilde{A}_{ij} = A_{[il+1:(i+1)l],[jp+1:(j+1)p]}$ (i.e. $\tilde{A}_{ij}$ is the (i, j)-th $(l \times p)$ block of A.*

We can decompose the block triangular matrix using a concept analogous to the principal leading sub-matrices of a lower triangular matrix.

Definition 4 (Leading Principal Block Submatrix)**.** *The i-th* leading principal block submatrix *of a $l \times p$ block matrix $X \in \mathbb{R}^{al\times bp}$, denoted by $\mathcal{BM}_i(X)$, is the $l \times p$ block matrix:*

$$\mathcal{BM}_i(X) = X_{[1:il],[1:ip]}$$

for all $i \in [1, min(a, b)]$, where $X_{[1:il],[1:ip]}$ indicates the submatrix formed by all entries of matrix X that are in both the first il rows and the first ip columns.

Several useful properties of the leading principal block matrix operator $\mathcal{BM}_i(\cdot)$ for block lower triangular matrices are stated next. Proofs of these statements can be found in A.1.

Lemma 3. *Let $W, X \in \mathbb{R}^{ap\times bq}$, $Y \in \mathbb{R}^{bq\times cr}$ and $Z \in \mathbb{R}^{as\times as}$. The following properties hold:*

1. *$\mathcal{BM}_i(W + X) = \mathcal{BM}_i(W) + \mathcal{BM}_i(X)$;*
2. *If X and Y are $p \times q$ and $q \times r$ block lower triangular, respectively, then $\mathcal{BM}_i(XY) = \mathcal{BM}_i(X)\mathcal{BM}_i(Y)$;*
3. *If Z is nonsingular and $s \times s$ block lower triangular, then $\mathcal{BM}_i(Z^{-1}) = (\mathcal{BM}_i(Z))^{-1}$,*

for all $i \in [1, min(a, b, c)]$.

Proposition 1. *Let $\bar{C}^{(1)}$ and $\bar{C}^{(2)}$ be $p \times n$ block lower triangular matrices that share the same j-th leading block principal submatrix:*

$$\mathcal{BM}_j(\bar{C}^{(1)}) = \mathcal{BM}_j(\bar{C}^{(2)}).$$

Also let $F^{(1)}$, $F^{(2)}$ be $n \times p$ block lower triangular matrices and let S be an $n \times n$ block lower triangular matrix, all with compatible block sizes. Define, for $i \in \{1, 2\}$,

$$Q^{(i)} \doteq F^{(i)}(I - \bar{C}^{(i)} S F^{(i)})^{-1}. \tag{2.2}$$

Then,

$$\mathcal{BM}_j(F^{(1)}) = \mathcal{BM}_j(F^{(2)}) \in \mathbb{R}^{jn \times jp}$$

if and only if

$$\mathcal{BM}_j(Q^{(1)}) = \mathcal{BM}_j(Q^{(2)}) \in \mathbb{R}^{jn \times jp}.$$

Proposition 2. *Consider the following pairs of matrices $(\bar{C}^{(1)}, \bar{C}^{(2)})$ and $(Q^{(1)}, Q^{(2)})$ that share the same j-th principal block leading submatrix amongst each pair*

$$\mathcal{BM}_j(\bar{C}^{(1)}) = \mathcal{BM}_j(\bar{C}^{(2)}),$$
$$\mathcal{BM}_j(Q^{(1)}) = \mathcal{BM}_j(Q^{(2)})$$

and consider two vectors $f^{(1)}$ and $f^{(2)}$ and a block lower triangular matrix S. Define, for $i \in \{1, 2\}$:

$$r^{(i)} = (I + Q^{(i)} \bar{C}^{(i)} S) f^{(i)}. \tag{2.3}$$

Then, the vectors $f^{(1)}$ and $f^{(2)}$ satisfy:

$$f_k^{(1)} = f_k^{(2)} \quad \forall k \in [1, jn]$$

if and only if the first jn entries of the vector $r^{(1)}$ is identical to that of $r^{(2)}$:

$$r_k^{(1)} = r_k^{(2)} \quad \forall k \in [1, jn].$$

2.4 Types of Dynamical Systems

The discrete-time hybrid system definition given in (1.1) and (1.2) can describe many processes in the real world (e.g., walking robots, collections of vehicles moving on a highway). This ability to express many different types of processes makes it often hard to apply formal methods principles to the general system, so we occasionally restrict our attention to systems with a particular form. Several of the forms discussed in this dissertation are briefly introduced below for the sake of completeness. To help with further investigate each of these restricted forms, a list of related references will be provided at the end of each subsection for the interested reader.

Frequently, we will refer to sequences of states of a dynamical system. Such a sequence $x(t_0), x(t_0+1), ..., x(t_0+T)$ will be abbreviated with $x(t_0 : t_0+T)$. For computational reasons, this can be represented as a single vector (i.e. if each state $x(k) \in \mathbb{R}^n$, then $x(t_0 : t_0 + T) \in \mathbb{R}^{n(T+1)}$).

2.4.1 Transition Systems

Definition 5 (Transition System [14])**.** *A* transition system TS *is a tuple* $TS = (S, Act, \rightarrow, I, AP, L)$ *where*

- S *is a set of states,*
- Act *is a set of actions,*
- $\rightarrow \subseteq S \times Act \times S$ *is a transition relation,*
- $I \subseteq S$ *is a set of initial states,*
- AP *is a set of atomic propositions, and*
- $L : S \rightarrow 2^{AP}$ *is a labeling function.*

The atomic propositions AP and labelling function L are critical to identifying whether or not there exists a policy that satisfies a desired goal/task. A policy here can be a function which takes the current state and takes an action that approaches some goal proposition $\gamma : S \rightarrow Act$. A policy can also take on other forms (e.g. it can observe its *history of previous states* and use that to make a decision on how to act).

This relates to the hybrid system (1.1) and (1.2) in the following ways:

- There is no unknown parameter θ.
- There is no continuous state x.
- The state of the system is directly observed (i.e. $y(t) = \omega(t, q(t)) = q(t)$).
- The dynamics are non-deterministic and explained by $\rightarrow$, (i.e. $(q(t), u(t), f(q(t), u(t))) \in \rightarrow$).

As alluded to in the definition, please see the textbook [14] for more information about transition systems. Adaptive transition systems (an extension that matches some of what is discussed in this dissertation) are discussed briefly in [133].

2.4.2 Linear Systems

2.4.2.1 Linear Systems with Missing Data

We consider discrete-time affine systems with state update and measurement equations defined as:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + w(t) + k,\ w(t) \in \mathcal{W}, \\ y(t) &= \begin{cases} Cx(t) + v(t), & q(t) = 1, \\ \emptyset, & q(t) = 0, \end{cases} \quad v(t) \in \mathcal{V}, \end{aligned} \tag{2.4}$$

where A, B, C, k are known system matrices, $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ is the continuous state, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ is the input, $w(t) \in \mathcal{W} \subseteq \mathbb{R}^n$ is the process noise, $y(t) \in \mathcal{Y} \subseteq \mathbb{R}^p \cup \{\emptyset\}$ is the output measurements of the system, $q(t) \in \mathbb{B}$ is the discrete state/mode of the system, with $q(t) = 1$ denoting that the measurement vector is available and $q(t) = 0$ denoting that the measurement vector is not available (i.e., "missing"), and $v(t) \in \mathcal{V} \subseteq \mathbb{R}^p$ is the measurement noise. The noise terms $w(t)$ and $v(t)$ are unknown but bounded, and their bounds are known (i.e., $\mathcal{W} = \{w \in \mathbb{R}^n \mid \|w\| \leq \eta_w\}$ and $\mathcal{V} = \{v \in \mathbb{R}^p \mid \|v\| \leq \eta_v\}$). Moreover, we assume that the control input $u(t)$ is bounded (i.e., $\mathcal{U} = \{u \in \mathbb{R}^m \mid \|u\| \leq \eta_u\}$).

This relates to the hybrid system (1.1) and (1.2) in the following ways:

- The unknown parameter θ is the missing data pattern.
- The state of the system is noisily observed when the data pattern allows (i.e., when $q(t) = 1$) but returns empty set when the data pattern enforces it (i.e. when $q(t) = 0$).
- The dynamics and measurement equation are normally linear, (i.e. $f_\theta(x(t), u(t), w(t)) = A_\theta x(t) + B_\theta u(t) + w(t)$ and $\omega_\theta(t, x(t), v(t)) = C_\theta x(t) + v(t)$ when $q(t) = 1$).

The definition of this system will be repeated in Chapter 3 for utility.

2.4.2.2 Linear Systems with Unknown Parameters

$$\begin{aligned} x(t+1) &= A_\theta x(t) + B_\theta u(t) + w(t), \quad w(t) \in \mathcal{W}_\theta \\ y(t) &= C_\theta x(t) + v(t), \quad v(t) \in \mathcal{V}_\theta \end{aligned} \tag{2.5}$$

where θ is the mode of the system taking values in a finite set Θ of models (i.e. $\theta \in \Theta$), x_t is the state of the system taking values in $\mathcal{X}$, u_t is the input at time t that must satisfy input constraints represented by polytope $\mathcal{U}$ (i.e. $u_t \in \mathcal{U}$), w_t is the unmeasured disturbance to the system that lies in the polytope $\mathcal{W}_\theta$ ($w_t \in \mathcal{W}_\theta$). The system starts with a fixed, yet unknown, mode θ. It is assumed that the set Θ (including the matrices A_θ, B_θ and polytopes $\mathcal{W}_\theta$ for each mode) and the polytope $\mathcal{U}$ are known.

This relates to the hybrid system (1.1) and (1.2) in the following way:

- There is no discrete state q.
- The dynamics and measurement equations are linear, (i.e. $f_\theta(x(t), u(t), w(t)) = A_\theta x(t) + B_\theta u(t) + w(t)$ and $\omega_\theta(t, x(t), v(t)) = C_\theta x(t) + v(t)$).

2.4.3 Piecewise-Affine (PWA) Systems

A discrete-time, piece-wise affine (PWA) system is a dynamical system

$$x(t+1) = f(x(t), u(t), w(t)) \tag{2.6}$$

defined over a partition $\cup_i \mathcal{D}^i = \mathcal{X}$ such that it is affine over each element of the partition $\mathcal{D}_i$. In math, a PWA system f can be defined as a set $f = \{(f^i, \mathcal{D}^i)\}_{i=1}^m$ with

$$f^i(x(t), u(t), w(t)) = A^i x(t) + B^i u(t) + B^i_w w(t) + F \tag{2.7}$$

and for all states $x(t) \in \mathcal{D}^i$

$$f(x(t), u(t), w(t)) = f^i(x(t), u(t), w(t)) \tag{2.8}$$

$x(t)$ is the state of the system, $u(t)$ is the controlled input to the system, $w(t)$ is the uncontrolled input (disturbance), and the matrices (A, B, B_w, F) are of the appropriate dimensions. The state space, space of allowed inputs, and the space of feasible uncontrolled inputs are referred to as $\mathcal{X}$, $\mathcal{U}$, and $\mathcal{W}$, respectively.

2.5 Invariant Sets

Controlled invariant sets play an important role in ensuring safety of systems with dynamics of the form (1.1). Formally, a robust controlled invariant set (RCIS) $\mathcal{C}_{\text{inv}}$ inside a given safe set $\mathcal{X}_{\text{safe}} \subseteq \mathcal{X}$ (i.e., $\mathcal{C}_{\text{inv}} \subseteq \mathcal{X}_{\text{safe}}$) is a set of states that satisfies:

$$x \in \mathcal{C}_{\text{inv}} \Rightarrow \exists u \in \mathcal{U} \;\; \forall d \in \mathcal{D} \;\; f(x, u, d) \in \mathcal{C}_{\text{inv}}. \tag{2.9}$$

In words, this means that if the state $x(t)$ is in $\mathcal{C}_{\text{inv}}$, there is an input $u(t)$ to ensure that $x(t+1)$ will be in $\mathcal{C}_{\text{inv}}$ for any disturbance within given bounds, thus allowing the states to stay in $\mathcal{C}_{\text{inv}}$ indefinitely.

2.5.1 Invariant Set Computation

There are many methods in the literature for computing or approximating controlled invariant sets [22, 26, 43, 126]. The main computational building block of these algorithms is the *one-step backward reachable set operation*, that we denote as $\text{Pre}(\cdot)$. For a given set R and dynamics f, the one-step robust backward reachable set of $\mathcal{R}$ under f is defined as

$$\text{Pre}^f(\mathcal{R}) = \{x \in \mathcal{X} \mid \exists u \in \mathcal{U} : \ f(x, u, \mathcal{W}) \subseteq \mathcal{R}\}. \tag{2.10}$$

Given a safe set $\mathcal{Q}_{safe}$, under mild conditions, the following iterations converge from outside to the maximal controlled invariant set in $\mathcal{Q}_{safe}$ when initialized with $\mathcal{C}_0 = \mathcal{Q}_{safe}$:

$$\mathcal{C}_{i+1} = \text{Pre}^f(\mathcal{C}_i) \cap \mathcal{Q}_{safe}. \tag{2.11}$$

If the update rule reaches a fixed point, i.e., $\mathcal{C}_i = \text{Pre}^f(\mathcal{C}_i) \cap \mathcal{Q}_{safe}$, then the solution to that equation is the maximal invariant set contained in $\mathcal{Q}_{safe}$. On the other hand, although this is a monotonically non-increasing (in the set inclusion sense) sequence, the iterations are not guaranteed to terminate in finitely many steps, a problem that can be mitigated by approximation techniques [43, 126].

Alternatively, if one has an initial simple RCIS $\mathcal{C}_0$, computed either analytically or numerically, contained in some safe set Q_{safe}, this set can be progressively expanded again via the same update rule (2.11). In this case, we obtain a monotonically non-decreasing sequence of sets $\Gamma_k \doteq \bigcup_{i=1}^{k} \mathcal{C}_i$, each of which themselves are robustly controlled invariant. Therefore, it can be terminated at anytime and one would obtain an RCIS. We call this method the *inside-out algorithm*.

Crucially, for PWA systems and sets described with unions of polytopes, the invariant set computation reduces to a set of polytopic operations. Moreover, when finding the exact $\text{Pre}(\cdot)$ is computationally hard, using an under-approximation does not compromise correctness when using the iterative algorithms in the sense that upon termination, the algorithm still results in an RCIS.

2.5.2 Graph Notation

A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a tuple containing a set of vertices $\mathcal{V}$ and a set of directed edges $\mathcal{E}$. Note that each edge is an ordered pair $(v_1, v_2) \in \mathcal{E}$ of vertices from $\mathcal{V}$ (i.e. $v_1, v_2 \in \mathcal{V}$). We say that a vertex v_1 is *connected* to a vertex v_2 if and only if $(v_1, v_2) \in \mathcal{E}$. By the nature of directed graphs, v_1 being connected to v_2 does not mean that v_2 is connected to v_1.Within directed graphs the *inward connections* of a vertex v can be defined as follows:

$$in_{\mathcal{G}}(v) = \{v' \in \mathcal{V} \mid (v', v) \in \mathcal{E}\}.$$

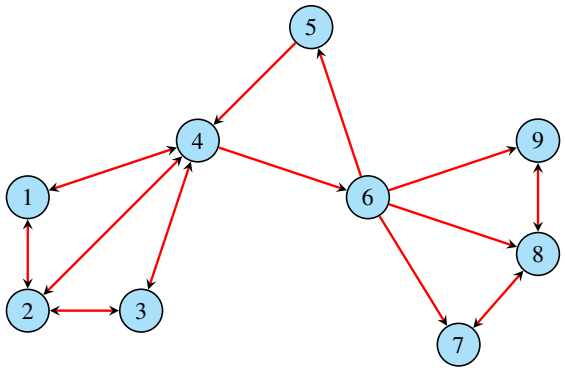


Figure 2.1: An example directed graph $\mathcal{G}$ that we use to demonstrate potential links between processors in a server farm.

The *outward connections* of a vertex v can be defined similarly:

$$out_{\mathcal{G}}(v) = \{v' \in \mathcal{V} \mid (v, v') \in \mathcal{E}\}.$$

CHAPTER 3

Correct-By-Construction Control with Missing Data

3.1 Introduction

Recall that one of the real-world limitations of CPSs is that sensor information may be delayed or lost when sent to a decision maker by things such as communication queues in networks. Some amount of delay is tolerable, as we see in settings like human driving where a small amount of delay in the speedometer's value does not lead to driving failures. Thus, safety in the presence of sensor uncertainties is typically possible for some classes of uncertainty. Defining classes of uncertainty for missing data systems, or systems where sensor measurements can be lost when transmitted to the decision, has been discussed in the past, however has not been applied to guaranteed reachability tasks. This chapter attempts to use convex optimization to design estimators and controllers for guaranteed reachability tasks where the class of uncertainty is defined by a missing data language (a missing data language is a language over the set $\mathbb{B}$ as defined in Section 2.1).

3.2 Problem Statement

As alluded to in the introduction, the goal of this chapter is to design feedback control or estimation mechanisms that are robust to missing measurements. This section describes the system model considered, including the missing data patterns. Then, we formally state the problem.

3.2.1 Model Description

We consider discrete-time affine systems with state update and measurement equations defined as:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + w(t) + k,\ w(t) \in \mathcal{W}, \\ y(t) &= \begin{cases} Cx(t) + v(t), & q(t) = 1, \\ \emptyset, & q(t) = 0, \end{cases} \quad v(t) \in \mathcal{V}, \end{aligned} \tag{3.1}$$

where A, B, C, k are known system matrices, $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ is the continuous state, $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$ is the input, $w(t) \in \mathcal{W} \subseteq \mathbb{R}^n$ is the process noise, $y(t) \in \mathcal{Y} \subseteq \mathbb{R}^p \cup \{\emptyset\}$ is the output measurements of the system, $q(t) \in \mathbb{B}$ is the discrete state/mode of the system, with $q(t) = 1$ denoting that the measurement vector is available and $q(t) = 0$ denoting that the measurement vector is not available (i.e., "missing"), and $v(t) \in \mathcal{V} \subseteq \mathbb{R}^p$ is the measurement noise. The noise terms $w(t)$ and $v(t)$ are unknown but bounded, and their bounds are known (i.e., $\mathcal{W} = \{w \in \mathbb{R}^n \mid \|w\| \leq \eta_w\}$ and $\mathcal{V} = \{v \in \mathbb{R}^p \mid \|v\| \leq \eta_v\}$). Moreover, we assume that the control input $u(t)$ is bounded (i.e., $\mathcal{U} = \{u \in \mathbb{R}^m \mid \|u\| \leq \eta_u\}$).

Remark 1. *We note that the methods developed in this chapter can be extended to work with systems that have time-varying matrices in the place of A, B, C, f. However, this case is omitted for clarity of notation. Similarly, the sets $\mathcal{W}$, $\mathcal{V}$ and $\mathcal{U}$ can be arbitrary polytopes but for simplicity, we constrain them to be hypercubes in the rest of the chapter.*

The performance of any controller or estimator designed for the system in (3.1) clearly depends on how much information is received, and yet the evolution of the discrete state $q(t)$ is not included in (3.1). To model this, we introduce a constraint on the evolution of $q(t)$. If the evolution of the discrete state is not constrained at all, one possibility is $q(t) = 0$ for all times, and there is no measurement for feedback. However, in many applications, it is possible to have a priori information about admissible missing data patterns (e.g., based on a network device's specification sheet or the knowledge about communication protocols that are packet-drop tolerant). To describe the evolution of the discrete state $q(t)$, we introduce the missing data language $\mathcal{L}$:

Definition 6 (Missing Data Language)**.** *A **missing data language** $\mathcal{L} \subseteq \mathbb{B}^T$ is a set of words q, called a missing data pattern, that describes the possible trajectories of $q(t)$ in the system* (3.1)*. A mode signal $q = q(t_0), q(t_0+1), \ldots, q(t_0+T-1)$ is said to **satisfy** the missing data language $\mathcal{L}$ if $q \in \mathcal{L}$.*

Remark 2. *Other works in the literature have considered representing missing data patterns or sequences with concepts such as (m, k)-firmness [77, 49] or a bound on the number of consecutive missing data packets. The representation that we describe here is general enough that it can express both of these concepts.*

3.2.2 Problem Statement

In what follows, we describe several constrained control and estimation problems for the system in (3.1). Designing a controller that enforces certain state constraints or an estimator that guarantees that estimation error remains bounded is particularly challenging when the controller or estimator does not have access to all measurements for all times. Therefore, we introduce a relaxed invariance-type objective that we call equalized recovery. Finally, we present a formal unified problem statement.

An observer or an estimator $\mathcal{O} : (\mathcal{Y} \times \mathcal{U} \times \mathbb{B})^* \mapsto \mathcal{X}$ maps the measured input/output sequence and the missing data sequence to an estimate $\hat{x}$ of the state. In particular, we will consider Luenberger-like estimator structures of the form:

$$\begin{aligned} \hat{x}(t+1) &= A\hat{x}(t) + Bu(t) - u_e(t) + k, \\ \hat{y}(t) &= C\hat{x}(t), \end{aligned} \tag{3.2}$$

where the injection term $u_e(t)$ is the design variable. The constrained estimation problem aims to impose constraints on the estimation error $e(t) = x(t) - \hat{x}(t)$ by appropriately designing $u_e(t)$.

Another problem of interest is that of synthesizing a controller $\mathcal{C} : (\mathcal{Y} \times \mathbb{B})^* \mapsto \mathcal{U}$, where the design variable is $u(t)$ in (3.1). The goal is to impose constraints on the state of the closed-loop system, while it is required to respect the constraints on the input imposed by the set $\mathcal{U}$. It is possible to pose a similar problem for tracking control, where there is a given desired trajectory $x_d(t_0), x_d(t_0+1), \ldots, x_d(t_0+T)$ and its corresponding $u_d(t_0), u_d(t_0+1), \ldots, u_d(t_0+T-1)$ such that $x_d(t+1) = Ax_d(t) + Bu_d(t) + k$, and the objective is to guarantee constraints on the tracking error $\xi(t) \doteq x(t) - x_d(t)$. One can also consider imposing constraints on affine functions of the states.

Both estimation and control synthesis problems can be mapped (with slight modifications) to a generic constrained control problem on a unified system:

$$\begin{aligned} \xi(t+1) &= A\xi(t) + B_\xi u_\xi(t) + w(t) + k_\xi,\ w(t) \in \mathcal{W}_\xi, \\ y_\xi(t) &= \begin{cases} C\xi(t) + v(t), & q(t) = 1, \\ \emptyset, & q(t) = 0, \end{cases} \quad v(t) \in \mathcal{V}_\xi, \end{aligned} \tag{3.3}$$

where the transformed state $\xi(t)$, the transformed output $y_\xi(t) \in \mathcal{Y}_\xi$ and the transformed input $u_\xi(t) \in \mathcal{U}_\xi$, as well as B_ξ, k_ξ, $\mathcal{U}_\xi$, and $\mathcal{Y}_\xi$ represent different signals, matrices and sets depending on the problem of interest (for the sake of completeness, they are provided in A.2). The proper objective for both problems is then to design a feedback law for $u_\xi(t)$ as a function of all previous outputs $\{y_\xi(\tau)\}_{\tau=t_0}^{t}$ and discrete states $\{q(\tau)\}_{\tau=t_0}^{t}$ such the states of the system (3.3) satisfies

certain constraints.

Finally, as mentioned earlier, when the measurements can be missing as in (3.3), it is not reasonable to expect that the constraints hold invariantly. For instance in an estimation or tracking problem, it might be reasonable to allow a larger bound during missing data events. To capture this, we define a new type of constraint that relaxes invariance.

Definition 7 (Equalized Recovery)**.** *A discrete-time dynamical system as in* (3.3) *is said to achieve an equalized recovery level M_1 with recovery time T and intermediate level $M_2 \geq M_1$ at time t_0 if for any initial state with $\|\xi(t_0)\| \leq M_1$, we have $\|\xi(t)\| \leq M_2$ for all $t \in [t_0, t_0 + T]$ and $\|\xi(t_0 + T)\| \leq M_1$.*

Equalized recovery expresses a form of boundedness for the trajectories of the system and can be viewed as a form of weak "invariance", where instead of enforcing the set $\mathcal{X}_1 \doteq \{x \mid \|x\| \leq M_1\}$ being invariant, we relax the invariance condition and allow the states to be in $\mathcal{X}_2 \doteq \{x \mid \|x\| \leq M_2\}$ as long as they recover back to the set $\mathcal{X}_1$. Equalized recovery's interpretation as weak invariance is somewhat related to multi-set invariance discussed in [10] for switched systems. For estimation problems, in the special case where $M_1 = M_2$ and $T = 1$, equalized recovery reduces to equalized performance [27, 28, 102], which essentially states that $\|\xi(t)\| \leq M_1$ should be invariant with ξ being the estimation error.

Remark 3. *Instead of imposing equalized recovery on the state $\xi(t)$, it also possible to consider equalized recovery on a subset or a linear combination of the state, i.e., $z(t) = L\xi(t)$, if desired. The proposed synthesis solution in Section 3.3 can be slightly modified to account for this in a straightforward manner. Furthermore, in addition to the infinity norm, we can also consider other "set templates" such as zonotopes with minor modifications to our proposed solution.*

Given all these elements, the problem we are interested in can be formally stated as follows:

Problem 1. *Consider a missing data language $\mathcal{L} \subseteq \mathbb{B}^T$, and a system of the form* (3.3)*, whose mode signal $q(t)$, $t \in [t_0, t_0 + T - 1]$ satisfies the missing data language $\mathcal{L}$. Given the recovery level M_1, intermediate level $M_2 \geq M_1$, and recovery time T, find a feedback law $\Gamma : (\mathcal{Y}_\xi \times \mathbb{B})^* \mapsto \mathcal{U}_\xi$ such that the system achieves an equalized recovery level M_1 with recovery time T and intermediate level M_2.*

3.3 Synthesis of a Prefix-Based Feedback

This section addresses the feedback synthesis problem in Problem 1 by developing convex optimization problems that can construct affine feedback laws. We start with a commonly used

time-based affine feedback law and show how language constraints can be integrated in this. Then, we present a new feedback structure that updates the gains based on the prefix of the missing data pattern seen so far and show that prefix-based feedback laws generalize the time-based ones and, moreover, that the synthesis of prefix-based affine feedback laws can be reduced to a convex optimization problem. The proofs of the main results are provided in A.3.

3.3.1 Time-Based Feedback Laws and Their Limitations

A common structure for feedback laws, which we call *time-based*, takes the following form [147]:

$$u_\xi(t) = f(t) + \sum_{\tau=t_0}^{t} F_{(t,\tau)} y_\xi(\tau). \tag{3.4}$$

In this chapter, we temporarily ignore the broad definition of f from Chapter 2; temporarily, this will simply be a time-varying function $f : \mathbb{N} \to \mathbb{R}^m$.

Note that if there is a single missing data pattern $q^{(*)}$ (i.e., the language $\mathcal{L}$ has only one word), one could interpret the system in (3.3) as a linear time-varying system instead of a switched system, by defining a time-varying measurement matrix $C(t) = 0$ when $q^{(*)}(t) = 0$, and $C(t) = C$ when $q^{(*)}(t) = 1$, which essentially sets the output to 0 when it is missing. If we define $u_\xi \doteq \begin{bmatrix} u_\xi(t_0)^\top, & u_\xi(t_0+1)^\top, & \cdots & u_\xi(t_0+T-1)^\top \end{bmatrix}^\top$ associated with a word $q^{(*)}$, the feedback laws in (3.4) can be written in matrix–vector form as follows:

$$u_\xi = f^{(*)} + F^{(*)} y_\xi,$$

where

$$f^{(*)} \doteq \begin{bmatrix} f(t_0)^\top & f(t_0+1)^\top & \cdots & f(t_0+T-1)^\top \end{bmatrix}^\top \tag{3.5}$$

and

$$F^{(*)} \doteq \begin{bmatrix} F_{(t_0,t_0)} & 0 & \cdots & 0 \\ F_{(t_0+1,t_0)} & F_{(t_0+1,t_0+1)} & & \vdots \\ \vdots & & \ddots & 0 \\ F_{(t_0+T-1,t_0)} & F_{(t_0+T-1,t_0+1)} & \cdots & F_{(t_0+T-1,t_0+T-1)} \end{bmatrix}. \tag{3.6}$$

The design of feedback laws of the form (3.4), or equivalently finding $(F^{(*)}, f^{(*)})$ that guarantees certain convex constraints on the state and input, is in general a non-convex problem due to the states of the closed-loop system being a non-convex function of the gains $(F^{(*)}, f^{(*)})$. Nevertheless, a non-linear change of variables, namely Q-parametrization, is used in [147] to render the design of such feedback laws a convex problem.

Our previous work [131] introduces a method for representing any given language $\mathcal{L}$ with a more difficult language $\mathcal{L}^*$ that satisfies $|\mathcal{L}^*| = 1$. We do this by introducing a partial order for words $q^{(1)}, q^{(2)} \in \mathbb{B}^T$ for arbitrary T as:

$$q^{(1)} \preceq q^{(2)} \quad \iff \quad (\forall i \in [1, T])(q^{(1)}_{[i]} = 0 \implies q^{(2)}_{[i]} = 0).$$

With this partial order, one can derive a word that is uniquely "harder" or "more missing" than any in the given language $\mathcal{L}$, in the sense of being the least upper bound for the set $\mathcal{L}$. We refer to this least upper bound as the worst-case word q^* and the worst-case language is then given by $\mathcal{L}^* \doteq \{q^*\}$. Solving for gains $(F^{(*)}, f^{(*)})$ associated with $\{q^*\}$ that guarantees an equalized recovery level, then, guarantees the same equalized recovery level for any missing data pattern in $\mathcal{L}$ when these gains are used for feedback [131].

However, the timed-based solution based on the use of the worst-case language $\mathcal{L}^*$ has some limitations. Consider the following language $\mathcal{L} = \{q^{(1)}, q^{(2)}\}$ with $q^{(1)} = 1011$ and $q^{(2)} = 1101$. In this case, the worst-case language $\mathcal{L}^*$ can be identified by performing a bitwise AND of $q^{(1)}$ and $q^{(2)}$: $\mathcal{L}^* = \{1001\}$.

Note that in the above example with the language $\mathcal{L} = \{q^{(1)}, q^{(2)}\}$, the discrete state's value at time $t = t_0 + 1$ directly describes the word in $\mathcal{L}$ that is being executed. i.e., if $q(t_0 + 1) = 1$, then we know that trajectory $q^{(2)}$ of the discrete state is occurring; otherwise, $q^{(1)}$ is occurring. So, if we make that identification at time t we use a set of feedback gains that are specific to an individual word after the second time step instead of the feedback gains designed with $\mathcal{L}^*$. In such a scheme, the feedback would only need to be open-loop on one time instance. Using a new set of gains that were specific to a single word in $\mathcal{L}$ instead of the worst-case word q^* would obviously reduce the recovery level in this case and across a broad number of other examples. On the other hand, time-based feedback ignores the observed mode sequence so far and tries to be robust against all words in the language in an open-loop (in the discrete mode) fashion.

The above might indicate that one could simply synthesize a feedback gain for each word q in the language $\mathcal{L}$ and choose the proper gain at runtime to arrive at a less conservative solution, but the selection of a proper gain can be tricky even in the simple example that was shown above. Ultimately, we cannot select the exact gain that we need in such a solution because we cannot know the word that is being executed at the beginning of the time horizon. For instance, in the example language above, it is impossible to discern which word is occurring after receiving the measurement at time t_0 of $q(t_0) = 1$ because 1 is a prefix of both words in $\mathcal{L}$. This motivates a new type of prefix-based feedback structure that we introduce in the next subsection.

3.3.2 Prefix-Based Feedback Laws

To overcome the limitations of the time-based feedback laws, the feedback laws for the controllers/estimators should have some understanding of the currently observed sequence of the discrete state, which we capture by the following feedback structure,

$$u_\xi(q_{[t_0:t]}) = f(t, q_{[t_0:t]}) + \sum_{\tau=t_0}^{t} F_{(t,\tau,q_{[t_0:t]})} y_\xi(\tau), \tag{3.7}$$

We call this a *prefix-based* feedback law. Similarly, estimators (3.2) and controllers with injection terms or outputs in the form (3.7) are called *prefix-based estimators* and *prefix-based controllers*, respectively.

While the time-based solutions use the available (non-missing) output measurement history, $\{y_\xi(\tau)\}_{\tau=t_0}^{t}$, for feedback, prefix-based feedback laws use both the output history and the discrete-state history, $\{y_\xi(\tau)\}_{\tau=t_0}^{t}$ and $\{q(\tau)\}_{\tau=t_0}^{t}$. By its definition, it essentially also performs estimation at the discrete-level (or online model detection) to detect which missing data pattern in $\mathcal{L}$ is active and adapts the filter gains accordingly. Whereas, the time-based estimator/controller is agnostic to the missing data pattern and tries to be robust rather than adaptive. The following proposition formally captures the fact that prefix-based estimators/controllers are more general than time-based estimators/controllers.

Proposition 3. *For any time-based feedback law for the dynamical system in* (3.3) *with missing data pattern given by a fixed-length language $\mathcal{L}$, identical performance can be obtained using a prefix-based feedback law.*

Proof. Let the transformed input term for the time-based feedback law be

$$u_\xi(t) = \bar{f}(t) + \sum_{\tau=t_0}^{t} \bar{F}_{(t,\tau)} y_\xi(\tau). \tag{3.8}$$

Define the gains of the prefix-based feedback law's transformed input term in (3.7) as $f(t, \lambda) \doteq \bar{f}(t)$, $F_{(t,\tau,\lambda)} \doteq \bar{F}_{(t,\tau)}$ for all $t \in [t_0, t_0 + T - 1]$, $\tau \in [t_0, t]$ and for all $\lambda \in \bigcup_{q\in\mathcal{L}} Pref(q)$. Then the two feedback laws are equivalent. □ □

In order to design prefix-based feedback gains, we associate with each word $q^{(i)} \in \mathcal{L}$ a pair of gain matrices $(F^{(i)}, f^{(i)})$ defined as in (3.5)–(3.6). However, since at run-time we do not know which word $q^{(i)}$ is active, we enforce some constraints on the gain matrices of words sharing

prefixes as follows:

$$(p \in Pref(q^{(i)}) \cap Pref(q^{(j)})) \implies \begin{array}{c} \left(\mathcal{BM}_{|p|}(F^{(i)}) = \mathcal{BM}_{|p|}(F^{(j)})\right) \\ \wedge \left((f^{(i)})_{1:|p|m} = (f^{(j)})_{1:|p|m}\right), \end{array} \forall q^{(i)}, q^{(j)} \in \mathcal{L}. \quad (3.9)$$

Moreover, we utilize the constrained optimal control approach in [147] to jointly solve for all $(F^{(i)}, f^{(i)})$, which involves the use of Q-parametrization to convert the non-convex problem into a convex one. However, it is well known that Q-parametrization does not generally lead to convex problems when additional structure on the gains $F^{(i)}$ and $f^{(i)}$ are imposed. One of our main results is thus to show that the prefix dependency of the gains in (3.9) still leads to a convex problem when using Q-parametrization. More precisely, in the following theorem, we will present a bijection relationship between the constraints (3.9) on the gain matrices/vectors of prefix-based feedback of the form in (3.7) with the parametrization in (2.2) and (2.3). Hence, the convexity of the corresponding Q-parametrization is preserved when imposing the prefix dependency constraints.

Theorem 1. *Given a prefix-based estimator/controller with the transformed input term* (3.7)*, we associate with it block matrices* $\{(F^{(i)}, f^{(i)})\}_{i=1}^{|\mathcal{L}|}$ *formed from the filter gains, where for all* $q^{(i)} \in \mathcal{L}$*, the* (j,k)*-block entry* $F_{jk}^{(i)}$ *of* $F^{(i)}$ *is defined as*

$$F_{jk}^{(i)} \doteq F_{(t_0+j-1, t_0+k-1, q_{[1:j]}^{(i)})} \quad (3.10)$$

$\forall k \in [1, j]$, $\forall j \in [1, T]$*, and* $F_{jk}^{(i)} = 0$ *otherwise; and the* j*-th block entry of the feedforward term* $f^{(i)}$ *is defined as*

$$f_j^{(i)} \doteq f(t_0 + j - 1, q_{[1:j]}^{(i)}) \quad (3.11)$$

$\forall j \in [1, T]$*. Let* S *and* $\bar{C}^{(i)}$ *be as:*

$$\begin{aligned} \bar{C}^{(i)} &\doteq \begin{bmatrix} diag(q^{(i)}) \otimes C & 0_{pT \times n} \end{bmatrix}, \\ S &\doteq \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B_\xi & 0 & \cdots & 0 \\ AB_\xi & B_\xi & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{T-1}B_\xi & A^{T-2}B_\xi & \cdots & B_\xi \end{bmatrix}. \end{aligned} \quad (3.12)$$

Then, Eqs. (2.2) *and* (2.3) *define a bijection such that any estimator* $\{(F^{(i)}, f^{(i)})\}_{i=1}^{|\mathcal{L}|}$ *is paired*

with one and only one element in the polyhedral set:

$$\mathcal{Q}(\mathcal{L}) \doteq \left\{ \{(Q^{(i)}, r^{(i)})\}_{i=1}^{|\mathcal{L}|} \middle| \begin{array}{l} Q^{(i)} \text{ is block lower diagonal}, \qquad \forall i, \\ (p \in Pref(q^{(i)}) \cap Pref(q^{(j)})) \implies \begin{array}{r} (\mathcal{BM}_{|p|}(Q^{(i)}) = \mathcal{BM}_{|p|}(Q^{(j)})) \\ \wedge \left((r^{(i)})_{1:|p|m} = (r^{(j)})_{1:|p|m}\right), \end{array} \forall q^{(i)}, q^{(j)} \in \mathcal{L} \end{array} \right\}. \tag{3.13}$$

Using the above theorem, a necessary and sufficient condition for the existence of prefix-based estimators and controllers that solve Problem 1 can then be formulated as follows:

Theorem 2 (Estimator and Controller Synthesis with Missing Data (Prefix-Based))**.** *There exists a prefix-based estimator/controller (i.e., $\{(F^{(i)}, u_0^{(i)})\}_{i=1}^{|\mathcal{L}|}$) that satisfies equalized recovery with parameters $(M_1, M_2, \mathcal{L})$ if and only if the following robust linear programming problem is feasible:*

$$\textit{Find} \qquad \left\{(Q^{(i)}, r^{(i)})\right\}_{i=1}^{|\mathcal{L}|} \in \mathcal{Q}(\mathcal{L}) \tag{3.14a}$$

$$\textit{subject to} \quad \begin{array}{l} \forall (\|w\| \le \eta_w, \|v\| \le \eta_v, \|\xi(t_0)\| \le M_1): \\ \quad \|u_\xi + u_d\| \le \eta_u, \|\xi^{(i)}\| \le M_2 \textit{ and } \|\begin{bmatrix} 0_{n \times nT} & I_n \end{bmatrix} \xi^{(i)}\| \le M_1, \end{array} \quad \forall i \in [1, |\mathcal{L}|], \tag{3.14b}$$

where

$$\begin{aligned} \xi^{(i)} &= (H + SQ^{(i)}\bar{C}^{(i)}H)w + SQ^{(i)}N^{(i)}v + (I + SQ^{(i)}\bar{C}^{(i)})(J\xi(t_0) + H\tilde{k}) + Sr, \\ u_\xi &= Q^{(i)}\bar{C}^{(i)}Hw + Q^{(i)}N^{(i)}v + Q^{(i)}\bar{C}^{(i)}(J\xi(t_0) + H\tilde{k}) + r, \end{aligned} \tag{3.15}$$

$$\bar{C}^{(i)} \textit{ and } S \textit{ are defined in } (3.12), \ \mathcal{Q}(\mathcal{L}) \textit{ is as defined in } (3.13), \tag{3.16}$$

$$N^{(i)} = diag(q^{(i)}) \otimes I, \ \tilde{k} = \mathbb{1}_T \otimes k_\xi, \tag{3.17}$$

$$J \doteq \begin{bmatrix} I_n \\ A \\ \vdots \\ A^{T-1} \\ A^T \end{bmatrix}, \ H = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ I_n & 0 & \cdots & 0 \\ A & I_n & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{T-1} & A^{T-2} & \cdots & I_n \end{bmatrix}. \tag{3.18}$$

Remark 4. *The above feasibility problem can be modified to minimize the intermediate level M_2 subject to a given equalized recovery level M_1 and given missing data language $\mathcal{L}$, which can be easily shown to be a robust linear program over the decision variables $\left\{(Q^{(i)}, r^{(i)})\right\}_{i=1}^{|\mathcal{L}|}$ and M_2.*

Since the feasibility problem in (3.14) contains semi-infinite constraints due to the "for all" quantifier on the uncertain terms, the problem is not readily solvable. However, as in [131], techniques from robust optimization and duality [20, 23] can be applied to obtain a linear programming

(LP) problem with only finitely many linear constraints. In particular, we have the following theorem:

Theorem 3 (Robustified Estimator and Controller Synthesis with Missing Data (Prefix-Based))**.** *The feasibility of prefix-based finite horizon affine estimators and controllers that solve Problem 1 is equivalent to the feasibility of the following linear optimization problem:*

$$
\begin{aligned}
&\textit{Find} \quad \left\{(Q^{(i)}, r^{(i)})\right\}_{i=1}^{|\mathcal{L}|} \in \mathcal{Q}(\mathcal{L}),\ \left\{(\Pi_1^{(i)}, \Pi_2^{(i)}, \Pi_3^{(i)})\right\}_{i=1}^{|\mathcal{L}|} \\
&\textit{subject to} \quad \forall i \in [1, |\mathcal{L}|],\ q^{(i)} \in \mathcal{L}, \\
&\qquad \Pi_1^{(i)} \geq 0, \Pi_2^{(i)} \geq 0, \Pi_3^{(i)} \geq 0, \\
&\qquad \Pi_1^{(i)} \begin{bmatrix} \eta_w \mathbb{1} \\ \eta_v \mathbb{1} \\ M_1 \mathbb{1} \end{bmatrix} \leq M_2 \mathbb{1} - \begin{bmatrix} I \\ -I \end{bmatrix} (Sr^{(i)} + (I + SQ^{(i)}\bar{C}^{(i)})H\tilde{k}), \\
&\qquad \Pi_2^{(i)} \begin{bmatrix} \eta_w \mathbb{1} \\ \eta_v \mathbb{1} \\ M_1 \mathbb{1} \end{bmatrix} \leq M_1 \mathbb{1} - \begin{bmatrix} I \\ -I \end{bmatrix} \begin{bmatrix} 0_{n \times nT} & I_n \end{bmatrix} (Sr^{(i)} + (I + SQ^{(i)}\bar{C}^{(i)})H\tilde{k}), \\
&\qquad \Pi_3^{(i)} \begin{bmatrix} \eta_w \mathbb{1} \\ \eta_v \mathbb{1} \\ M_1 \mathbb{1} \end{bmatrix} \leq \eta_u \mathbb{1} - \begin{bmatrix} I \\ -I \end{bmatrix} (r^{(i)} + Q^{(i)}\bar{C}^{(i)}H\tilde{k} + u_d), \\
&\qquad \Pi_1^{(i)} P_\eta = \begin{bmatrix} I \\ -I \end{bmatrix} G^{(i)},\ \Pi_2^{(i)} P_\eta = \begin{bmatrix} I \\ -I \end{bmatrix} \begin{bmatrix} 0_{n \times nT} & I_n \end{bmatrix} G^{(i)},\ \Pi_3^{(i)} P_\eta = \begin{bmatrix} I \\ -I \end{bmatrix} \tilde{G}^{(i)},
\end{aligned}
\tag{3.19}
$$

where $\tilde{k}$, J, H and $N^{(i)}$ are as defined in (3.17) *and* (3.18)*, $\bar{C}^{(i)}$ and S are as defined in* (3.12)*, $\mathcal{Q}(\mathcal{L})$ is as defined in* (3.13)*, and*

$$
G^{(i)} = \begin{bmatrix} (I + SQ^{(i)}\bar{C}^{(i)})H & SQ^{(i)}N^{(i)} & (I + SQ^{(i)}\bar{C}^{(i)})J \end{bmatrix},
$$

$$
\tilde{G}^{(i)} = \begin{bmatrix} Q^{(i)}\bar{C}^{(i)}H & QN^{(i)} & Q^{(i)}\bar{C}^{(i)}J \end{bmatrix},
$$

$$
P_\eta = \begin{bmatrix} I & 0 & 0 \\ -I & 0 & 0 \\ 0 & I & 0 \\ 0 & -I & 0 \\ 0 & 0 & I \\ 0 & 0 & -I \end{bmatrix}.
$$

Moreover, if (3.19) is feasible, then we may invert the mappings in (2.2) and (2.3) to obtain:

$$F^{(i)} \doteq (I + Q^{(i)}\bar{C}^{(i)}S)^{-1}Q^{(i)}, \tag{3.20}$$

$$f^{(i)} \doteq (I + Q^{(i)}\bar{C}^{(i)}S)^{-1}r^{(i)}, \tag{3.21}$$

and we can establish that

1. Each $F^{(i)}$ is block lower triangular;
2. For all $\lambda \in Pref(q^{(i)}) \cap Pref(q^{(j)})$, we have $\mathcal{BM}_{|\lambda|}(F^{(i)}) = \mathcal{BM}_{|\lambda|}(F^{(j)})$ and $f_k^{(i)} = f_k^{(j)}$ for all $k \in [1 : |\lambda|n]$;
3. A prefix-based estimator or controller solving Problem 1 is defined by

$$u_\xi(\lambda) = f(t, \lambda) + \sum_{\tau=t_0}^{t} F_{(t,\tau,\lambda)} y_\xi(\tau), \tag{3.22}$$

where $\lambda \in \bigcup_{q^{(i)} \in \mathcal{L}} Pref(q^{(i)})$, $t \doteq t_0 + |\lambda| - 1$, and the matrices $F_{(t,\tau,\lambda)}$ and $f(t, \lambda)$ are defined according to (3.10) and (3.11).

3.4 Discussions

In this section, we discuss how the proposed finite-horizon estimators and controllers can be implemented. We also highlight the relation between equalized recovery and more familiar notions of detectability and stabilizability.

3.4.1 Implementation Strategies

Assuming that the optimization problems proposed in the previous section have a feasible solution, there are multiple scenarios in which the estimator or controller can be applied. First, if the problem under consideration is one that is finite horizon, we can directly use the obtained gains. Second, if the missing data pattern repeats itself with a period of T time-steps, then the same gains can be used with period T since they guarantee that the recovery period M_1 is reached at the end of the period.

Alternatively, the gains can be used in conjunction with an estimator that guarantees equalized performance or a controller that guarantees forward invariance of M_1 level when there is no missing data. In particular, if we consider languages $\mathcal{L}$ with words that start with a $q(t) = 0$, then

we can switch from the equalized performance estimator/invariance controller to equalized recovery one whenever a missing measurement occurs and revert back to the equalized performance estimator/invariance controller after the recovery time T.

In addition, instead of using hypercubes as set templates due to our use of infinity norms, we could also use more flexible set templates, e.g., zonotopes. The incorporation of such set templates may be done using linear constraints as described in [135].

3.4.2 Relationship to Detectability and Stabilizability

In particular, we state the results in terms of the recent detectability definition for linear systems with data loss events [79]. Similar results also hold true for stabilizability. Consider a system:

$$\begin{aligned} x(t+1) &= Ax(t), \\ y(t) &= \begin{cases} Cx(t), & q(t)=1, \\ \emptyset, & q(t)=0. \end{cases} \end{aligned} \tag{3.23}$$

This is the same as the system model in (3.1) with input and noise terms omitted. We denote a system of the form (3.23) subject to a missing data language $\mathcal{L} \subseteq \mathbb{B}^\omega$ consisting of infinite words (hence the superscript ω) as $(A, C, \mathcal{L})$. Detectability of such a system is defined as follows [79]:

Definition 8 (Missing Data Detectability)**.** *The system $(A, C, \mathcal{L})$ is said to be **detectable** if for any (infinite-length) $q \in \mathcal{L}$ and any initial state $x_0 \in \mathbb{R}^n$ with $y(t, x_0, q) = 0$ for all $t \in \mathbb{N}$, it holds that $x(t, x_0, q) \to 0$ as $t \to \infty$, where $y(t, x_0, q)$ and $x(t, x_0, q)$ are the output and state, respectively, at time t when the initial state is x_0.*

In the following, we show that the existence condition for an equalized recovery estimator is a strict superset of the missing data detectability property defined above. First, we prove that the detectability of a system with missing data implies that an equalized recovery estimator exists and then, we provide an example where an equalized recovery estimator exists even when the system is not detectable. The proofs of these propositions are given in A.4.

Proposition 4. *If a system in the form of* (3.23) *with a missing data language $\mathcal{L}$, is detectable according to Definition 8, then for any recovery level $M_1 > 0$, there exist an intermediate level $M_2 \geq M_1$ and a recovery time $T \in \mathbb{N}$ such that there exists an estimator that achieves equalized recovery for the estimation error with these parameters for the missing data language $\mathcal{L}' = \{q \mid |q| = T,\ q \in Pref(\mathcal{L})\}$.*

Moreover, equalized recovery is slightly more general than detectability as stated next.

Proposition 5. *The set of discrete-time systems in the form* (3.23) *for which an estimator exists whose state estimation error satisfies equalized recovery is a strict superset of all detectable discrete-time systems with missing data.*

3.5 Examples

In this section, three different examples are used to illustrate important properties of the theory presented above. First, the improvement of prefix-based estimators over our previous work's time-based estimation is visualized by revisiting an example from [131]. Second, prefix-based controllers are utilized to discuss the problem of guaranteeing safety of a lane-keeping system when its sensors have missing data events. Finally, the scalability of these methods is shown with a multi-agent tracking problem where a group of followers attempts to follow a leader down a narrow passageway.

3.5.1 Estimator Synthesis

In vehicle safety systems, there are many state estimation problems that must be solved within a finite time horizon (e.g., a vehicle that would like to understand where other vehicles are on the road before merging into a lane or exiting the highway). In these situations, a vehicle's safety (and the safety of its occupants) relies on execution of a maneuver within a time limit, T, because if a vehicle waits too long, it may miss its opportunity to continue towards its destination or the lane may end. One method of guaranteeing safety during such a maneuver is to provide bounds on how "good" the estimates of the other vehicles in the environment are during the time window. If the maneuver can be executed without entering any of the sets defined by our bounded error estimates of the other vehicles, then the vehicle can be guaranteed to be safe.

The Adaptive Cruise Controller is a driver assistance system which controls the acceleration of the user's car (the ego car) with two objectives: (i) to maintain a desired speed, if there is no vehicle in front of it, and (ii) to maintain a safe following distance, if there is a vehicle in front of it. Typically implemented with a radar or computer vision system, the adaptive cruise controller is a hybrid controller, but when considering the estimation error system we may analyze only the following linear system:

$$\begin{aligned} \xi(t+1) &= A\xi(t) + u_e(t) + Ew(t), \qquad w(t) \in \{w \in \mathbb{R}^3 \mid \|w\| \leq \eta_w\}, \\ y_\xi(t) &= \begin{cases} C\xi(t) + v(t), & q(t) = 1, \\ \emptyset, & q(t) = 0, \end{cases} \quad v(t) \in \{v \in \mathbb{R}^2 \mid \|v\| \leq \eta_v\}, \end{aligned} \tag{3.24}$$

where $\xi(t) = [\Delta v_e(t), \Delta h(t), \Delta v_L(t)]^T$ is the estimation error state, consisting of the error in the speed v_e of the ego vehicle, headway h, and speed v_L of the lead vehicle. Each matrix in (3.24) is defined as

$$A = \begin{bmatrix} e^{-\kappa T_s} & 0 & 0 \\ \frac{e^{-\kappa T_s}-1}{\kappa} & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \quad E = \begin{bmatrix} 0 \\ \frac{T_s^2}{2} \; T_s \end{bmatrix},$$

with $\kappa \triangleq \bar{k}_1/m$. Considering the parameter values selected in Table 3.1, one can completely define the system above. Then, when given a missing data language $\mathcal{L}$ and a set of parameters M_1 and M_2, we can pose this in the estimator synthesis form of Problem 1.

We will discuss the results of synthesis when considering the following language:

$$\begin{aligned} \mathcal{L}_1 &= \left\{ q \in \mathbb{B}^6 \;\middle|\; (q_{[1]} = q_{[6]} = 1) \wedge (\exists_1 i \in [2,5] \text{ s.t. } q_{[i]} = 0) \right\} \\ &= \Big\{ \; 101111, \;\; 110111, \;\; 111011, \;\; 111101, \;\; 111111 \; \Big\}, \end{aligned}$$

where the symbol $\exists_1 i$ indicates that there may exist up to one such element i. For this language, the worst-case language $\mathcal{L}_1^*$ can be found to be $\mathcal{L}_1^* = \{100001\}$.

For the language $\mathcal{L}_1$ and the recovery level $M_1 = 1$, the minimal value of the intermediate level M_2 is found using Remark 4 and Theorem 3. In contrast, another solution to the problem can be obtained using time-based feedback, the worst-case language $\mathcal{L}_1^*$, and Theorem 3 via a robust optimization similar to that of (3.14). From the intuition described in Section 3.3, we expect that the prefix-based estimator can in general guarantee tighter estimation error bounds than the time-based estimator. A comparison is shown and discussed in more detail within Figure 3.1 for the Adaptive Cruise Control example. In both cases, the system is initialized with random initial conditions and disturbances that satisfy the specified sets.

The time-based estimator's optimization required 1999 free variables and 0.2489 s to solve when using Gurobi [63], while the prefix-based estimator's optimization required 7993 free variables and 0.3637 s to solve.

Table 3.1: Constants used in the Automatic Cruise Control (ACC) Example.

m	1370 kg	T_s	0.5 s
$\bar{k}_0$	7.58 N	η_w	0.1
$\bar{k}_1$	9.9407 Ns/m	η_v	0.05

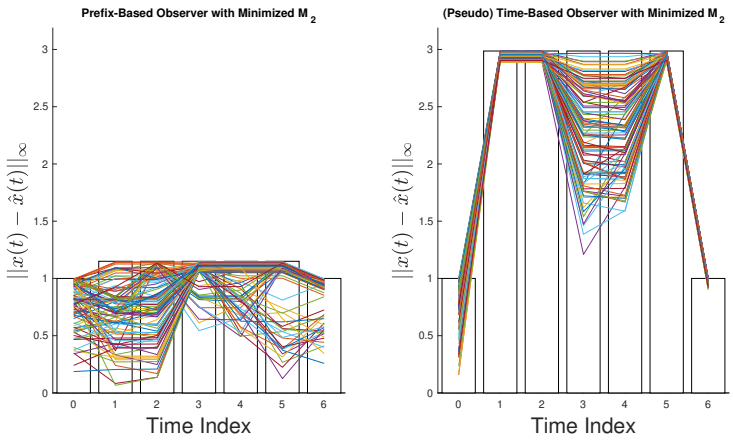


Figure 3.1: Estimation error levels achieved by prefix-based (left) and time-based (right) estimators for the adaptive cruise control system. The minimum M_2 value for which equalized recovery is feasible, with $M_1 = 1$ and $T = 6$, is found by solving the robust linear program for the prefix-based and time-based feedback laws. The optimal M_2 that the prefix-based feedback can guarantee is $M_2 = 1.1498$ while the optimal M_2 that the time-based feedback can provide is $M_2 = 2.9864$.

3.5.2 Controller Synthesis

Consider an automatic lane-keeping system. In such a system, one can imagine that the system has an estimate (with bounded error) of the vehicle's lateral position with respect to the center of the lane. Estimates might come from a computer vision system or other noisy sensors and thus can be subject to glare or misidentification of lane boundaries. This is where missing data events can arise.

In this system, instead of making estimates about where the vehicle is, we will try to control it so that it remains near the center of the lane. The simplified lane keeping model is defined by the following double integrator system:

$$\begin{aligned} \xi(t+1) &= \begin{bmatrix} 0 & 1 \\ 0 & -20 \end{bmatrix} \xi(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} w(t),\ w(t) \in \mathcal{W} \doteq \{w \in \mathbb{R} \mid |w| \leq 0.05\} \\ y_\xi(t) &= \begin{cases} \xi(t) + v(t), & q(t) = 1, \\ \emptyset, & q(t) = 0, \end{cases} \qquad v(t) \in \mathcal{V} \doteq \{v \in \mathbb{R}^2 \mid \|v\| \leq 0.1\}, \end{aligned} \tag{3.25}$$

where the state $\xi(t) = [x_c(t), \dot{x}_c(t)]^T$ consists of the deviation $x_c(t)$ from the centerline of the lane and the lateral velocity $\dot{x}_c(t)$, and the control input $u(t)$ is the lateral force applied through steering.

Again, the trajectory of $q(t)$ is defined by a missing data language $\mathcal{L}$ which would come from the properties of the roads that the autonomous vehicle operates on as well as the specifications of its sensors. With all of the above information, the problem of guaranteeing safety can be posed as

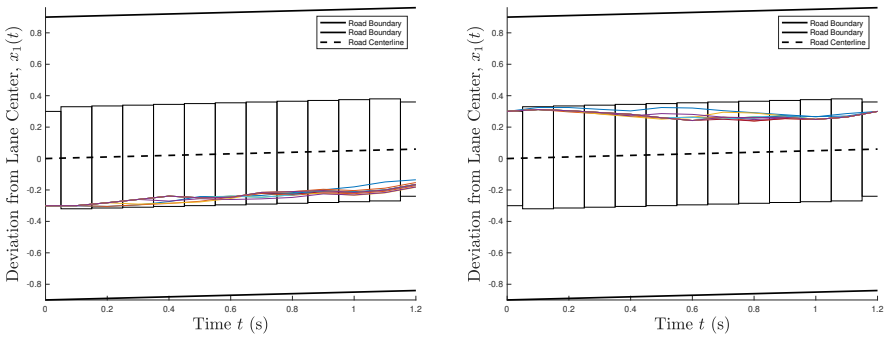


Figure 3.2: Consider any one of the panels above. In each panel, multiple trajectories of the lane keeping system are visualized, where each trajectory is initialized at the same state on the $M_1 = 0.3$ hypercube's boundary and experiences the exact same disturbance (a carefully chosen, maximum norm disturbance). The only thing that varies across each trajectory is the missing data pattern σ. Thus, what causes the trajectories to diverge is how the prefix-based controller handles these missing data events when they happen. Regardless of the missing data pattern, it is shown that these adversarily chosen initial conditions and disturbances can still be guaranteed to return to the desired level M_1 and the system achieves equalized performance.

follows: Given that the vehicle state starts near the center of the lane with near zero lateral velocity, can we design a prefix-based feedback controller such that the vehicle never deviates outside of the lane boundaries despite missing data events from $\mathcal{L}$?

The formal interpretation of such a problem would be that given some specification of the initial position (i.e., our M_1 value), the lane width, along with the disturbance sets $\mathcal{W}$ and $\mathcal{V}$ and the missing data model $\mathcal{L}$, find a feedback law (3.7) such that the decision variable M_2 is minimized. If the minimal M_2 is below the lane width value, we can guarantee that the system will not deviate from the center of the lane when using our controller during a missing data event from $\mathcal{L}$.

Consider the system in (3.25), with the following missing data language:

$$\mathcal{L}_2 = \left\{ q \in \mathbb{B}^{12} \;\middle|\; \exists_1 i \in [1, 10] \text{ s.t. } \begin{array}{l} (q_{[i]} = 0) \wedge \\ (q_{[i+1]} = 0) \wedge \\ ((j \neq i) \wedge (j \neq i+1) \implies q_{[j]} = 1) \end{array} \right\}.$$

Let the initial state of the lane keeping system be within an infinity norm ball of radius $M_1 = 0.3$. Given that the process disturbances come from the set $\mathcal{W} = \{w \in \mathbb{R} \mid |w| \leq 0.05\}$ and the measurement disturbances come from the set $\mathcal{V} = \{v \in \mathbb{R}^2 \mid \|v\|_\infty \leq 0.1\}$, we synthesize a controller that minimizes the worst case deviation from the center of the lane. Note that the disturbance set $\mathcal{V}$ is overly conservative (most sensors can detect the lane boundaries of a lane to a precision of 0.01 m), but this is meant to simply show one of the many possible settings that the controller synthesis framework can handle.

In Figure 3.2, we illustrate how some of the trickier initial conditions are handled. The figure contains multiple trajectories. Each trajectory begins with the same state on the boundary of the M_1 norm ball and experiences identical disturbances from the sets $\mathcal{W}$ and $\mathcal{V}$. The aspect that changes between each trajectory is the missing data pattern (which leads to different prefix-based feedback). One can see that for some of the hardest missing data patterns, the deviation from the center of the lane gets very close to the edge of our guarantee set (i.e., the boxes of width M_2), but always recovers to the proper level in the end.

This synthesis problem contained 46,773 free variables and was solved after 2.5420 s with Gurobi [63].

3.5.3 Controller Synthesis: Formation Control

Finally, we consider the problem of coordinating the movement of a fleet of agents through an obstacle filled environment. Precisely controlling formations of controlled agents has become very important across multiple domains including space exploration and disaster relief. For example, when using microsatellites to obtain many spacially distributed observations during orbit, maintenance of specific formations allows the devices to spend less fuel while achieving their mission [153]. In these contexts and many others, carefully controlled formations of the fleet make it easier to achieve a task and decrease the probability of collision with obstacles (e.g., space debris, other satellites).

Unfortunately, while navigating through obstacle-filled fields, localization methods that are based on a relative navigation sensing system or computer vision algorithms may experience missing measurements. In the following example, we show how a prefix-based control strategy for formations of such agents can be used to guarantee safe navigation of a narrow channel while maintaining a formation close to the desired one.

Consider the problem of designing a controller for a set of $n_r \cdot n_c$ agents with single-integrator dynamics moving in a two-dimensional plane. The agents seek to align themselves in n_r rows and n_c columns behind an uncontrolled lead agent (forming a grid). The rows are defined such that there are n_c agents in each row and the space between agents in a given row is always 2 m. Furthermore, the rows are organized in the y-direction such that they are evenly distributed between $\ell_y + 1$ and $\ell_y - 1$ where ℓ_y is the y-component of the leader's position.

The lead agent is moving with unknown, but bounded actions within the set $\mathcal{W} = \{w \in \mathbb{R}^2 \mid \|w\| \leq 1.5\}$. The following agents' movement can be defined in terms of error states in the x and y directions, $e_x^{(i,j)}(t) \doteq x^{(i,j)}(t) - \bar{x}^{(i,j)}(t)$ and $e_y^{(i,j)}(t) \doteq y^{(i,j)}(t) - \bar{y}^{(i,j)}(t)$, representing the difference in the (i,j)-th follower's x- and y- positions from its desired formation (i.e., grid) position.

In this work, the formation is maintained by every first agent in the row (any agent with $j = 1$ observing/sensing the states of the leader) and all other agents observing/sensing the states of the agent before it (agent j measures the position of agent $j - 1$). When the desired states are known to be a constant offset from the leader's position or positions of other agents, the system may be written as follows:

$$e_x^{(i,j)}(t+1) = \begin{cases} (u_x^{(i,j)}(t) - w_x(t)) \cdot \Delta t, & j = 1, \\ (u_x^{(i,j)}(t) - u_x^{(i,j-1)}(t)) \cdot \Delta t, & \text{otherwise}, \end{cases}$$

$$e_y^{(i,j)}(t+1) = \begin{cases} (u_y^{(i,j)}(t) - w_y(t)) \cdot \Delta t, & j = 1, \\ (u_y^{(i,j)}(t) - u_y^{(i,j-1)}(t)) \cdot \Delta t, & \text{otherwise}, \end{cases}$$

$$y_{e,i}(t) = \begin{cases} \begin{bmatrix} e_x^{(i,j)}(t) \\ e_y^{(i,j)}(t) \end{bmatrix} + v_i(t), & q(t) = 1, \\ \emptyset, & q(t) = 0, \end{cases}$$

where the discretization step Δt is 0.1 s, the measurement disturbances $v_i(t)$ come from the set $\mathcal{V} \doteq \{v \in \mathbb{R}^2 \mid \|v\| \leq 0.5\}$ for all i and the input of the (i, j)-th agent as instantaneous speed in the x or y direction is written as $u_x^{(i,j)}(t)$ and $u_y^{(i,j)}(t)$, respectively.

In Figures 3.3 and 3.4, the followers (black drones) are behind the leader (maize and blue drone) as the leader moves from left to right through a narrow passage, where localization information is sometimes dropped according to the language $\mathcal{L}_3$:

$$\mathcal{L}_3 = \left\{ \begin{matrix} 11111111,\ 01111111, \\ 00111111,\ 10111111, \\ 01011111,\ 11011111 \end{matrix} \right\}.$$

The worst-case language in this case is $\mathcal{L}_3^* = \{00011111\}$. A time-based controller using $\mathcal{L}_3^*$ in this channel would have to adopt an open-loop strategy for the first three time steps and could not guarantee that the followers would avoid collision (see Figure 3.3), but the prefix-based controller maintains the state error within a safe bound throughout the channel (see Figure 3.4).

The 2×2 version of this problem contained 1519 free variables and was solved in 10.9708 s with Gurobi [63]. To illustrate the scalability of our control synthesis method, the solution time for this problem is compared in Table 4.2, where the number of following agents is varied. These simulations show that our proposed approach does involve longer computation times when the number of states are increased. Note, however, that our control synthesis is done offline and the computation time is thus not a critical limiting factor.

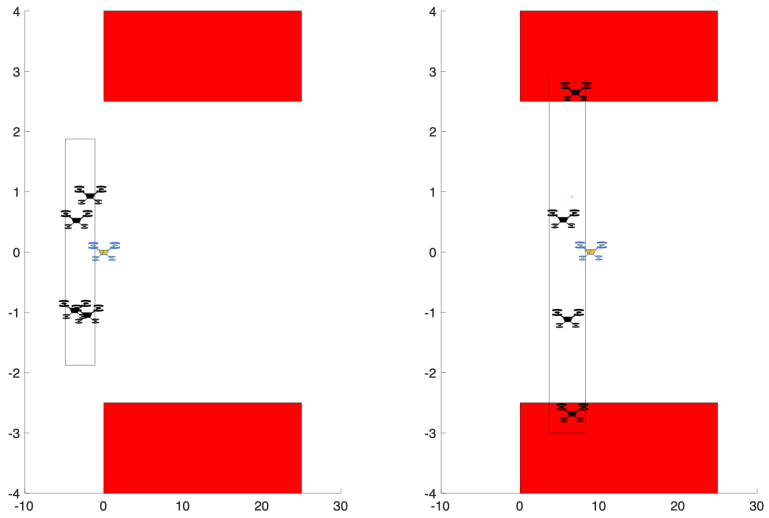


Figure 3.3: A time-based controller using the worst-case language could not guarantee that the followers would safely exit the channel. It can guarantee that followers (black drones) will remain in the black outline defined by $M_2 = 2$ which overlaps with the red wall (thus collisions may happen)

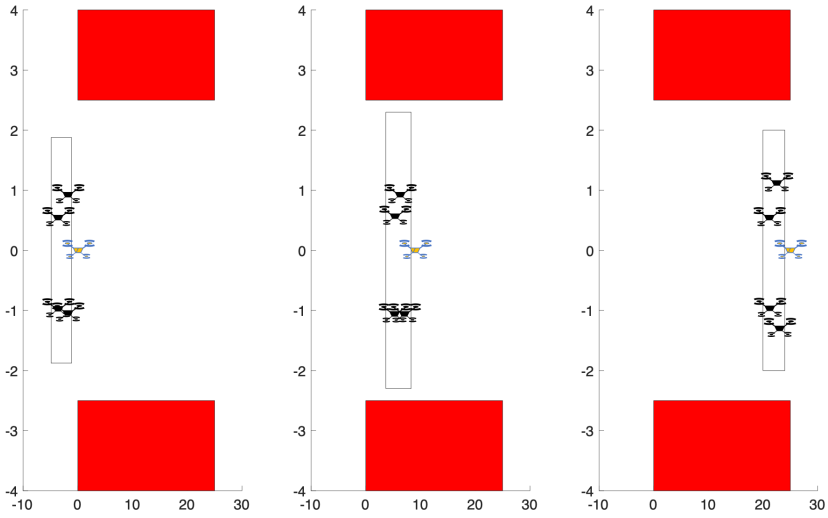


Figure 3.4: While in the channel, the system experiences missing data events according to language $\mathcal{L}_3$, but a prefix-based controller can guarantee that the followers (black drones) will travel through the channel without colliding with either wall ($M_2 = 1.3$ and the black outline does not ever touch the red wall)

Table 3.2: Analysis of the control synthesis time when the number of controlled agents increases.

Number of Followers	Average Solver Time (s)
4 (2×2 grid formation)	7.55
9 (3×3 grid formation)	51.60
16 (4×4 grid formation)	122.56

CHAPTER 4

Synthesis of Finite-Horizon Adaptive Controllers for Hybrid Systems Using Bilinear Optimization

4.1 Introduction

Recall that one of the real-world limitations of CPSs is that their dynamics model may contain some parameters which the decision-maker is initially uncertain about. The parameters are uncertain when a CPS interacts with a large variety of objects or people on a regular basis (e.g., a robotic arm that must organize a large variety of toys in a room). We model this variety with different, but a priori known models in this chapter. For this set of known models or modes, we develop an estimator that can produce set-valued estimates of the true model for the system when given a history of state measurements and input data. This estimator is then incorporated into an adaptive controller and we describe how an adaptive controller with this structure can be designed to satisfy a guaranteed reachability task using bilinear optimization.

4.2 Problem Formulation

In this chapter, we consider the linear system with unknown parameters from Section 2.4.2.2. Recall that this is of the form:

$$x(t+1) \quad = A_\theta x(t) + B_\theta u(t) + w(t) \quad w(t) \in \mathcal{W}_\theta \tag{4.1}$$

where θ is the mode of the system taking values in a finite set Θ of models (i.e. $\theta \in \Theta$), x_t is the state of the system taking values in $\mathcal{X}$, u_t is the input at time t that must satisfy input constraints represented by polytope $\mathcal{U}$ (i.e. $u_t \in \mathcal{U}$), w_t is the unmeasured disturbance to the system that lies in the polytope $\mathcal{W}_\theta$ ($w_t \in \mathcal{W}_\theta$). The system starts with a fixed, yet unknown, mode θ. It is assumed that the set Θ (including the matrices A_θ, B_θ and polytopes $\mathcal{W}_\theta$ for each mode) and the polytope $\mathcal{U}$ are known. We are interested in the following problem.

Problem 2. *Given a system of the form* (4.1)*, an initial condition* $x(0)$*, a time horizon* T*, and a polytopic target set* $\mathcal{X}_T$*, find a controller* $\gamma : \mathcal{X} \times (\mathcal{X} \times \mathcal{U})^* \to \mathcal{U}$ *such that, no matter what* $\theta \in \Theta$ *is the active model, the state of the closed loop system reaches the target set* $\mathcal{X}_T$ *at time* T *for any possible disturbances in* $\mathcal{W}_\theta$*.*

Remark 5. *Without much modification, the assumption that the initial state* x_0 *is known can be replaced with the initial state belonging to a set* $\mathcal{X}_0$ *(i.e.* $\mathcal{X}_0$*) and the results of this chapter will still hold. The rest of this chapter utilizes the assumption that the initial state is known to avoid over-complicating the exposition.*

If one can find a controller that can robustly achieve the reachability task, agnostic of σ, this would solve the Problem 2. However, as we demonstrate with the system in the next example, such robust controllers might not always exist.

Example 1. *Consider the following two mode system (i.e.* $|\Theta| = 2$*), where* $(A_1, B_1, \mathcal{W}_1) = (1, 1, [0.1, 0.3])$ *and* $(A_2, B_2, \mathcal{W}_2) = (1, 1, [-0.3, -0.1])$*. Note that the only difference between these one-dimensional systems is the disturbance set which has the same width in both cases, but is centered on different values. Let the target set be* $\mathcal{X}_T = [-0.1, 0.1]$*, with* $T = 2$*, input set be* $\mathcal{U} = [-0.5, 0.5]$*, and the initial state* $x(0) = 0$*. Then, a control strategy of the form* $u(0) = 0$ *and* $u(1) = \begin{cases} -x(1) - 0.2 & \text{if } x(1) > 0 \\ -x(1) + 0.2 & \text{if } x(1) < 0 \end{cases}$ *will solve the problem. This strategy essentially identifies the mode in the first step and applies an appropriate controller depending on the mode. On the other hand, there is no controller that can robustly achieve the control objective in a way that is agnostic to the mode since such a controller needs to be robust to all disturbances in the set* $[-0.3, 0.3]$*, which is the convex hull of the union of the disturbance sets.*

Controllers such as the one described in Example 1 can satisfy the task by passively observing the systems trajectory (i.e. applying zero input) for a few time indices and then executing a strategy for a single mode. We know, however, that not all systems can be treated that way. The system identification and mode discrimination literature is rife with examples such as the following, where a sequence of inputs need to be carefully chosen in order to discriminate the true mode of the system.

Example 2. *Consider the following two mode system (i.e.* $|\Theta| = 2$*), where* $(A_1, B_1, \mathcal{W}_1) = (1, 1, [-0.5, 0.5])$ *and* $(A_2, B_2, \mathcal{W}_2) = (-1, 1, [-0.5, 0.5])$*. The only difference between the two systems is the sign of the* A *matrix. Let the target set be* $\mathcal{X}_T = [-1, 1]$ *with* $T = 4$*, the input set* $\mathcal{U} = [-5, 5]$ *and the initial state* $x(0) = 0.25$*. The initial state satisfies* $\|x(0)\| \leq 0.25$*, which implies that the system can not be identified at time* $t = 1$ *regardless of the input. The initial disturbance* $w(0)$ *can always be chosen such that* $w(0) = -x(0)$ *which makes mode discrimination*

impossible. But by choosing a large enough input at time $t = 0$, the controller can place the system at a state $x(1)$ from which it is possible to identify the system at time $t = 2$.

In the above example, the mode discrimination approach will be able to discriminate the true mode of the system using a non-arbitrary sequence of inputs and can then achieve the task. However if the goal was to stay close to the origin at time $T = 2$, the discriminating strategy might render the reachability objective impossible. There is yet another class of models Θ where mode discrimination itself is impossible. For such systems, an adaptive controller should not be designed with the goal of guaranteed mode discrimination, but should instead do some combination of opportunistic discrimination and robust control to solve Problem 2. The following example is such a collection.

Example 3. *Consider the following two mode system with $(A_1, B_1, \mathcal{W}_1) = (1, 1, [-1, 0.5])$ and $(A_2, B_2, \mathcal{W}_2) = (1, 1, [-0.5, 1])$. Note that the only difference between these one-dimensional systems is the disturbance set which has the same width in both cases, but is centered on different values. Let the initial state be $x(0) = 2$. Both disturbance sets share some common values $[-0.5, 0.5]$ and thus it is impossible to guarantee that one can discriminate between the two of them. However if the target set to be reached is wide enough (e.g., $\mathcal{X}_T = [-1, 1]$) and there is sufficient input authority (e.g., $\mathcal{U} = [2, 2]$), a simple linear controller $\gamma(x) = -x$ achieves the task robustly for any $T \geq 1$.*

As we see from these examples, neither being agnostic to the mode, nor trying to identify the model first might be the best strategy, depending on the problem in hand. Motivated by this fact, we want to design an adaptive controller that can adjust its gain based on an estimate of the mode as needed subject to a given exploration-exploitation profile. Rather than searching for arbitrary adaptive controllers, we will search through a set of adaptive controllers that have an estimator and a controller with linear gains that are adapted based on the output of the estimator as pictured in Figure 4.1. Thus, we will obtain a sufficient condition to Problem 2. We note that similar estimation based control structures have been used for safe control of hybrid systems [159] but not in the context of adaptive control.

4.3 Estimator Structure

In this section, the first part of the proposed adaptive controller in Figure 4.1, the mode estimator μ, is introduced. The mode estimator is a set valued function. We also define two sets that relate the estimator's output over time (an estimation sequence) to state-input trajectories that can generate similar estimates.

To help with the definition of the mode estimator, consider the following reachable set:

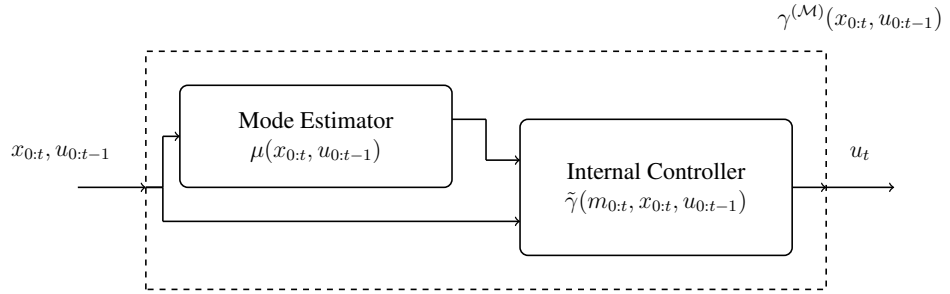


Figure 4.1: The structure of a consistent belief controller γ which, at each time, receives the external behavior $(x(0:t), u(0:t-1))$ and produces the control input $u(t)$.

Definition 9. *The reachable behavior set of mode θ at time t is*

$$\mathcal{R}(\theta, t) = \left\{ (x(0:t), u(0:t-1)) \;\middle|\; \begin{array}{l} x(0) \in \mathcal{X}_0, \\ u(0:t-1) \in \mathcal{U}^t, \\ \forall k \in [0, t-1], \\ \quad x(k+1) - A_\theta x(k) - B_\theta u(k) \in \mathcal{W}_\theta \end{array} \right\} \tag{4.2}$$

The reachable behavior set is the set of all state-input trajectories that can be produced by the mode σ.

We use a simple set-membership mode estimator $\mu : \mathcal{X} \times (\mathcal{X} \times \mathcal{U})^* \to 2^\Theta$ defined as:

$$\mu(x(0:t), u(0:t-1)) \triangleq \{\theta \in \Theta \mid (x(0:t), u(0:t-1)) \in \mathcal{R}(\theta, t)\} \tag{4.3}$$

In words, the mode estimate $\mu(x(0:t), u(0:t-1))$ at time t is a set containing all modes that could produce the observed behavior $(x(0:t), u(0:t-1))$. We also say that $\mu(x(0:t), u(0:t-1))$ at time t contains all modes that are *consistent* with the observed behavior up to time t. Note that, by this definition, the first estimate is always $\mu(x(0)) = \Theta$. To simplify notation, we denote the output of the mode estimator at time t with $\mu(t)$ (i.e. $\mu(t) \triangleq \mu(x(0:t), u(0:t-1))$) and a sequence of estimates with $\mu(0:t)$ as is done for the state and input.

Remark 6. *The estimator* (4.3) *can be naively implemented by, at each time and for each mode, checking if state-input data is contained in an appropriately constructed polytope (one per mode) given in* (4.3)*. Since the length of the state-input data grows with time, so does the dimension of polytopes in* (4.3) *for the naive implementation. Yet, it is also possible to implement an equivalent estimator recursively by using the estimate $\mu(t-1)$ at time t in a way to avoid this growth:*

$$\mu(t) \triangleq \{\theta \in \mu(t-1) \mid x(t) - A_\theta x(t-1) - B_\theta u(t-1) \in \mathcal{W}_\theta\}. \tag{4.4}$$

We will be interested in the set of all state-input trajectories $(x(0:t), u(0:t-1))$ that would

produce a given estimate $\mu(0:t)$, which motivates the next definition.

Definition 10 (Consistency Set)**.** *Consider a vector* $\mathbf{m} \in (2^{\Theta})^t$*. The consistency set* $\mathcal{C}(\mathbf{m})$ *is the set of all state-input trajectories* $(x(0:t), u(0:t-1))$ *that lead to the estimate* $\mathbf{m}$*. That is:*

$$\mathcal{C}(\mathbf{m}) = \{(x(0:t), u(0:t-1)) \mid \mathbf{m} = \boldsymbol{\mu}(x(0:t), u(0:t-1))\}.$$

While consistency sets are useful to characterize the behavior of the proposed adaptive controller in the closed-loop, they are generally non-convex and hard to manipulate. However, we can relate them to a convex set using reachable behavior sets.

First, we overload the reachable behavior set notation to define reachable behavior set for mode estimates $\tilde{\Theta} \subseteq \Theta$:

$$\mathcal{R}(\tilde{\Theta}, t) = \{(x(0:t), u(0:t-1)) \mid \forall \theta \in \tilde{\Theta} : \ (x(0:t), u(0:t-1)) \in \mathcal{R}(\theta, t)\}$$

and for sequences of subsets of modes $\mathbf{m} \in (2^{\Theta})^t$:

$$\mathcal{R}(\mathbf{m}) = \left\{ (x(0:k), u(0:k-1)) \;\middle|\; \begin{array}{l} \forall k \in [0,t] : \\ \quad (x(0:k), u(0:k-1)) \in \mathcal{R}(\mathbf{m}_k, k) \end{array} \right\}. \tag{4.5}$$

This set, denoted as $\mathcal{R}(\mathbf{m})$ when t is either clear from context or irrelevant, provides a number of helpful properties in the analysis and optimization-based control synthesis approach. First and foremost, $\mathcal{R}(\mathbf{m})$ is a polytope for any choice of $\mathbf{m}$.

Lemma 4. $\mathcal{R}(\mathbf{m})$ *is a polytope for any choice of* $\mathbf{m} \in (2^{\Theta})^T$ *with some finite length* T*.*

Proof. First, consider the set $\mathcal{R}(\theta, t)$ for any choice of $\theta \in \mathbf{m}_{t+1}$ and $0 \leq t \leq T-1$. Because $\mathcal{X}_0$ and $\mathcal{U}$ are polytopes, the first two constraints in (4.2) are polytopic constraints on the state-input trajectory $(x(0:t), u(0:t-1))$. The last two lines of (4.2) form a finite number of polytopic constraints (containment by $\mathcal{W}_\theta$) on a linear function of the state-input trajectory. Therefore, $\mathcal{R}(\theta, t)$ is defined entirely by an intersection of finitely many polytopes (i.e. it is a polytope). Next, observe that $\mathcal{R}(\mathbf{m}_{t+1}, t) = \cap_{\theta \in \mathbf{m}_{t+1}} \mathcal{R}(\theta, t)$ and $\mathcal{R}(\mathbf{m}) = \cap_{t \in [0, T-1]} \mathcal{R}(\mathbf{m}_{t+1}, t)$, which is a finite intersection of polytopes by finiteness of T and Σ. Therefore, $\mathcal{R}(\mathbf{m})$ is a polytope. □

We define a partial order on estimation sequences as follows: $\mathbf{m} \subseteq \mathbf{m}'$ if and only if $\mathbf{m}_{k+1} \subseteq \mathbf{m}'_{k+1}$ for all $k \in [0, t-1]$. Then, the consistency set corresponding to an estimation sequence $\mathbf{m}$ contains all the state-input trajectories that exactly leads to the estimates in $\mathbf{m}$, whereas the reachable behavior sets include the state-input trajectories that can lead to the estimates in $\mathbf{m}$ but also those that can lead to $\mathbf{m}'$ for $\mathbf{m} \subset \mathbf{m}'$. The relation of these two sets is illustrated in Figure 4.2 and formalized next.

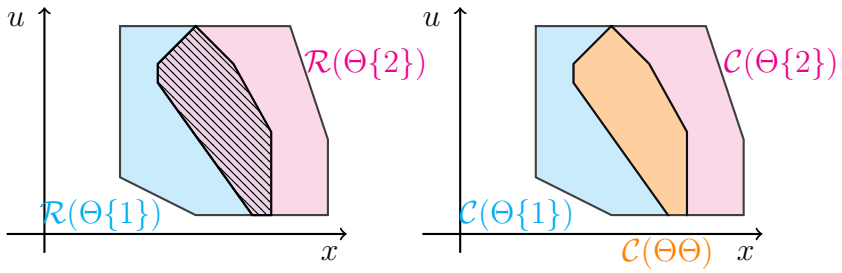


Figure 4.2: An illustration of reachable behavior sets (left) and consistency sets (right) for a system with $\Theta = \{1, 2\}$. We omit $x(0)$ dimension as it is a singleton so the sets are shown in $\mathcal{X} \times \mathcal{U}$ space. Take, for instance, the length-two estimation sequence $\mathbf{m} = \Theta\{2\}$. The set $\mathcal{C}(\Theta\{2\})$ contains all state-input pairs that will lead to the estimate $\{2\}$ but not to the estimate $\{1\}$ at time $t = 1$, whereas the set $\mathcal{R}(\Theta\{2\})$ contains all state-input pairs that would lead to an estimate $\mu(1)$ such that $\{2\} \subseteq \mu(1)$. Reachable behavior sets are always convex, but can be overlapping (left). Consistency sets, however, can be non-convex and collectively partition the space of possible reachable behaviors (right).

Lemma 5. *For a given sequence $\mathbf{m} \in (2^{\Theta})^t$, the consistency set and the reachable behavior set are related by the following equality:*

$$\mathcal{C}(\mathbf{m}) = \mathcal{R}(\mathbf{m}) \setminus \left(\bigcup_{\mathbf{m}' : \mathbf{m} \subset \mathbf{m}'} \mathcal{R}(\mathbf{m}') \right) \tag{4.6}$$

Proof. To prove this, first note that, by utilizing (4.3) and (4.5), the reachable behavior set $\mathcal{R}(\mathbf{m})$ is alternatively written as:

$$\mathcal{R}(\mathbf{m}) = \{(x(0:t), u(0:t-1)) \mid \mathbf{m} \subseteq \mu(0:t)\}.$$

Thus the result follows. □

4.4 Internal Controller Structure

In this section, the second part of the adaptive controller in Figure 4.1 is defined. It is the inner controller and receives as input both the state-input trajectory as well as the estimation sequence from the mode estimator. We define the inner controller as a partial function:

$$\tilde{\gamma} : \mathcal{M} \times (\mathcal{X} \times (\mathcal{X} \times \mathcal{U})^*) \to \mathcal{U}, \tag{4.7}$$

where $\mathcal{M}$ denotes a set of estimation sequences that we call an *exploration-exploitation profile*. Due to the structure of the estimator, not all estimation sequences are feasible. One main restriction

is that the estimator is always "narrowing down" the hypotheses for the true system and will never "add in" a new guess to its list. Hence, the largest exploration-exploitation profile that does not contain unnecessary estimation sequences that cannot occur for any system with any controller, is given as follows:

$$\mathcal{M}^* \triangleq \{\mathbf{m} \in (2^{\Theta})^T \mid \mathbf{m}_{0+1} = \Theta \wedge \forall k \in [1, T-1], \mathbf{m}_{k+1} \subseteq \mathbf{m}_k\}.$$

Given an exploration-exploitation profile $\mathcal{M} \subseteq \mathcal{M}^*$, the actual domain of $\tilde{\gamma}$ is a subset of the set $\mathcal{M} \times (\mathcal{X} \times (\mathcal{X} \times \mathcal{U})^*)$. This is because the arguments of the function are coupled due to the feedback loop. In particular, the inputs $\mathbf{m} \in \mathcal{M}$ and state-input data $(x(0:t), u(0:t-1))$ must satisfy $(x(0:t), u(0:t-1)) \in \mathcal{C}(\mathbf{m})$.

The first input of the inner controller has only a finite number of values. This fact allows for the partial function $\tilde{\gamma}$ to be analyzed as $|\mathcal{M}|$ individual partial functions, each of which is defined with respect to a single $\mathbf{m} \in \mathcal{M}$ and denoted by $\tilde{\gamma}|_{\mathbf{m}} : \mathcal{X} \times (\mathcal{X} \times \mathcal{U})^* \to \mathcal{U}$. Overall, the inner controller has two coupled design parameters: the exploration-exploitation profile $\mathcal{M}$ and the set of controllers $\{\tilde{\gamma}|_{\mathbf{m}}\}_{\mathbf{m}\in\mathcal{M}}$. With this in mind, we will first discuss closed-loop consistency sets and a sufficient condition for a controller to provide a solution to Problem 2. Then, we will provide a parameterization for $\{\tilde{\gamma}|_{\mathbf{m}}\}_{\mathbf{m}\in\mathcal{M}}$ that allows for searching controllers within an optimization framework.

Again, we introduce a slight abuse of notation $\tilde{\gamma}|_{\mathbf{m}}(t) \triangleq \tilde{\gamma}|_{\mathbf{m}}(x(0:t), u(0:t-1))$.

4.4.1 Closed-Loop Trajectories and Robust Reachability

The largest possible exploration-exploitation profile $\mathcal{M}^*$ is used in a controller of the form (4.7). However, the design of this controller $\gamma^{(\mathcal{M}^*)}$ shapes the state-input trajectories $(x(0:t), u(0:t-1))$ that can be observed of the system. We will call this new set of state-input trajectories, constrained by the controller, the *closed-loop consistency set*. We then show that for a controller to be a solution to Problem 2, the closed-loop consistency set it induces should lead to behaviors that reach the target set $\mathcal{X}_T$.

Definition 11. *[Closed-Loop Consistency Set] The closed-loop consistency set $\mathcal{C}(\mathbf{m}', \tilde{\gamma}|_{\mathbf{m}})$ for the estimation sequence $\mathbf{m}' \in (2^{\Theta})^T$ with the controller associated with $\mathbf{m} \in \mathcal{M}$ is the set of all elements of $\mathbf{m}'$'s consistency set that are reached with the disturbance controller $\tilde{\gamma}|_{\mathbf{m}}$. In math, the closed-loop consistency set is:*

$$\mathcal{C}(\mathbf{m}', \tilde{\gamma}|_{\mathbf{m}}) = \left\{ \begin{bmatrix} x(0:t) \\ u(0:t-1) \end{bmatrix} \in \mathcal{C}(\mathbf{m}') \;\middle|\; \begin{array}{l} \forall k \in [0, t-1]: \\ \quad u(k) = \tilde{\gamma}|_{\mathbf{m}}(x(0:k), u(0:k-1)) \end{array} \right\}.$$

The closed-loop reachable behavior set can be defined similarly:

$$\mathcal{R}(\mathbf{m}', \tilde{\gamma}|_{\mathbf{m}}) \triangleq \left\{ \begin{bmatrix} x(0:t) \\ u(0:t-1) \end{bmatrix} \in \mathcal{R}(\mathbf{m}') \;\middle|\; \begin{array}{l} \forall k \in [0, t-1]: \\ \quad u(k) = \tilde{\gamma}|_{\mathbf{m}}(x(0:k), u(0:k-1)) \end{array} \right\},$$

where $\mathbf{m}, \mathbf{m}' \in \mathcal{M}^*$. It follows that $\mathcal{C}(\mathbf{m}', \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{R}(\mathbf{m}', \tilde{\gamma}|_{\mathbf{m}})$ since $\mathcal{C}(\mathbf{m}') \subseteq \mathcal{R}(\mathbf{m}')$ by Lemma 5.

In the example shown in Figure 4.2, the closed-loop consistency set $\mathcal{C}(\Theta\Theta, \gamma|_{\Theta\Theta})$ is the restriction of the orange set to the manifold defined by $u(k) = \tilde{\gamma}|_{\Theta\Theta}(x(0:k), u(0:k-1))$ for all $k \in [0, t]$. This might be a convex or non-convex set depending on the function $\tilde{\gamma}|_{\Sigma\Sigma}$.

Now, we introduce the reachability condition that the controller must satisfy in order to solve Problem 2. The condition can be written in terms of a reachability constraint for each sequence $\mathbf{m} \in \mathcal{M}^*$.

Proposition 6. *Consider a system of the form* (4.1) *and an adaptive controller* $\gamma^{(\mathcal{M}^*)}$ *defined as in Figure 4.1 with mode estimator* μ *given in* (4.3) *and inner controller* $\tilde{\gamma}$ *given in* (4.7)*. If for all* $\mathbf{m} \in \mathcal{M}^*$ *it is true that*

$$\begin{array}{l} \forall (x(0:T-1), u(0:T-2)) \in \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}): \\ \quad \forall \theta \in \mathbf{m}_T, \forall w_{T-1} \in \mathcal{W}_\theta: \\ \qquad (A_\theta x(T-1) + B_\theta \tilde{\gamma}|_{\mathbf{m}}(T-1) + w(T-1)) \in \mathcal{X}_T, \end{array} \tag{4.8}$$

then all closed-loop trajectories of the system's state reach the target set at time T *(i.e.* $x(T) \in \mathcal{X}_T$ *with* $\gamma^{(\mathcal{M}^*)}$*).*

Proof. Consider an arbitrary closed-loop state-input trajectory $(x(0:T), u(0:T-1))$ for the fixed controller $\gamma^{(\mathcal{M}^*)}$ and the corresponding estimation sequence $\mathbf{m}^* = \mu(0:T-1) \in \mathcal{M}^*$. By definition of consistency sets, we have $(x(0:T-1), u(0:T-2)) \in \mathcal{C}(\mathbf{m}^*)$ and, by Lemma 5, $(x(0:T-1), u(0:T-2)) \in \mathcal{R}(\mathbf{m}^*)$. Moreover, since the inner controller $\tilde{\gamma}|_{\mathbf{m}^*}$ is used, we have $(x(0:T-1), u(0:T-2)) \in \mathcal{R}(\mathbf{m}^*, \tilde{\gamma}|_{\mathbf{m}^*})$. With this in mind, if the condition (4.8) holds for all $\mathbf{m} \in \mathcal{M}^*$, it also holds for $\mathbf{m}^*$. Hence, by the last line of (4.8), the control input $u(T-1) = \gamma^{(\mathcal{M}^*)}(T-1)$ guarantees that the final state reaches the target (i.e., $x(T) \in \mathcal{X}_T$). □

Finding arbitrary functions that satisfy Proposition 6 is a fairly difficult task. To simplify the search process, we constrain the class of the controllers considered to those with a certain affine parameterization.

4.4.2 Disturbance Feedback Parameterization

In this section, we restrict the class of controllers so that Proposition 6 can be verified via optimization. In particular, for each $\mathbf{m}$, we restrict the controller $\tilde{\gamma}|_{\mathbf{m}}$ to be an affine disturbance feedback controller with memory.

A disturbance feedback controller takes the form

$$\tilde{\gamma}|_{\mathbf{m}}(x(0:t), u(0:t-1)) = K^{(\mathbf{m}_{[1:t+1]})}\hat{w}(0:t-1) + k^{(\mathbf{m}_{[1:t+1]})}, \tag{4.9}$$

where the disturbance at all times before the current time step are recovered using the state-input trajectory via

$$\hat{w}(k) = x(k+1) - A_{\theta'(k)}x(k) - B_{\theta'(k)}u(k) \quad \forall k \in [0, t-1], \tag{4.10}$$

where $\theta'(t)$ is a specific mode in $\mathbf{m}_{t+1}$ chosen as is done in [166]. One can observe that, $\tilde{\gamma}|_{\mathbf{m}}$ is a linear function of the state-input trajectory as it is a composition of a linear function with a linear function. Moreover, parameterizing the controller with gains $K^{(\mathbf{m}_{[1:t+1]})}$, $k^{(\mathbf{m}_{[1:t+1]})}$ that multiply the disturbances results in control parameters entering the problem in a linear way [60].

The inner controller $\tilde{\gamma}|_{\mathbf{m}}$ can then be defined entirely by a pair of matrices $(K^{(\mathbf{m}_{[1:t+1]})}, k^{(\mathbf{m}_{[1:t+1]})})$ at each time t (i.e., T pairs of matrices in total for a horizon of T). To simplify the controller analysis, the inner controller will instead be represented with the following *block* matrices for each $\mathbf{m}$:

$$\mathbf{k}^{(\mathbf{m})} \triangleq \begin{bmatrix} (k^{(\mathbf{m}_{1:1})})^\top & (k^{(\mathbf{m}_{1:2})})^\top & \cdots & (k^{(\mathbf{m}_{1:T})})^\top \end{bmatrix}^\top,$$

$$\mathbf{K}^{(\mathbf{m})} \in \mathbb{R}^{n_u T \times n_w T},$$

where for $\tau > 0$, the $(\tau+1)^{th}$ block row (i.e., the rows $\tau n_u + 1$ to rows $(\tau+1)n_u$) are given by

$$(\mathbf{K}^{(\mathbf{m})})_{\tau n_u + 1:(\tau+1)n_u} \triangleq \begin{bmatrix} K^{(\mathbf{m}_{1:\tau+1})} & 0_{n_u \times (T-\tau)n_w} \end{bmatrix}.$$

Note that the first block row is all zeros in $\mathbf{K}^{(\mathbf{m})}$. There is no component $K^{(\mathbf{m}_{1:1})}$ because no estimate can be constructed with just $x(0)$. Without the estimate at time 0, there is no need for the proportional gain in (4.9).

For this specific choice of controller, many of the constraints and sets discussed thus far remain polytopic. For instance closed-loop consistency and closed-loop reachable behavior sets can be expressed as in Definition 10 and Equation 4.5 with additional linear constraints. A linearly constrained polytope is still a polytope, which allows us to make the following claim about the closed-loop reachable behavior set:

Lemma 6. *For any pair of estimation sequences $\mathbf{m}' \in (2^{\Theta})^T$ and $\mathbf{m} \in \mathcal{M}$ created according to* (4.3)*, the closed-loop reachable behavior set $\mathcal{R}(\mathbf{m}', \tilde{\gamma}|_{\mathbf{m}})$ is a polytope.*

Proof. First, note that $\mathcal{R}(\mathbf{m})$ is a polytope as shown in Lemma 4. Next, because we have made the assumption that the controller component uses disturbance feedback of the form (4.9), the constraint $u(k) = \tilde{\gamma}|_{\mathbf{m}}(k)$ is equal to

$$u(k) = K^{(\mathbf{m}_{0:k})}\hat{w} + k^{(\mathbf{m}_{0:k})} \quad \text{for all } k.$$

Therefore, the closed loop reachable behavior set adds an affine constraint between $u(k)$ and $\hat{w}(0 : k-1)$ (which is itself a linear function of the state-input trajectory $(x(0:k), u(0:k-1))$). Thus, it is still a polytope. □

Given that the closed loop reachable behavior set is also a polytope, we can now write down the previous conditions on reachable behavior sets in terms of the new closed-loop reachable behavior set. Specifically, the reachability constraints take a nice linear form when disturbance feedback controllers are used. In particular, we have the following counterpart to Eq. (4.8). If for every $\mathbf{m} \in \mathcal{M}$ the controller $\tilde{\gamma}$ defined in (4.9) satisfies

$$\begin{aligned} &\forall (x(0:T-1), u(0:T-2)) \in \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) : \\ &\quad \forall \theta \in \mathbf{m}_T, \forall w(T-1) \in \mathcal{W}_\theta : \\ &\qquad (A_\theta x(T-1) + B_\theta(K^{(\mathbf{m})}\hat{w}(0:T-2) + k^{(\mathbf{m})}) + w(T-1) \in \mathcal{X}_T \end{aligned} \tag{4.11}$$

where $\hat{w}(k)$ is the estimate of the disturbance created using (4.10), then the system will reach the target set.

4.5 Optimization-Based Solution

In this section, we present an optimization problem that searches for a solution to Problem 2 in the form of a controller with the structure of Figure 4.1. The optimization problem is shown to be bilinear, hence it can be solved using off-the-shelf tools for exploration-exploitation profile $\mathcal{M}^*$. We first analyze this optimization problem and then point out design methods for generating alternative exploration-exploitation profiles $\mathcal{M}$.

Proposition 2 is encoded into a bilinear constraint for an adaptive controller in the following proposition:

Proposition 7. *Consider a linear system with the collection of modes Θ, a time horizon T and a*

target set $\mathcal{X}_T$. If there exists a solution to the following feasibility problem

$$
\begin{aligned}
&\textit{Find } \left\{\mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})}\right\}_{\mathbf{m}\in\mathcal{M}^*} \\
&\quad \textit{s.t. } \forall \mathbf{m} \in \mathcal{M}^* : \\
&\qquad \tilde{\gamma}|_{\mathbf{m}} \textit{ defined as in } (4.9) && \text{(4.12a)} \\
&\qquad (4.11) \textit{ holds} && \text{(4.12b)} \\
&\qquad \forall (x(0:T-1), u(0:T-2)) \in \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}), \forall t \in [0, T-1] : && \text{(4.12c)} \\
&\qquad\quad \tilde{\gamma}|_{\mathbf{m}}(t) \in \mathcal{U} && \text{(4.12d)} \\
&\quad \forall \mathbf{m}, \mathbf{m}' \in \mathcal{M}^* : \\
&\qquad \mathbf{m}_{[1:t]} = \mathbf{m}'_{[1:t]} \implies \\
&\qquad\quad K^{(\mathbf{m}_{[1:t]})} = K^{(\mathbf{m}'_{[1:t]})}, \quad k^{(\mathbf{m}_{[1:t]})} = k^{(\mathbf{m}'_{[1:t]})}, && \text{(4.12e)}
\end{aligned}
$$

then the estimator (4.3) *and the disturbance feedback gains which solve the problem define an adaptive controller that solves Problem 2. Moreover, this problem can be rewritten as a bilinear program.*

Proof. First, note that the set of gains $\left\{\mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})}\right\}_{\mathbf{m}\in\mathcal{M}^*}$ which satisfies (4.11) alone may not define a controller function $\gamma^{(\mathcal{M}^*)}$ that is causal in the estimated mode sequence. This is because the set of gains may define a non-causal controller or a controller that does not satisfy the input constraints of the problem. (4.12e) guarantees causality of the controller by enforcing the same gains to be used whenever the mode estimation prefixes are the same, similar to our earlier work [127].

In addition to the causality constraints, we also constrain our controller to produce control inputs within the set $\mathcal{U}$. The constraints (4.12d) and (4.12c) guarantee that the input obeys this constraint for any possible state-input trajectory at any possible time in the time horizon T.

Furthermore, because the constraint (4.11) is satisfied, the causal controller defined by $\left\{\mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})}\right\}_{\mathbf{m}\in\mathcal{M}^*}$ also solves Problem 2.

To show that (4.12) is a bilinear problem, we will analyze each constraint.

First, note that the constraints (4.12e) are linear equality constraints on the gain variables.

Second, the constraint (4.11) can be written as the following set containment:

$$
\begin{aligned}
&\forall \theta \in \mathbf{m}_{T-1} : \\
&\quad R_{T-1}\mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \oplus B_\theta(K^{(\mathbf{m})}\widehat{\mathcal{W}}_\theta(\mathbf{m}) \oplus \{k^{(\mathbf{m})}\}) \oplus \mathcal{W}_\theta \subseteq \mathcal{X}_T
\end{aligned}
$$

where R_{T-1} is the matrix that extracts the state $x(T-1)$ from any state-input trajectory

$\begin{bmatrix} x(0:T-1) \\ u(0:T-2) \end{bmatrix}$, and the estimated disturbance set is defined as

$$\widehat{\mathcal{W}}^{(\mathbf{m})} = \left\{ H_{\widehat{\mathcal{W}}(\mathbf{m})}(\mathbf{m}, K^{(\mathbf{m})}, k^{(\mathbf{m})}) w \leq h_{\widehat{\mathcal{W}}_i(\mathbf{m})}(\mathbf{m}, K^{(\mathbf{m})}, k^{(\mathbf{m})}) \right\}$$

i.e. a polytope whose hyperplane matrices are a linear function of the decision variables. This containment constraint, upon application of Lemma 1, results in bilinear constraints between the containment dual variable Λ and the decision variables. Similarly, (4.12c) and (4.12d) results in bilinear terms because they are similar polytope containment conditions. Thus, the feasibility problem contains linear and bilinear optimization constraints. □

A few remarks are in order to put Proposition 7 into context.

First, a robust controller (i.e. a controller which ignores mode information) can be obtained by imposing the additional constraint that $(\mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})})$'s are the same for all $\mathbf{m} \in \mathcal{M}^*$. Such a controller does not need a mode-estimator to be implemented. By definition, the adaptive controller developed in this chapter is feasible whenever a robust controller for Problem 2 exists.

Next, the computational complexity of the problem (4.12) depends on the cardinality $|\mathcal{M}^*|$ of the exploration-exploitation profile $\mathcal{M}^*$. As the number of elements in $\mathcal{M}^*$ increases or decreases, so do the number of bilinear constraints (from (4.12c), (4.12d), and (4.11)); the number of controller parameters in the problem scales with $O(|\mathcal{M}^*|)$. The quantity $|\mathcal{M}^*|$ itself can scale prohibitively, especially with large time horizons, $|\mathcal{M}^*| = O(2^{|\Sigma|^T})$. Finally, we note that since we avoid state-space discretization, the scalability with respect to the continuous state-space dimension is more favorable when using an off-the-shelf industry-grade solver [63] compared to abstraction-based methods.

In the next section, we discuss alternative exploration-exploitation profiles together with the implications of using different profiles on the computational complexity.

4.5.1 Other Exploration-Exploitation Profiles

So far we have assumed that the exploration-exploitation profile $\mathcal{M}^*$ is used. This section considers the adaptive controller design problem when profile $\mathcal{M} \subset \mathcal{M}^*$ is chosen. The computational advantage of choosing a different $\mathcal{M}$ is discussed before the requirements of such an exploration-exploitation profile are evaluated.

While Proposition 7 provides a sufficient condition for the existence of an adaptive controller, the number of decision variables scales poorly with the time horizon T and the number of modes $|\Theta|$. For this reason, it would be helpful to choose the exploration-exploitation profile $\mathcal{M} \subset \mathcal{M}^*$ instead. For some systems (e.g. Example 1), reachability analysis of the first few steps in the time

horizon can reveal that some elements $\mathbf{m} \in \mathcal{M}^*$ are not possible (i.e., the estimator will identify the mode at time $t = 1$, so there is no need to consider $\mathbf{m} \in \mathcal{M}^*$ where $|\mathbf{m}_{k+1}| > 1$ for any $k > 2$). Verifying that an estimator sequence $\mathbf{m} \in \mathcal{M}^*$ is not possible can be done with the following lemma:

Lemma 7. *An estimation sequence $\mathbf{m} \in \mathcal{M}^*$ can not be produced by any state-input trajectory of the system* (4.1) *if $\mathcal{R}(\mathbf{m}) = \emptyset$.*

Proof. This proof follows from the definition of $\mathcal{R}(\mathbf{m})$. □

We can use Lemma 7 to obtain a lower cardinality exploration-exploitation profile $\mathcal{M}$ by removing from $\mathcal{M}^*$ the estimation sequences $\mathbf{m}$ such that $\mathcal{R}(\mathbf{m}) = \emptyset$. This will result in less variables in the optimization problem (4.12). However, if we want to reduce the cardinality further, this should be done carefully.

Specifically, a solution to Problem 2 with a smaller profile $\mathcal{M} \subset \mathcal{M}^*$ must satisfy both a reachability condition *and* a condition on the set of all state-input trajectories produced by controller. The reachability condition has been discussed previously (i.e., Proposition 6). The condition on all state-input trajectories is stated as: For every closed-loop state-input trajectory $(x(0 : T-1), u(0 : T-2))$, $\boldsymbol{\mu}(x(0 : T-1), u(0 : T-2)) \in \mathcal{M}$. This condition is guaranteed when considering the profile $\mathcal{M}^*$ because, by definition of $\mathcal{M}^*$, $\boldsymbol{\mu}(T-1) \in \mathcal{M}^*$ for any state-input trajectory of a system. Such a guarantee is not provided for $\mathcal{M} \subset \mathcal{M}^*$.

To guarantee that the inner controller is defined for every estimation sequence $\mathbf{m}_{1:t+1}$ (and its associated state-input sequence), a condition is added which enforces that all $\mathbf{m}$ generated by the system's closed loop state-input trajectories belong to the profile $\mathcal{M}$. To achieve this, we introduce the following proposition:

Proposition 8. *Consider the system* (4.1) *with modes Θ. Let $\mathcal{M} \subseteq (2^{\Theta})^T$ and $\{\tilde{\gamma}|_{\mathbf{m}}\}_{\mathbf{m}\in\mathcal{M}}$ be such that $\forall \mathbf{m}^c \in \mathcal{M}^* \setminus \mathcal{M}, \mathbf{m} \in \mathcal{M}$,*

$$\mathbf{m}^c \subseteq \mathbf{m} \implies \mathcal{R}(\mathbf{m}^c, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \tag{4.13a}$$

$$\mathbf{m}^c \nsubseteq \mathbf{m} \implies \mathcal{R}(\mathbf{m}^c, \tilde{\gamma}|_{\mathbf{m}}) = \emptyset \tag{4.13b}$$

then the set $\mathcal{M}$ contains mode estimation sequences for all possible closed-loop state-input trajectories, i.e.

$$\mathcal{R}^* = \bigcup_{\mathbf{m}\in\mathcal{M}} \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \tag{4.14}$$

where

$$\mathcal{R}^* \doteq \left\{ (x(0:T-1), u(0:T-2)) \;\middle|\; \begin{array}{l} x(0) \in \mathcal{X}_0, u(0:t-1) \in \mathcal{U}^t \\ \exists \theta \in \Theta : \forall k \in [0, T-2] : \\ \quad x(k+1) - A_\theta x(k) + B_\theta u(k) \in \mathcal{W}_\theta \\ \quad u_k = \gamma^{(\mathcal{M})}(x(0:k), u(0:k-1)) \end{array} \right\}.$$

Proof. By definition, $\mathcal{R}^* \supseteq \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$ for any $\mathbf{m} \in (2^\Theta)^T$. $\mathcal{R}^* \supseteq \bigcup_{\mathbf{m}\in\mathcal{M}} \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$ quickly follows from that.

Now, consider the arbitrary state-input trajectory $(x(0:t), u(0:t-1)) \in \mathcal{R}^*$. If there exists a sequence $\mathbf{m} \in \mathcal{M}$ such that $(x(0:t), u(0:t-1)) \in \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$, then we can show that $\mathcal{R}^* \subseteq \bigcup_{\mathbf{m}\in\mathcal{M}} \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$, completing the proof.

First, note that for $(x(0:t), u(0:t-1))$ there must exist a value $\mathbf{m}^* \in (2^\Theta)^T$ such that $(x(0:t), u(0:t-1)) \in \mathcal{R}(\mathbf{m}^*)$. By definition, $\mathbf{m}^*$ will belong to $\mathcal{M}^*$.

Suppose that $\mathbf{m}^* \notin \mathcal{M}$. It is not possible for $\mathbf{m}^* \not\subseteq \mathbf{m}$ for all $\mathbf{m} \in \mathcal{M}$. By (4.13b), if $\mathbf{m}^* \not\subseteq \mathbf{m}$ for all $\mathbf{m} \in \mathcal{M}$, then $\mathcal{R}(\mathbf{m}^*) = \emptyset$ and thus no data will cause $\mathbf{m}^*$.

It is also not possible for $\mathbf{m}^* \in \mathcal{M}^c$ with at least one $\mathbf{m} \in \mathcal{M}$ such that $\mathbf{m}^* \subseteq \mathbf{m}$. By (4.13a), any state-input trajectory that satisfies $(x(0:t), u(0:t-1)) \in \mathcal{R}(\mathbf{m}^c, \tilde{\gamma}|_{\mathbf{m}})$ will simultaneously satisfy $(x(0:t), u(0:t-1)) \in \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$. Thus, the estimator will select $\mathbf{m}$ instead of $\mathbf{m}^*$ as its estimate (because $\mathbf{m}^* \subseteq \mathbf{m}$) and $\mathbf{m}^*$ is not possible. Thus, by contradiction, $\mathbf{m}^* \in \mathcal{M}$ for an arbitrary element $(x(0:t), u(0:t-1)) \in \mathcal{R}^*$ and $\mathcal{R}^* \subseteq \bigcup_{\mathbf{m}\in\mathcal{M}} \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$. □

For any $\mathcal{M} \subset \mathcal{M}^*$, satisfaction of Propositions 6 and 8 guarantees that Problem 1 is correctly solved. This requires reasoning about the emptiness of polytopic sets (see constraint (4.13b)), which can be done with a bilinear constraint using Lemma 2.

Proposition 9. *Consider a linear system of the form* (4.1)*, a time horizon T, an exploration-exploitation profile $\mathcal{M} \subset \mathcal{M}^*$ and a target set $\mathcal{X}_T$. If there exists a solution to the following feasibility problem*

$$\begin{aligned} &\textit{Find } \left\{\mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})}\right\}_{\mathbf{m}\in\mathcal{M}} \\ &\quad \textit{s.t. } (4.12) && \text{(4.15a)} \\ &\qquad \forall \mathbf{m}^c \in \mathcal{M}^* \setminus \mathcal{M}, \mathbf{m} \in \mathcal{M} : \\ &\qquad\quad \mathbf{m}^c \subseteq \mathbf{m} \implies \mathcal{R}(\mathbf{m}^c, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) && \text{(4.15b)} \\ &\qquad\quad \mathbf{m}^c \not\subseteq \mathbf{m} \implies \mathcal{R}(\mathbf{m}^c, \tilde{\gamma}|_{\mathbf{m}}) = \emptyset && \text{(4.15c)} \end{aligned}$$

then the estimator (4.3) *and the disturbance feedback gains which solve the problem define an adaptive controller that solves Problem 2. Moreover, this problem can be rewritten as a bilinear*

program.

Proof. A solution to (4.15) necessarily satisfies (4.12) and thus guarantees that all state-input trajectories from the set $\cup_{\mathbf{m}\in\mathcal{M}}\mathcal{R}(\mathbf{m})$ are steered to reach the target set $\mathcal{X}_T$. By Proposition 8, $\cup_{\mathbf{m}\in\mathcal{M}}\mathcal{R}(\mathbf{m}) = \mathcal{R}^*$ and thus *all reachable behaviors* of the system reach the target set $\mathcal{X}_T$. Therefore, a solution to (4.15) defines a causal controller that solves Problem 2.

The problem is bilinear due to the constraints discussed in (4.12) as well as the novel constraints (4.15b) and (4.15c). Specifically, (4.15b) defines a set containment constraint for polytopes of the form:

$$\mathcal{R}(\mathbf{m}^c, \tilde{\gamma}|_{\mathbf{m}}) = \left\{ H_{\mathcal{R}(\mathbf{m}^c,\tilde{\gamma}|_{\mathbf{m}})}(\mathbf{m}^c, \mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})}) \begin{bmatrix} x(0:T-1) \\ u(0:T-2) \end{bmatrix} \leq h_{\mathcal{R}(\mathbf{m}^c,\tilde{\gamma}|_{\mathbf{m}})}(\mathbf{m}^c, \mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})}) \right\}$$

where $H_{\mathcal{R}(\mathbf{m}^c,\tilde{\gamma}|_{\mathbf{m}})}(\mathbf{m}^c, \mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})})$ and $h_{\mathcal{R}(\mathbf{m}^c,\tilde{\gamma}|_{\mathbf{m}})}(\mathbf{m}^c, \mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})})$ are linear functions of the matrices $\mathbf{K}^{(\mathbf{m})}$ and $\mathbf{k}^{(\mathbf{m})}$. To include this containment constraint into the optimization, Lemma 1 is used and will introduce constraints where Λ will multiply
$H_{\mathcal{R}(\mathbf{m}^c,\tilde{\gamma}|_{\mathbf{m}})}(\mathbf{m}^c, \mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})})$. This product is a bilinear expression. Furthermore, (4.15b) is included into the optimization problem via the application of Lemma 2. This constraint takes the form similarly introduces a new variable λ which is multiplied by one of the matrices $H_{\mathcal{R}(\mathbf{m}^c,\tilde{\gamma}|_{\mathbf{m}})}$ $(\mathbf{m}^c, \mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})})$.

Thus, the feasibility problem contains only linear and bilinear optimization constraints. □

Also note that the choice of $\mathcal{M}$ affects what information can be revealed about the system. For instance, taking $\mathcal{M} = \{2^\Sigma\}^T$, the only $\mathcal{M}$ with cardinality 1, we will be searching for controllers that guarantee reachability while making it impossible to identify the mode for any outside observer. By selecting $\mathcal{M}$, one can control at what time and to what level of accuracy the potential true system modes are discovered. This, in return, can be used to encourage controllers that explore or exploit more.

4.6 Results

In this section, Proposition 7 is used to synthesize controllers for four different case studies.[1] The first case study contains a problem where the exploration-exploitation profile $\mathcal{M}^*$ and another profile $\mathcal{M}$ satisfying $|\mathcal{M}| = 2$ are used to design an adaptive controller for the same system. For the first example, any controller that is applied will immediately gain information about the true

[1] An implementation of the optimization (4.12) and the code for reproducing the results can be found in https://tinyurl.com/ywfk8m7m.

mode of the system and can exploit it. The second case study is composed of a system with modes where the state of the system rotates counterclockwise about an unknown axis of rotation in the plane. The mode of the system (i.e., the axis of rotation) is very difficult to identify and thus it must be solved with a large exploration-exploitation profile. The third case study is a simplified version of a drone system that may or may not be carrying a payload. In this instance, the controller adapts to the payload so that the target set is reached. The final case study considers two similar triple integrator system with an uncontrollable subspace in their dynamics. By varying the dimension of this uncontrollable subspace, this case study displays how the time required to solve the controller synthesis problem varies with the dimension of the state dimension.

In all of the case studies, Gurobi [63] is used to solve the bilinear optimization problem (4.12). We use YALMIP [90] to formulate the constraints of the optimization problem which are later given to Gurobi and report both the constraint formulation time as well as the time it takes for Gurobi to find a solution.

4.6.1 Case Study 1: All Controllers Discriminate

Our first example consists of a system where modes are opposing rotations. Specifically, consider the following two mode system (i.e. $|\Theta| = 2$), where $(A_1, B_1, \mathcal{W}_1) = (R_{\pi/4}, I_2, \mathcal{W})$ and $(A_2.B_2, \mathcal{W}_2) = (R_{-\pi/4}, I_2, \mathcal{W})\}$. Note that the two systems are nearly identical except for the state update matrix A which is either a clockwise rotation by $\frac{\pi}{4}$ radians ($R_{\pi/4}$) or a counterclockwise rotation by $-\frac{\pi}{4}$ radians ($R_{-\pi/4}$).

The disturbance polytope $\mathcal{W}$ is a scaled hypercube $\mathcal{W} = \{w \in \mathbb{R}^2 \mid \|w\| \leq \eta_w\}$, $\eta_w = 0.25$, the initial state $x_0 = \begin{bmatrix} -1 & 0 \end{bmatrix}^\top$, the target set is and the time horizon $\mathcal{X}_T = \{x \in \mathbb{R}^2 \mid \|x - x_c\| \leq (4 + \sqrt{2})\eta_w\}$ centered on the point $x_c = \begin{bmatrix} 2 + 0.5\sqrt{2} & 0 \end{bmatrix}^\top$.

For this system, the two modes are immediately distinguishable as alluded to by Figure 4.3 (left) which shows how the reachable sets immediately separate when zero input is applied. A controller that does not use this information (i.e., is mode agnostic) cannot solve the problem. The adaptive controller produced by Proposition 7 does solve the problem as shown in Figure 4.3 (right).

When using $\mathcal{M}^*$, (4.12)'s constraints are constructed in 7.59 seconds via YALMIP and the bilinear optimizer solves the problem in 0.05 seconds.

The immediate separation of the reachable sets suggests that many of the consistency sets for $\mathbf{m} \in \mathcal{M}^*$ satisfy $\mathcal{C}(\mathbf{m}) = \emptyset$. In fact, the algorithm correctly determines that the only two estimation sequences that have nonempty consistency sets (out of 7) are:

$$\mathbf{m}^{(1)} = \Theta\{1\}\{1\}\{1\}, \text{ and } \mathbf{m}^{(2)} = \Theta\{2\}\{2\}\{2\}.$$

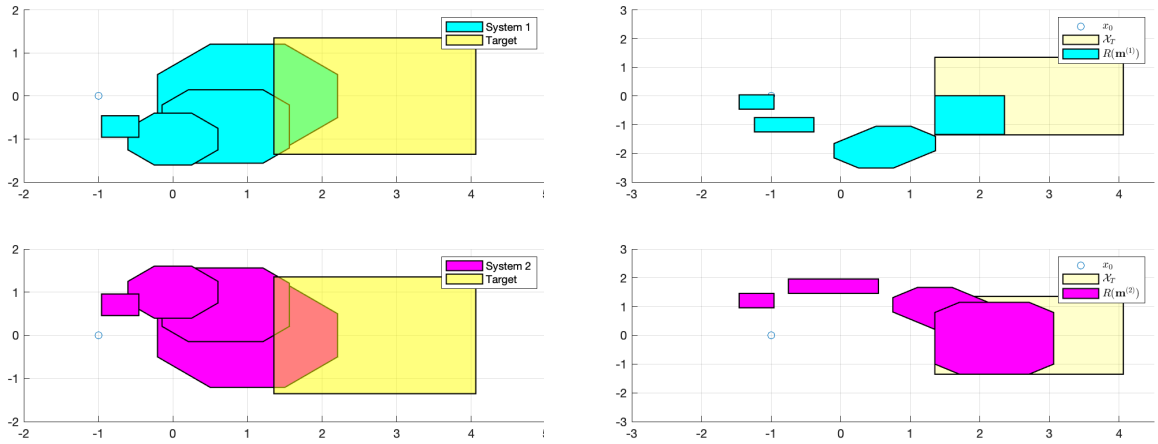


Figure 4.3: Left: Reachable sets for the two modes in the Opposing Rotations system. Note that, with zero input, each system cannot guarantee that the target set will be reached, but with a very simple closed loop controller, both systems can be identified at time 1 and then steered into the target set. Right: The closed-loop reachable sets where the adaptive controller guarantees that the target state (yellow) is reached regardless of if system 1 (top figure) or system 2 (bottom figure) is the true mode of the system.

The two estimation sequences are used to create $\mathcal{M} = \{\mathbf{m}^{(1)}, \mathbf{m}^{(2)}\}$, only $\mathcal{M} = \{\mathbf{m}^{(1)}, \mathbf{m}^{(2)}\}$. This is exactly as we expected from our intuition about the problem. For this value of $\mathcal{M}$, the constraints of (4.12) are formed by YALMIP in 11.07 seconds and Gurobi solves the problem 0.04 seconds.

4.6.2 Case Study 2: Mode Discrimination is Not Possible

Now we consider a system where the state update matrices of each mode rotates the state about an unknown pivot point in $\mathbb{R}^2$. In particular, consider the following two mode system (i.e. $|\Theta| = 2$), where $(A_1, B_1, \mathcal{W}_1) = (R_{\pi/12}, I_2, \mathcal{W}_1)$ and $(A_2, B_2, \mathcal{W}_2) = (R_{\pi/12}, I_2, \mathcal{W}_2)\}$. Here the state update matrix A is identical along with the input matrix B. A is a rotation matrix for $\frac{\pi}{12}$ radians and I_2, the input matrix, is the two-dimensional identity matrix. The two systems only differ in their disturbance matrices. Here:

$$\mathcal{W}_1 = -R_{\pi/12}\begin{bmatrix}0\\11\end{bmatrix} + \begin{bmatrix}0\\11\end{bmatrix} + \mathbb{B}_2(0.5)\ ,\ \mathcal{W}_2 = -R_{\pi/12}\begin{bmatrix}0\\12\end{bmatrix} + \begin{bmatrix}0\\12\end{bmatrix} + \mathbb{B}_2(0.5)$$

where $\mathbb{B}_n(\eta) = \{w \in \mathbb{R}^n \mid \|w\|_\infty \leq \eta\}$ is the n-dimensional hypercube centered at the origin with sidelength 2η.

In this example, the state of the system is rotating about a point (either $[0 \ \ 11]^\top$ or $[0 \ \ 12]^\top$) as the controller seeks to reach a target set $\mathcal{X}_T = \mathbb{B}_2(2) + [9.5 \ \ 5]^\top$ at the end of the time horizon

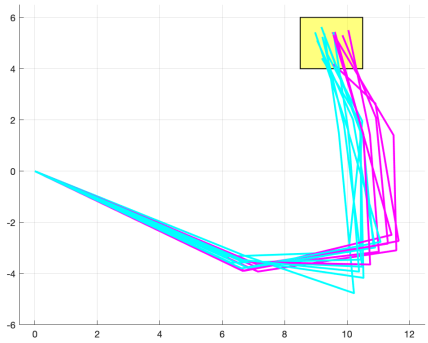


Figure 4.4: Ten different runs of the system in Example 4.6.2 with the controller synthesized. Runs of mode 1 (cyan) and mode 2 (magenta) both reach the target though they may or may not be identified.

$T = 4$.

For this system, it is not possible to guarantee that the mode is discriminated at any time in the time horizon.

Optimization problem (4.12)'s constraints are constructed in 25.93 seconds via YALMIP and the bilinear optimizer solves the problem in 0.26 seconds.

4.6.3 Case Study 3: Simplified-Drone System

In this example, a delivery drone performing altitude control is considered. The drone's attempt to reach a target altitude is complicated by the fact that it may or may not be carrying a package. This package has a known mass and the delivery drone would like to use a controller to identify the mass while making progress in the delivery at the same time.

A simplified model of the drone's motion can be described with the following system with two modes (i.e. $|\Theta| = 2$):

$$x(k+1) = A_\theta x(k) + B_\theta(u(k) + w(k)) + f_\theta$$

where

$$A_\theta = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \; B_\theta = \begin{bmatrix} 0 \\ \frac{\Delta t}{m_\theta} \end{bmatrix}, \; f_\theta = \begin{bmatrix} 0 \\ -g\Delta t \end{bmatrix} \quad \forall \theta \in \{1, 2\},$$

$\mathcal{W} = [-0.5, 0.5]$, $\mathcal{U} = [-40, 40]$

and $\mathcal{X}_T = [2, 4] \times [-10, 10]$.

The parameters used are given in Table 4.1.

Symbol	m_1	m_2	Δt	g	T
Value	1 kg	1.5 kg	0.1 s	10 m/s^2	8

Table 4.1: Constants used in the drone system's definition.

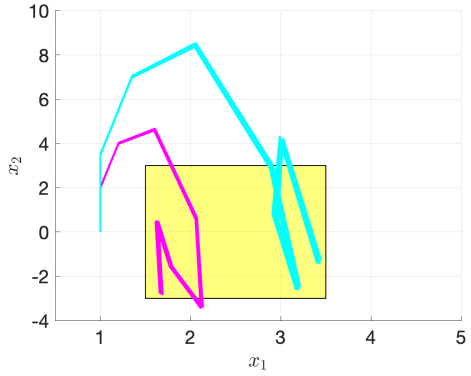


Figure 4.5: Several runs of the drone altitude controller with mass randomly selected for each run.

Proposition 7 is used to find a controller that identifies the mass while making progress towards the target (i.e., solves Problem 2). Optimization problem (4.12)'s constraints are constructed in 31.67 seconds via YALMIP and the bilinear optimizer solves the problem in 0.18 seconds.

4.6.4 Case Study 4: Scalability Analysis

In this example, the impact of system dimension on the control synthesis method is studied through a modified version of the system from [172]. The new system is composed of three loosely coupled double integrators and a tuneable number n_{uc} of uncontrollable dimensions. Thus, the dimension of the system is $6 + n_{uc}$.

Consider the following two mode system (i.e. $|\Theta| = 2$), where $(A_1^{(d)}, B^{(d)}, \mathcal{W})$ and $(A_2^{(d)}, B^{(d)}, \mathcal{W})$. The modes of the system are governed by the matrices:

$$A^{(c)} = \begin{bmatrix} \tilde{A} & A_{ur} \\ 0 & A_{uc} \end{bmatrix}, \quad B^{(c)} = \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix}$$

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

n_{uc}	Setup Time - YALMIP (s)	Solve Time - Gurobi (s)
0	65.79	0.55
2	81.87	0.82
4	87.24	1.28
6	111.80	1.74
8	129.26	2.64

Table 4.2: Controller synthesis times for systems of increasing dimension.

$$A_{ur} = \begin{bmatrix} A_{ur}^1 & A_{ur}^2 & A_{ur}^3 & A_{ur}^1 & A_{ur}^2 & A_{ur}^3 & \cdots \end{bmatrix}$$

$$A_{ur}^1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \; A_{ur}^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, \; A_{ur}^3 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$A_{uc} = \text{blockdiag}(\begin{bmatrix} -10^{-2} & 1 \\ 1 & -10^{-2} \end{bmatrix}, \begin{bmatrix} -10^{-4} & 2 \\ 2 & -10^{-4} \end{bmatrix}, \cdots, \begin{bmatrix} -10^{-n_{uc}} & \frac{n_{uc}}{2} \\ \frac{n_{uc}}{2} & -10^{-n_{uc}} \end{bmatrix})$$

With the discretization step of $\Delta t = 0.5$, $A_1^{(d)}$ is the time-discretized form of $A^{(c)}$, $B_1^{(d)}$ is the time-discretized form of $B^{(c)}$, and A_2 is the time-discretized version of $\tilde{A}'$ which contains the same entries as $\tilde{A}$ except $(\tilde{A}')_{4,2}$ which is defined as 0.2.

In this case study, an adaptive controller is sought for the system starting from $x_0 = \begin{bmatrix} -60 & 0 & -60 & 0 & -70 & 0_{1\times 1+n_{uc}} \end{bmatrix}^\top$ with the disturbance set $\mathcal{W} = [-1, 1]^{6+n_{uc}}$, time horizon $T = 5$ and target set:

$$\mathcal{X}_T = [-80, -40] \times [-10, 10] \times [-80, -40] \times [-10, 10] \times [-60, -20] \times [-10, 10] \times [-10^3, 10^3]^{n_{uc}}.$$

An analysis of the time required to find a solution to (4.12) for various values of n_{uc} is presented in Table 4.2.

CHAPTER 5

Controller Synthesis for KLTL Tasks

5.1 Introduction

As mentioned in Chapter 1 and discussed in detail in Chapter 4, designing decision-making systems that adapt to parametric uncertainty is necessary for formal methods to be applied in many real-world tasks. The method presented in Chapter 4 designed controllers using the set of potential models and the exploration-exploitation profile as inputs. The exploration-exploitation profile can be complicated to specify for a designer. The complexity of specifying exploration-exploitation profiles for a task is much higher than that of specifying temporal logics like LTL and STL, for a task. So, in this chapter, we present a method for performing the design task discussed in Chapter 4 but with a specification written in terms of a temporal logic (KLTL).

5.2 Problem Statement

We are interested in simplifying the results from Chapter 4 to make the synthesis problem be written in terms of simpler components. In that chapter, we presented a method for designing a controller that correctly satisfies tasks while trading off internally between exploration and exploitation. The trading off of exploration and exploitation was guided by an exploration-exploitation profile $\mathcal{M}$ in that work, but this parameter is novel and unfamiliar to others.

A better method for specifying exploration-exploitation trade-offs can be found through temporal logics like KLTL [31]. The temporal logic KLTL adopts many operators from LTL [14], but with an important component necessary for output feedback.

Definition 12 (KLTL Grammar)**.** *The grammar of KLTL is as follows:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \varphi \mathcal{U} \varphi \mid \mathcal{K}\varphi \tag{5.1}$$

in which $p \in AP$ is an atomic proposition, while $\bigcirc$ and $\mathcal{U}$ are the "next" and "until" operators. Formulas of the type $\mathcal{K}\varphi$ are read as "the system knows that the formula φ holds".

As is traditionally done in LTL [14] and STL [96], the operators (or "macros") $\Diamond$ (eventually) and $\Box$ (always) can be defined from the above grammar.

The semantics of this temporal logic will be defined over the uncertain linear system (2.5) discussed in Chapter 4. Recall its form below:

$$x(t+1) = A_\theta x(t) + B_\theta u(t) + w(t), \qquad w(t) \in \mathcal{W}_\theta \tag{5.2}$$

For more details about this system, please see Chapter 2 or 4. To develop semantics for this system, we introduce the concept of the labelling function and the set of trajectory-mode pairs.

Definition 13 (Labelling Function)**.** *A labelling function for a set of atomic propositions AP for the system* (5.2) *is a function that describes which atomic propositions are satisfied at a given state and with a given parameter, (i.e. $L : \mathcal{X} \times \Theta \to 2^{AP}$).*

Let the set of all feasible trajectory-mode pairs (x, θ) that (5.2) can produce for a given controller $\gamma^{(\mathcal{M}^*)}$ over time horizon T (i.e. $x = x(0:T)$) be:

$$\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)}) \triangleq \left\{ (x, \theta) \;\middle|\; \begin{array}{ll} \theta \in \Theta & \\ x(0) \in \mathcal{X}_0 & \\ x(k+1) - A_\theta x(k) - B_\theta u(k) \in \mathcal{W}_\theta & \forall k \in [0, T-1] \\ u(k) = \gamma^{(\mathcal{M}^*)}(x(0:k), u(0:k-1)) & \forall k \in [0, T-1] \end{array} \right\}$$

With this labelling function and the set $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ in mind, we now define the KLTL semantics for systems with unknown parameters.

Definition 14 (KLTL Semantics for Systems With Unkown Parameters)**.** *For any pair of a state trajectory x and unknown parameter θ, interpret the KLTL operators as follows:*

- $(x, \theta, i) \models p$ *if* $p \in L(x(i), \theta)$
- $(x, \theta, i) \models \neg p$ *if* $(x, \theta, i) \not\models p$
- $(x, \theta, i) \models \varphi_1 \wedge \varphi_2$ *if* $(x, \theta, i) \models \varphi_1$ *and* $(x, \theta, i) \models \varphi_2$
- $(x, \theta, i) \models \bigcirc\varphi$ *if* $(x, \theta, i+1) \models \varphi$
- $(x, \theta, i) \models \varphi_1 U \varphi_2$ *if* $\exists j \geq 0$ *such that* $(x, \theta, j) \models \varphi_2$ *and* $\forall 0 \leq k < j$, $(x, \theta, k) \models \varphi_1$,
- $(x, \theta, i) \models \mathcal{K}\varphi$ *if for all* $(x', \theta') \in \mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ *s.t.* $(x, \theta) \sim_i (x', \theta')$, *we have* $(x', \theta', i) \models \varphi$.

In our case, we define the similarity relationship $\sim_i$ as $(x, \theta) \sim_i (x', \theta')$ if and only if $x(0:i) = x'(0:i)$.

Remark 7. *Normally, temporal logic semantics are independent of controllers like $\gamma^{(\mathcal{M})}$. We note that this is also the case for KLTL semantics, but to simplify this exposition we have included the set of all closed loop trajectories into the semantic definition of $\mathcal{K}$. The set $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ can be replaced by any set of trajectory-mode pairs $\mathcal{T}$ that the system* (2.5) *is restricted to (either with or without control).*

Remark 8. *KLTL appears to be so similar to LTL that some might wonder whether or not it is truly a distinct language (i.e., a curious reader might ask whether or not the $\mathcal{K}$ operator is a macro). KLTL is distinct from LTL as is described in Appendix C. The distinction can be stated in fewer words by saying that KLTL can be used to define hyperproperties while LTL can not.*

Remark 9. *KLTL is defined independently of the estimator used in an adaptive controller. This degree of freedom also makes the synthesis problem different from other LTL-based approaches.*

Now that we have introduced KLTL, we can formally state the problem that we would like to solve in this chapter.

Problem 3. *Consider an uncertain linear system* (2.5) *and a KLTL formula φ. Identify a controller γ such that the closed loop system satsifies the formula (i.e. $(x, \theta, 0) \models \varphi$ for all $(x, \theta) \in \mathcal{T}(\Theta, \gamma^{(\mathcal{M})})$) according to the semantics from Definition 14.*

5.3 Approach

Our approach solves the problem by manipulating the set of all trajectory-mode pairs $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$. This set can be decomposed into a union of polytopes, as shown in the following section. Each element in this union is a set of trajectories that satisfy a particular KLTL formula. By identifying the KLTL formulas that are implied by the overall task φ in Problem 3, we create a framework that can be used to find controllers γ that properly manipulate $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ and thus solve Problem 3.

In this section, we will repeatedly use the above approach to discuss how to find controllers that solve Problem 3 for several classes of KLTL formulas (i.e., specific KLTL formula templates). After the approach is understood for these KLTL formula templates, then it should be possible to formulate a solution to Problem 3 for any KLTL formula.

Before presenting our analysis of different templates, we first introduce the following assumptions:

Assumption 1. *Throughout this chapter assume that each atomic proposition p is associated with a polytopic set $\mathcal{S}_p \subseteq \mathcal{X}$ or a discrete set $\tilde{\Theta}_p \subseteq \Theta$ such that either:*

- $x \in \mathcal{S}_p \iff p \in L(x, \theta)$, *or*
- $\theta \in \tilde{\Theta}_p \iff p \in L(x, \theta)$.

A proposition that has an associated polytopic set representation in the state space will be called a *state-based proposition* and a proposition that has an associated discrete set Θ_p will be called a *model-based proposition*.

For the sake of illustrating our sketch of a solution to Problem 3, we discuss what constraints need to be satisfied in order for the system (5.2) with controller γ to satisfy simple formulas. We discuss what constraints are needed in order to satisfy:

- $(x, \theta, 0) \models p$
- $(x, \theta, 0) \models \bigcirc^k \varphi$
- $(x, \theta, 0) \models \varphi_1 U \varphi_2$
- $(x, \theta, 0) \models \mathcal{K}\varphi$

The other formula components ($\neg\varphi$, $\varphi_1 \wedge \varphi_2$) can all be interpreted in a straightforward manner from our definitions of the above operators.

We are interested in making guarantees about $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ for system (5.2) with controller $\gamma^{(\mathcal{M}^*)}$. To analyze $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$, we represent it with a combination of convex sets.

Lemma 8. *The set of all feasible, closed-loop trajectory-mode pairs $(x(0:T), \theta)$ that a system (2.5) can produce for a given controller $\gamma^{(\mathcal{M}^*)}$ over time horizon T is equivalent to the following union of polytopes:*

$$\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)}) = \bigcup_{\mathbf{m} \in \mathcal{M}^{**}} \bigcup_{\theta \in \mathbf{m}_{T+1}} R_{0:T} \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \times \{\theta\} \tag{5.3}$$

where $R_{[0:T]}$ is a matrix that selects the state from time 0 to the end of the trajectory ($x(k:T-1)$) of the vector $\begin{bmatrix} x(0:T-1)^\top & u(0:T-2)^\top \end{bmatrix}^\top$ and we introduce the one-step extension $\mathcal{M}^{**}$ of the profile $\mathcal{M}^*$ with controller $\gamma^{(\mathcal{M}^*)}$ as:

$$\mathcal{M}^{**} \triangleq \left\{ \mathbf{m} \in (2^\Theta)^{T+1} \;\middle|\; \begin{array}{l} \exists \theta \in \Theta, (x(0:T), u(0:T-1)) : \\ \quad x(k+1) - A_\theta x(k) - B_\theta u(k) \in \mathcal{W}_\theta \quad \forall k \in [0, T-1] \\ \quad u(k) \in \mathcal{U} \quad \forall k \in [0, T-1] \\ \quad \mathbf{m}_{k+1} = \mu(x(0:k), u(0:k-1)) \quad \forall k \in [0, T-1] \end{array} \right\}.$$

Proof. This proof holds given (4.14) and the fact that $x(0:T)$ can be exactly defined as belonging to a polytopic projection of $\mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$. □

Remark 10. *We have assumed that the controller $\gamma^{(\mathcal{M}^*)}$ is written with respect to $\mathcal{M}^*$ to guarantee that the set is well-defined. The profile $\mathcal{M}^*$ may be replaced with any exploration-exploitation profile $\mathcal{M} \subseteq \mathcal{M}^*$ that will also guarantee $\mathcal{T}(\Theta, \gamma^{(\mathcal{M})})$ is well-defined (i.e., any $\mathcal{M}$ such that* (4.13) *are satisfied).*

5.3.1 Satisfying Atomic Propositions

The following two propositions describe how to verify whether or not a formula composed of a single atomic proposition can be evaluated when given Assumption 1.

Proposition 10. *Consider an uncertain linear system* (5.2) *and a state-based atomic proposition p. The system under controller γ satisfies the proposition $\varphi = p$ if for all $\mathbf{m} \in \mathcal{M}^{**}$:*

$$R_0 \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{S}_p \tag{5.4}$$

where R_0 is a matrix that selects the state at time 0 ($x(0)$) of the vector $\begin{bmatrix} x(0:T)^\top & u(0:T-1)^\top \end{bmatrix}^\top$.

Proof. Each trajectory-mode pair $(x(0:T), \theta) \in \mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ satisfies the formula φ if the first state of the trajectory is in the set $\mathcal{S}_p$.

The proof of this proposition holds because of the union in (5.3). If each of the reachable sets $\widetilde{\mathcal{R}}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}, \theta)$ satisfies (5.4), then the full set $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ satisfies the set containment as well. Thus, all trajectories of the system satisfy the single state-based formula. □

The constraint (5.4) is a polytope containment constraint. Thus, we know it can be transformed into a set of bilinear constraints using Lemma 1. So, for state-based atomic proposition p, finding a controller that satisfies the bilinear constraint (5.4) guarantees that the proposition p is satisfied.

Proposition 11. *Consider an uncertain linear system* (5.2) *and a model-based atomic proposition p. The system under controller γ satisfies the proposition $\varphi = p$ if and only if $\Theta_p = \Theta$.*

Proof. The formula is only satisfied if $\theta \in \Theta_p$ for all $(x, \theta) \in \mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$. By definition, every mode $\theta \in \Theta$ is represented in $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$, so the formula is only satisfied if Θ_p also contains all modes (i.e. $\Theta_p = \Theta$). □

Thus, only one specific model-based atomic proposition can be satisfied in this way. This makes sense as the model is not influenced by the controller and can take any value in Θ.

5.3.2 Satisfying Formulas with Repeated Next Operators

If Problem 3 is solved for a formula ($\bigcirc\varphi$), then all trajectories of system (5.2) satisfy formula φ at time $t = 1$. Such trajectories satisfy the formula at the "next" time after $t = 0$ (where our semantics normally start). So, when the next operator is repeated more than once:

$$\bigcirc^k = \underbrace{\bigcirc\bigcirc\cdots\bigcirc}_{k \text{ times}}$$

then the formula φ should be satisfied at time step $t = k$ in order to satisfy $\bigcirc^k\varphi$.

This is equivalent to simply applying a linear transform to the set $\mathcal{R}^*$ and thus can be easily incorporated into convex optimization-based approaches.

Proposition 12. *Consider an uncertain linear system* (5.2) *and an state-based proposition* p*. The system under controller* $\gamma^{(\mathcal{M}^*)}$ *satisfies the formula* $\bigcirc^k p$ *if for all* $\mathbf{m} \in \mathcal{M}^{**}$*:*

$$R_0 \cdot R_{[k:]}\mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{S}_p \tag{5.5}$$

where $R_{[k:]}$ *is a matrix that selects the state from time* k *to the end of the trajectory (*$x(k:T)$*) of the vector* $\begin{bmatrix} x(0:T)^\top & u(0:T-1)^\top \end{bmatrix}^\top$.

Proof. The proof of this is similar to that of Proposition 10. By proving that every k^{th} element of the state sequence satisfies proposition p, we prove that the formula is satisfied. If the k^{th} element of the trajectory satisfies proposition p in each $\widetilde{\mathcal{R}}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$, then by (5.3) all trajectory-mode pairs in $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ satisfy the formula. □

Again, (5.5) is a polytope containment constraint. Thus, we know it can be transformed into a set of bilinear constraints using Lemma 1. So, for state-based atomic proposition p, finding a controller that satisfies the bilinear constraint (5.5) guarantees that the proposition $\bigcirc^k p$ is satisfied.

Proposition 13. *Consider an uncertain linear system* (5.2) *and a model-based proposition* p*. The system under controller* γ *satisfies the proposition* $\bigcirc^k p$ *if and only if* $\Theta_p = \Theta$*.*

Proof. This proof follows from the proof of Proposition 11. □

Thus, only one specific model-based atomic proposition for which $\bigcirc^k$ can be satisfied. This makes sense as the model is not influenced by the controller and can take any value in Θ.

Remark 11. *The approach for finding controllers that satisfy* $\varphi^k p$ *can be extended to find controllers that satisfy* $\Diamond p$ *as well. This is because the formula* $\Diamond p$ *is satisfied if there exists a* k *such that* $\bigcirc^k\varphi$ *is satisfied. Finding a* k *and a controller such that* $\bigcirc^k\varphi$ *is satisfied (for finite time horizon problems) can be done by formulating a MILP with the constraints from* (5.5) *and decision variable* k*.*

5.3.3 Satisfying Formulas with the Until Operator

If Problem 3 is solved for a formula $(\varphi_1 U \varphi_2)$, then all trajectories of system (5.2) satisfy formula φ_1 for all times up until some time $k \geq 0$ where φ_2 must be satisfied.

When the formulas φ_1 and φ_2 are both state-based atomic propositions, this is equivalent to a sequence of many polytope containment constraints which can be easily incorporated into convex optimization-based approaches.

Proposition 14. *Consider an uncertain linear system* (5.2) *and state-based atomic propositions* p_1, p_2*. The system under controller* γ *satisfies the formula* $p_1 U p_2$ *if there exists a* $k \geq 0$ *such that for all* $\mathbf{m} \in \mathcal{M}^{**}$*:*

$$R_i \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{S}_{p_1} \quad \forall i \leq k \tag{5.6}$$

and

$$R_{k+1} \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{S}_{p_2}. \tag{5.7}$$

Proof. The proof of this proposition can be thought of as a combination of the proofs of Propositions 10 and 12. □

Again, (5.6) and (5.7) define a collection of many polytope containment constraints. Thus, we know they can be transformed into a set of bilinear constraints using Lemma 1. So, for state-based atomic propositions p_1 and p_2, finding a controller that satisfies the bilinear constraint (5.5) guarantees that the formula $p_1 U p_2$ is satisfied.

5.3.4 Satisfying Formulas with the Knowledge Operator

If Problem 3 is solved for a formula $(\mathcal{K}\varphi)$, then any similar trajectory-mode pair (x', θ') of system (5.2) (i.e. (x', θ') such that $(x', \theta') \sim_0 (x, \theta)$) satisfies φ. This formula is difficult to satisfy because control inputs do not affect knowledge at time $t = 0$. Control inputs can impact the satisfaction of a formula the knowledge operator when other temporal operators are included. An example of such a formula is discussed at the end of this section and is referred to as the learn, then reach formula template.

Proposition 15. *Consider an uncertain linear system* (5.2) *and a state-based proposition* p*. The system under controller* γ *satisfies the proposition* $\mathcal{K}p$ *if and only if the system satisfies* p*.*

Proof Sketch. This is because the state of the system is directly observed and thus the system "knows" if it satisfies the formula immediately. □

Proposition 16. *Consider an uncertain linear system* (5.2) *and a model-based proposition* p*. The system under controller* γ *satisfies the proposition* $\mathcal{K}p$ *if and only if* $\Theta_p = \Theta$*.*

5.3.4.1 Satisfying the Learn, then Reach Formula

The formula that we call the learn, then reach formula is:

$$\varphi = \bigcirc^j(\mathcal{K}p_1 \implies \bigcirc^k p_2).$$

This more complicated formula uses the $\mathcal{K}$ operator as a precondition in an implies statement. Thus, this statement only is applied to trajectories that satisfy $\mathcal{K}p_1$ at time j and no other trajectories. We can extract this set of trajectories using the following Lemma:

Lemma 9. *Consider an uncertain linear system* (5.2) *and model-based proposition* p *with* $\widetilde{\Theta}_p \subseteq \Theta$. *For such a system and formula, the following sets are equivalent:*

$$\{(x,\theta) \mid (x,\theta,0) \models \bigcirc^j \mathcal{K}p\} = \bigcup_{\mathbf{m}\in\mathcal{M}^{**} \text{ s.t. } \mathbf{m}_{j+1}\subseteq\widetilde{\Theta}} \mathcal{C}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \times \mathbf{m}_{T+1} \tag{5.8}$$

where the consistency set definition is in Definition 10.

Proof. Note that, by definition of $\sim_k$, the set of trajectory-mode pairs (x',θ') that satisfy $(x,\theta) \sim_k (x',\theta')$ is $\mathcal{C}(\mathbf{m},\gamma|_{\mathbf{m}}) \times \mathbf{m}_{k+1}$ where $\mathbf{m}_{1:k+1} = \mu(x(0:k), u(0:k-1))$. If the initial trajectory-mode pair (x,θ) is chosen properly, then $\mathbf{m}_{j+1} \subseteq \widetilde{\Theta}_p$ and the set of related trajectories is all guaranteed to satisfy $\bigcirc^j\mathcal{K}p$. Thus, the formula $\bigcirc^j\mathcal{K}p$ is satisfied for any trajectory (x,θ) such that $\mu(j) \subseteq \widetilde{\Theta}_p$. Thus, the equality (5.8) holds.

□

With this result in mind, we can present the following result for finding solutions to Problem 3 with the formula $\bigcirc^j(\mathcal{K}p_1 \implies \bigcirc^k p_2)$.

Proposition 17. *Consider an uncertain linear system* (5.2)*, a model-based atomic proposition* p_1 *and a state-based atomic proposition* p_2. *The system under controller* γ *satisfies the proposition* $\bigcirc^j(\mathcal{K}p_1 \implies \bigcirc^k p_2)$ *if for all* $\mathbf{m} \in \mathcal{M}^{**}$ *such that* $\mathbf{m}_j \subseteq \widetilde{\Theta}_{p_1}$*:*

$$R_{k+j}\mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{S}_{p_2}. \tag{5.9}$$

Proof. The precondition of the implies indicates that the formula is satisfied if all trajectory-mode pairs (x,θ) that satisfy $\bigcirc^j\mathcal{K}p_1$ also satisfy $\bigcirc^{j+k}p_2$. The set of trajectory-mode pairs (x,θ) that satisfy $\bigcirc^j\mathcal{K}p_1$ is given in Lemma 9. Furthermore, using the properties of consistency sets from Chapter 4, the set satisfies

$$\bigcup_{\mathbf{m}\in\mathcal{M}^{**} \text{ s.t. } \mathbf{m}_i\subseteq\widetilde{\Theta}} \mathcal{C}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \bigcup_{\mathbf{m}\in\mathcal{M}^{**} \text{ s.t. } \mathbf{m}_i\subseteq\widetilde{\Theta}} \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}).$$

Therefore, if (5.9) is satisfied for all $\mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$ where $\mathbf{m} \in \mathcal{M}^{**}$ then it also holds for all $\mathcal{C}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}})$. Then, applying the results from Proposition 12, we can verify that all such trajectory-mode pairs satisfy the final formula $\bigcirc^{j+k} p_2$. □

5.4 Satisfying More Complex Formulas

In the last section, we discussed how to verify whether or not a system (5.2) with controller $\gamma^{(\mathcal{M}^*)}$ satisfied some basic formula templates. Each of these templates typically involved the application of one operator at a time. In most specifications used in the real world, however more than one operator needs to be used in order to completely specify the task. We provide examples in this section of two realistic formula templates that would be of interest to adaptive controller designers and how to design them using the tools discussed in this section.

Proposition 18. *Consider an uncertain linear system* (5.2) *and a state-based atomic proposition* p_T *that is satisfied when the system reaches* $\mathcal{X}_T$*. A solution to the optimization* (4.12) *defines an adaptive controller* γ *that guarantees the formula* $\varphi = \bigcirc^T p_T$ *is satisfied for the system* (5.2)*.*

Proof. A solution to (4.12) defines an adaptive controller that satisfies Problem 2 for $\mathcal{X}_T$. The finite-time reachability task in Problem 2 is exactly $\bigcirc^T p_T$ where p_T is a state-based atomic proposition with associated polytope $\mathcal{X}_T$. Thus, the solution to (4.12) determines a solution to Problem 3 for the finite-time reachability specification $\bigcirc^T p_T$. □

We can slightly modify the proposition above to define controllers that satisfy more complicated formulas:

Proposition 19. *Consider an uncertain linear system* (5.2) *with* $|\Theta| = 2$ *and a target time horizon* T*. Assume that the target time horizon* T *is large enough that modes can be distinguished. Also, consider the following atomic propositions:*

- *model-based propositions* $p_K^{(1)}$ *and* $p_K^{(2)}$ *(with associated sets* $\{\theta_1\}$ *and* $\{\theta_2\}$*, respectively), along with*
- *state-based propositions* $p_T^{(1)}$ *and* $p_T^{(2)}$ *(with associated sets* $\mathcal{X}_T^{(1)}$ *and* $\mathcal{X}_T^{(2)}$*, respectively).*

The system is guaranteed to satisfy the formula:

$$\varphi = \left[\Diamond(\mathcal{K} p_K^{(1)} \implies \bigcirc \Diamond p_T^{(1)})\right] \wedge \left[\Diamond(\mathcal{K} p_K^{(2)} \implies \bigcirc \Diamond p_T^{(2)})\right]$$

if the following optimization problem is feasible

$$
\begin{aligned}
&\textit{Find } \left\{\mathbf{K}^{(\mathbf{m})}, \mathbf{k}^{(\mathbf{m})}\right\}_{\mathbf{m}\in\mathcal{M}^*} \\
&\quad \textit{s.t. } \forall \mathbf{m} \in \mathcal{M}^* : \\
&\qquad \tilde{\gamma}|_{\mathbf{m}} \textit{ defined as in } (4.9) && \text{(5.10a)} \\
&\qquad \tilde{\gamma}|_{\mathbf{m}}(t) \in \mathcal{U} \qquad \forall t \in [0, T-1] && \text{(5.10b)} \\
&\quad \forall \mathbf{m} \in \{\mathbf{m} \in \mathcal{M}^{**} \mid \exists \tau \in [1, T-1], \mathbf{m}_\tau = \{1\}\} : \\
&\qquad R_T \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{X}_T^{(1)} && \text{(5.10c)} \\
&\quad \forall \mathbf{m} \in \{\mathbf{m} \in \mathcal{M}^{**} \mid \exists \tau \in [1, T-1], \mathbf{m}_\tau = \{2\}\} : \\
&\qquad R_T \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{X}_T^{(2)} && \text{(5.10d)} \\
&\quad \forall \mathbf{m}, \mathbf{m}' \in \mathcal{M}^* : \\
&\qquad \mathbf{m}_{[1:t]} = \mathbf{m}'_{[1:t]} \implies K^{(\mathbf{m}_{[1:t]})} = K^{(\mathbf{m}'_{[1:t]})}, \quad k^{(\mathbf{m}_{[1:t]})} = k^{(\mathbf{m}'_{[1:t]})}. && \text{(5.10e)}
\end{aligned}
$$

Proof. The proof of this follows from the proof of Proposition 18 and the definition of the KLTL operators in the previous section.

First, note that the constraints (5.10a), (5.10b), and (5.10e) define an adaptive controller that incorporates an exploration-exploitation profile $\mathcal{M}^*$. The adaptive controller considers $\mathcal{M}^*$ so the set of trajectory-mode pairs $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ is properly defined.

Let us now analyze the other constraints to identify whether or not φ is satisfied for all trajectory-mode pairs $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$. First, note that this formula is composed of two parts which are identical in structure. For that reason, we will focus on the first part of the formula:

$$\Diamond(\mathcal{K} p_K^{(1)} \implies \Diamond p_T^{(1)}).$$

This part of the formula is satisfied if for any time j at which $\mathcal{K} p_K^{(1)}$ is satisfied, then at a later time $j + k$ at the state-based proposition $p_T^{(1)}$ is satisfied. This can be written mathematically as saying that $\Diamond(\mathcal{K} p_K^{(1)} \implies \Diamond p_T^{(1)})$ if for any j and k such that the learn, then reach specification $\bigcirc^j(\mathcal{K} p_1 \implies \bigcirc^k p_2)$ is satisfied.

The "learn then reach" specification is satisfied for j and $k = T - j$ if for all $\mathbf{m} \in \mathcal{M}^{**}$ such that $\mathbf{m}_j \subseteq \{\theta_1\}$:

$$R_{k+j} \mathcal{R}(\mathbf{m}, \tilde{\gamma}|_{\mathbf{m}}) \subseteq \mathcal{S}_{p_2}$$

according to Proposition 17. This constraint is satisfied for $j < T$ if (5.10) is satisfied (due to constraint (5.10d)).

All trajectory-mode pairs in $\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)})$ satisfy the formula φ. optimization (5.10) guarantees formula φ Constraint (5.10c) $\square$

5.5 Results

In this section, the propositions developed in Section 5.4 are applied to two illustrative examples. In the first example, the system is reduced to a point where the implementation of Proposition 19 can be done by observation. In this example, we show that the system satisfies the constraints of (5.10) and that it also satisfies the associated formula template. In the second example, a drone system is presented with a task that is a learn, then reach formula. For this system, the bilinear optimization problem is solved quickly and an adaptive controller is produced that properly adapts to a faulty drone dynamics.

5.5.1 Similar Rotation System

The first example in this section will consider will not include control. When this is the case, the number of trajectories that can be produced is a bit simpler and the analysis can be done without software.

Consider a similar rotation system similar to that of Section 4.6.1. Similarly, we consider the following two mode system (i.e. $|\Theta| = 2$), where $(A_1, B_1, \mathcal{W}_1) = (R_{\pi/4}, 0, \{0\})$ and $(A_2, B, \mathcal{W}_2) = (R_{-\pi/4}, 0, \{0\})\}$. Note that the two systems are nearly identical except for the state update matrix A which is either a counterclockwise rotation by $\frac{\pi}{4}$ radians ($R_{\pi/4}$) or a counterclockwise rotation by $-\frac{\pi}{4}$ radians ($R_{-\pi/4}$).

The initial state set of the system is $\mathcal{X}_0 = \{\begin{bmatrix}-1 & 0\end{bmatrix}^\top\}$.

Now consider the following formula:

$$\varphi_1 = \left[\Diamond(\mathcal{K}p_K^{(1)} \implies \bigcirc\Diamond p_T^{(1)})\right] \wedge \left[\Diamond(\mathcal{K}p_K^{(2)} \implies \bigcirc\Diamond p_T^{(2)})\right]$$

where

- $p_K^{(1)}$ and $p_K^{(2)}$ are model-based propositions with associated sets $\widetilde{\Theta}^{(1)} = \{1\}$ and $\widetilde{\Theta}^{(2)} = \{2\}$, respectively, and
- $p_T^{(1)}$ and $p_T^{(2)}$ are state-based propositions with associated sets $\mathcal{X}^{(1)} = [-0.5, 0.5] \times [-1.5, -0.5]$ and $\mathcal{X}^{(2)} = [-0.5, 0.5] \times [0.5, 1.5]$.

For this problem, one only needs to evaluate the trajectory-mode pairs (x, θ) where $x = x(0:2)$ in order to determine that φ is satisfied. Note the following reachable behavior sets:

- At $t = 0$: $\mathcal{R}(\{\Theta\}) = \mathcal{X}_0 = \left\{\begin{bmatrix}-1 & 0\end{bmatrix}^\top\right\}$
- At $t = 1$:

- $\mathcal{R}(\Theta\{1\}) = \left\{ (\begin{bmatrix} -1 & 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}^\top, 0) \right\}$
- $\mathcal{R}(\Theta\{2\}) = \left\{ (\begin{bmatrix} -1 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}^\top, 0) \right\}$
- $\mathcal{R}(\Theta\Theta) = \emptyset$

- At $t = 2$:
  - $\mathcal{R}(\Theta\{1\}\{1\}) = \left\{ (\begin{bmatrix} -1 & 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & -1 \end{bmatrix}^\top, \begin{bmatrix} 0 & 0 \end{bmatrix}^\top) \right\}$
  - $\mathcal{R}(\Theta\{2\}\{2\}) = \left\{ (\begin{bmatrix} -1 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 1 \end{bmatrix}^\top, \begin{bmatrix} 0 & 0 \end{bmatrix}^\top) \right\}$
  - $\mathcal{R}(\Theta\Theta\{1\}) = \emptyset$
  - $\mathcal{R}(\Theta\Theta\{2\}) = \emptyset$
  - $\mathcal{R}(\Theta\Theta\Theta) = \emptyset$

Thus, we know that the set of all trajectory-mode pairs of length 2 is defined as

$$\mathcal{T}(\Theta, \gamma^{(\mathcal{M}^*)}) = \left\{ (\begin{bmatrix} -1 & 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & -1 \end{bmatrix}^\top, \{1\}), (\begin{bmatrix} -1 & 0 & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 1 \end{bmatrix}^\top, \{2\}) \right\}$$

for this system. By observation, the following properties hold for this set of trajectory-mode pairs:

- The first trajectory-mode pair satisfies $\bigcirc^j \mathcal{K} p_K^{(1)}$ for $j \geq 1$ (i.e. $(x_1, \theta_1, 0) \models \bigcirc^j \mathcal{K} p_K^{(1)} \; \forall j \geq 1$) . Similarly, the second trajectory-mode pair satisfies $\bigcirc^j \mathcal{K} p_K^{(2)}$ (i.e. $(x_2, \theta_2, 0) \models \bigcirc^j \mathcal{K} p_K^{(2)} \; \forall j \geq 1$). This is true because the number of trajectory-mode pairs that satisfy each $\bigcirc^j \mathcal{K} p_K^{(k)}$ for $j \geq 1$ and $k \in \{1, 2\}$ is one for each assignment of j and k.
- The first trajectory-mode pair satisfies $\bigcirc^2 p_T^{(1)}$ (i.e. $(x_1, \theta_1, 0) \models \bigcirc^2 p_T^{(1)}$) and the second trajectory-mode pair satisfies $\bigcirc^2 p_T^{(2)}$ (i.e. $(x_2, \theta_2, 0) \models \bigcirc^2 p_T^{(2)}$).

The two observations above are enough to conclude that φ is satisfied. Now, to verify that Proposition 19 correctly detects this, we evaluate each of the constraints in (5.10). Because we assume that there is no input (i.e. $u(k) = 0 \; \forall k$), we assume that all input constraints are trivially satisfied (i.e. (5.10a), (5.10b) and (5.10e) may be ignored).

Consider constraint (5.10c). This constraint needs to hold for all $\mathbf{m} \in \{\Theta\{1\}\{1\}\}$ (there is only one sequence that contains the mode $\{1\}$). Thus, constraint (5.10c) only needs to be satisfied for $\mathcal{R}(\Theta\{1\}\{1\}) = \left\{ (\begin{bmatrix} -1 & 0 & \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 & -1 \end{bmatrix}^\top, \begin{bmatrix} 0 & 0 \end{bmatrix}^\top) \right\}$. Now,

$$R_2 \mathcal{R}(\Theta\{1\}\{1\}) = \left\{ \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right\}$$

and $R_2\mathcal{R}(\Theta\{1\}\{1\}) \subseteq \mathcal{X}_T^{(1)}$.

A similar line of reasoning proves that constraint (5.10d) also holds for this example.

5.5.2 Drone with Corrupted Velocity Commands

The second example in this section will consider will include control. In this example, we consider a drone which is modeled as a single integrator (i.e. we assume that we control the drone's velocity) in the two-dimensional plane.

Consider the system as a two mode instance (i.e. $|\Theta| = 2$) of (5.2), where the state is a two-dimensional vector $x(t) = \begin{bmatrix} p_x(t) & p_y(t) \end{bmatrix}^\top$ representing the drone's position in the x- and y-axes of the plane. Let the system be represented by two tuples $(A, B_1, \mathcal{W}_1) = (I, R_0, \mathcal{W})$ and $(A, B_2, \mathcal{W}_2) = (I, R_{\pi/16}, \mathcal{W})\}$. Note that the two systems are nearly identical except for the input matrix B which is either a counterclockwise rotation by $\frac{\pi}{16}$ radians ($R_{\pi/16}$) or no rotation (I_2). This indicates that either velocity commands are corrupted (i.e. rotated by $\frac{\pi}{16}$ radians) or not. Corruption of the velocity command can be the result of damage to the drone, software errors, and much more.

The set of disturbances $\mathcal{W} = \{w \in \mathbb{R}^2 \mid \|w\| \leq 0.1\}$, the set of allowable inputs is $\mathcal{U} = \{u \in \mathbb{R}^2 \mid \|u\|_\infty \leq 0.5\}$ and the initial state set of the system is $\mathcal{X}_0 = \{\begin{bmatrix} 0 & 0 \end{bmatrix}^\top\}$.

The task for this system is as follows:

$$\varphi_1 = \left[\Diamond(\mathcal{K}p_K^{(1)} \implies \bigcirc\Diamond p_T^{(1)})\right] \wedge \left[\Diamond(\mathcal{K}p_K^{(2)} \implies \bigcirc\Diamond p_T^{(2)})\right]$$

where

- $p_K^{(1)}$ and $p_K^{(2)}$ are model-based propositions with associated sets $\widetilde{\Theta}^{(1)} = \{1\}$ and $\widetilde{\Theta}^{(2)} = \{2\}$, respectively, and
- $p_T^{(1)}$ and $p_T^{(2)}$ are state-based propositions with associated sets $\mathcal{X}_T^{(1)} = [1.2, 1.8] \times [1.0, 1.4]$ and $\mathcal{X}_T^{(2)} = [0.5, 1.1] \times [1.0, 1.4]$.

In other words, if the drone learns that its control commands are corrupted then it should reach region $\mathcal{X}_T^{(2)}$. If the drone learns that its control commands are not corrupted, then it should reach region $\mathcal{X}_T^{(1)}$.

The optimization presented in Proposition 19 can be solved to find a controller that guarantees this specification is satisfied. The optimization is solved in 1.38 seconds (with a constraint setup time of 56.34 seconds). The result of this optimization can be applied to the simulated system above for various realizations of the unknown parameters as shown in Figure 5.1 or applied on a real drone system (i.e., the Crazyflie 2.1) as is visualized in Figure 5.2.

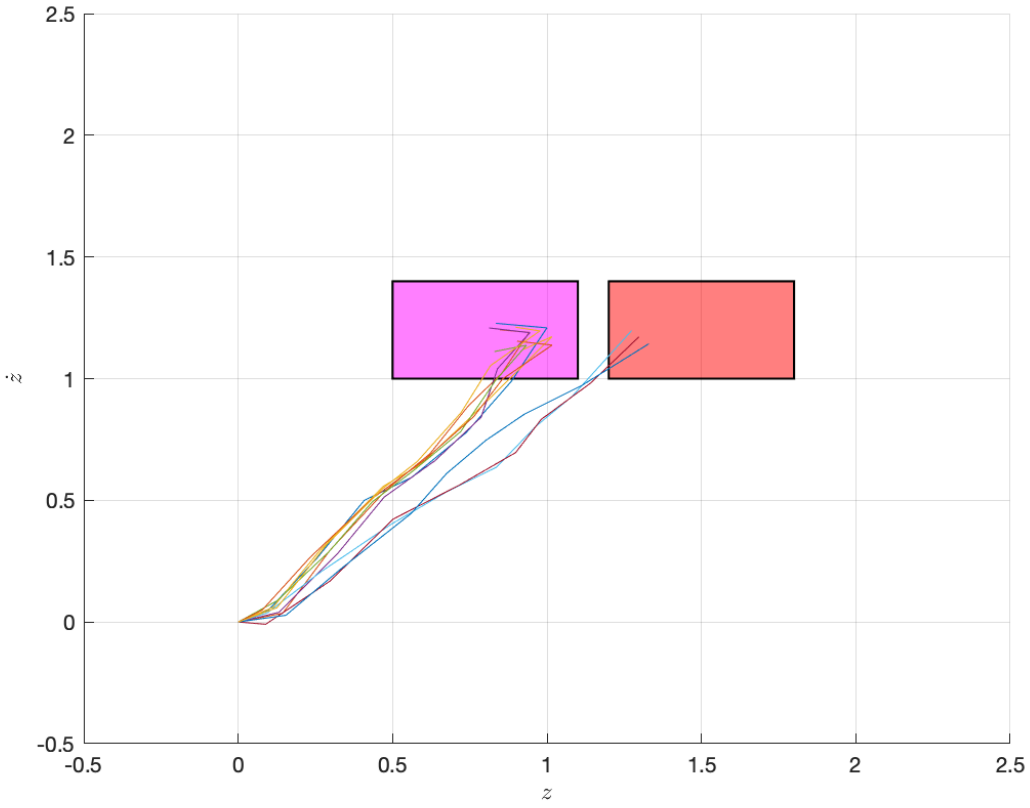


Figure 5.1: For the drone with potentially corrupted velocity commands, we design a controller using Proposition 19. The adaptive controller correctly steers the corrupted or normal system into the correct regions ($\mathcal{X}_T^{(1)}$ in red, or $\mathcal{X}_T^{(2)}$ in magenta) as is guaranteed by the proposition.

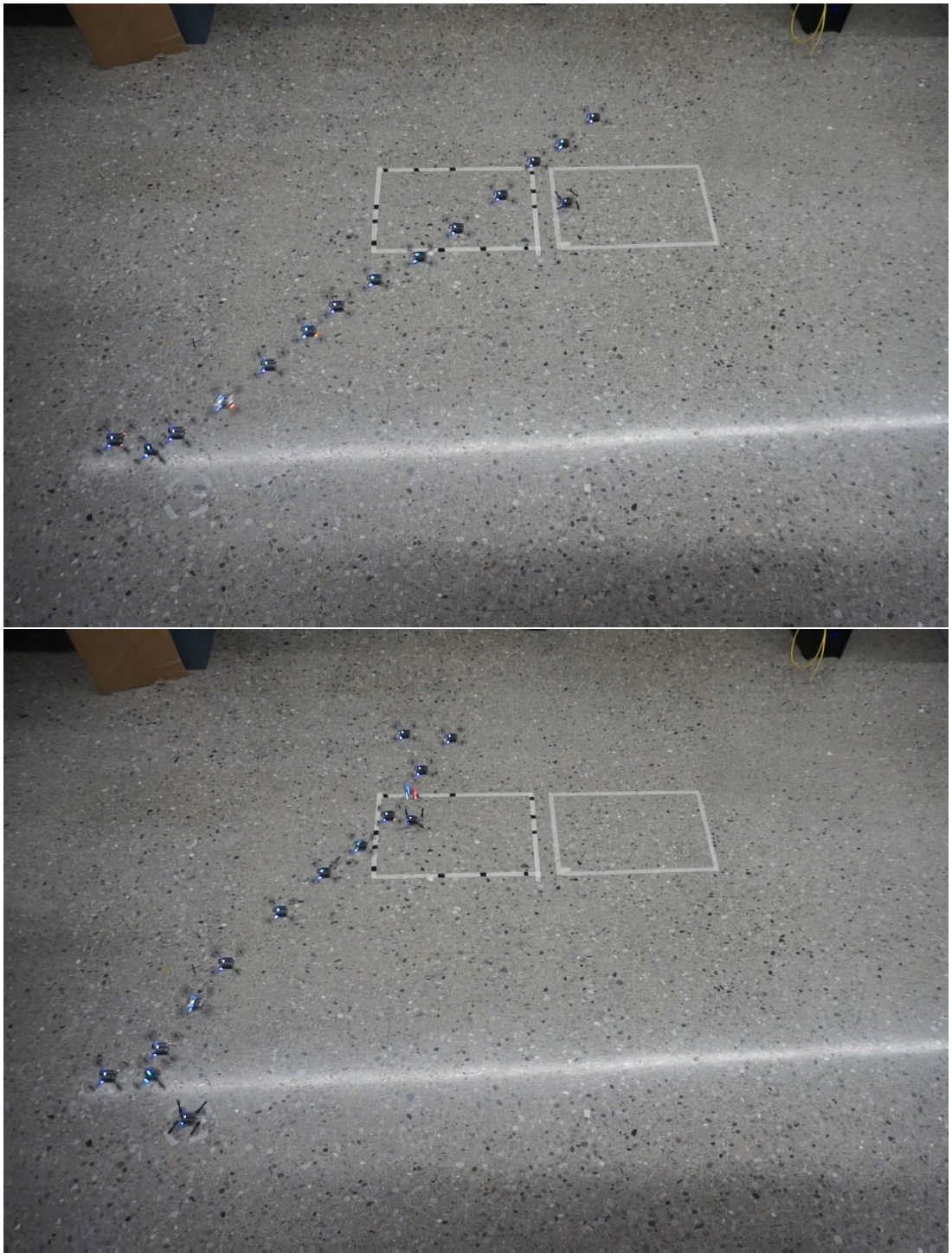


Figure 5.2: The potentially corrupted drone system from Section 5.5.2 is implemented by modifying the software of a Crazyflie drone. For the task discussed in the same subsection, we design an adaptive controller that is guaranteed to steer the drone into region 1 (marked in only masking tape) on the ground or the repair region (outlined with masking + electrical tape) on the ground. The decision of which area to land in is determined by what the controller learns over the course of the experiment.

CHAPTER 6

Intention-Aware Supervisory Control with Driving Safety Applications

6.1 Introduction

We have discussed how to handle parametric uncertainty for tasks with a known deadline in the past two chapters (Chapters 4 and 5). The assumption that a fixed deadline is known a priori is a bit strong. In many situations, we are more concerned with task completion (e.g., reaching a target state safely) than the time it takes to do so.

An example of such a situation is an autonomous vehicle on a highway that wishes to overtake the vehicle in front of it. There typically is no deadline to execute the maneuver. What is more important is that the autonomous vehicle completes the maneuver while avoiding collisions and obeying traffic laws (e.g., speed limits). This problem can not be solved by the finite time horizon methods of Chapters 4 and 5 because a time horizon typically cannot be determined a priori. Instead, we must develop solutions that can operate for an unlimited amount of time before achieving the task at hand. This chapter discusses how to develop such solutions and returns to the motivating example (overtake scenario) frequently.

6.2 Problem Statement and Architecture

We start by describing the abstract problem that we are interested in solving. Let PWA system f of the form (2.6) represent the interaction of an ego agent with other agents where $x \in \mathcal{X}$ is the combined states of all agents, control input $u(t) = [u_e(t)^\top \quad u_o(t)^\top]^\top \in \mathcal{U}_e \times \mathcal{U}_o$ is partitioned into two parts where *ego input* $u_e(t)$ is controlled by the ego agent and *external input* $u_o(t)$ is controlled by all other agents, and *disturbance* $w(t) \in \mathcal{W}$ captures model uncertainty. We assume that the other agents behave according to a fixed *intention model* $I_{i^*} : \mathcal{X} \to \mathcal{P}_{\geq 1}(\mathcal{U}_o)$, which is a set valued mapping that returns a set of external control inputs given a state. That is, if the system is

currently at x, then the external control input u_o is restricted such that $u_o \in I_{i^*}(x) \subseteq \mathcal{U}_o$. While the actual specific intention model I_{i^*} is unbeknownst to the ego agent, a finite set $\mathcal{I} = \{I_1, \ldots, I_n\}$ of intention models is known a priori such that $I_{i^*} \in \mathcal{I}$. There are two sources of uncertainty from the perspective of the ego agent: one due to the fact that i^* is not known, another due to I_{i^*} being a set-valued map, capturing the variability within a specific intention. With a slight abuse of notation, we define $\mathcal{I}(x) \doteq \bigcup_{I \in \mathcal{I}} I(x)$, the set of all possible external control inputs that the ego agent presumes, given the current state x.

Our goal is to design a *supervisor module*, which restricts the inputs of the ego agent when needed, to ensure that the states of the system remain indefinitely in a safe set $\mathcal{X}_{safe} \subseteq \mathcal{X}$. However, due to the dynamics and disturbances in (2.6), we can only enforce that the system stays in a subset of $\mathcal{X}_{safe}$, which is an RCIS that is computed according to Section 2.5.1.

Let us define a supervisor module before stating the problem of interest formally.

Definition 15. *Given a system in the form of* (2.6)*, a set of intention models $\mathcal{I}$, and a safe set $\mathcal{X}_{safe}$, a supervisor module*

$$\mathcal{S}_{\mathcal{I}} : \mathcal{X}_{safe} \mapsto \mathcal{P}(\mathcal{U}_e) \tag{6.1}$$

*takes a state measurement x and outputs a set $\mathcal{S}_{\mathcal{I}}(x) \subseteq \mathcal{U}_e$ of **admissible** ego inputs such that the admissible inputs $u_e(t) \in \mathcal{S}_{\mathcal{I}}(x)$ enforce the system to indefinitely remain in the safe set regardless of the external input and the disturbance, i.e., $\mathcal{S}_{\mathcal{I}}(x(t)) \neq \emptyset \implies \mathcal{S}_{\mathcal{I}}(x(t+1)) \neq \emptyset$ for all $u_e(t) \in \mathcal{S}_{\mathcal{I}}(x(t))$, $u_o(t) \in \mathcal{I}(x(t))$ and $w(t) \in \mathcal{W}$ where $x(t+1) = f(x(t), u(t), w(t))$.*

A supervisor's goal is to keep the system in the safe set. If the admissible ego input safe is empty, the system must either be in an unsafe state, or it is not possible for the ego agent to guarantee that the system stays in the safe set indefinitely. That is, there exists a finite sequence of external inputs, over which the ego agent has no control, and a finite sequence of disturbances that would eventually steer the system into an unsafe state, regardless of the ego input. On the other hand, the above definition implies that the set $\mathcal{C} = \{x \in \mathcal{X}_{safe} \mid \mathcal{S}_{\mathcal{I}}(x) \neq \emptyset\}$ is an RCIS. Given two supervisors $\mathcal{S}^1_{\mathcal{I}}$ and $\mathcal{S}^2_{\mathcal{I}}$, we say $\mathcal{S}^1_{\mathcal{I}}$ is more *permissive* if $\mathcal{S}^2_{\mathcal{I}}(x) \subseteq \mathcal{S}^1_{\mathcal{I}}(x)$ for all $x \in \mathcal{X}_{safe}$. The key insight in this chapter is that, intuitively, a smaller set of intention models should lead to more permissive supervisors. That is, if $\tilde{\mathcal{I}} \subset \mathcal{I}$, for any $\mathcal{S}_{\mathcal{I}}$, there exists $\mathcal{S}_{\tilde{\mathcal{I}}}$ that is more permissive.

We now formally define the problem we are interested in solving and provide a solution method.

Problem 4. *Let a PWA system f in the form of* (2.6)*, a set of intention models $\mathcal{I}$ and a safe set $\mathcal{X}_{safe} \subset \mathcal{X}$ be given. Find a supervisor module $\mathcal{S}_{\mathcal{I}}$ as in Definition 15 and a set of initial states $\mathcal{C} \subseteq \mathcal{X}_{safe}$ such that any trajectory that starts from an arbitrary state $x(0) \in \mathcal{C}$ is guaranteed to indefinitely remain in $\mathcal{C}$ as long as the control input u_e is chosen from the set of admissible inputs, i.e., $u_e(t) \in \mathcal{S}_{\mathcal{I}}(x(t))$ for all t.*

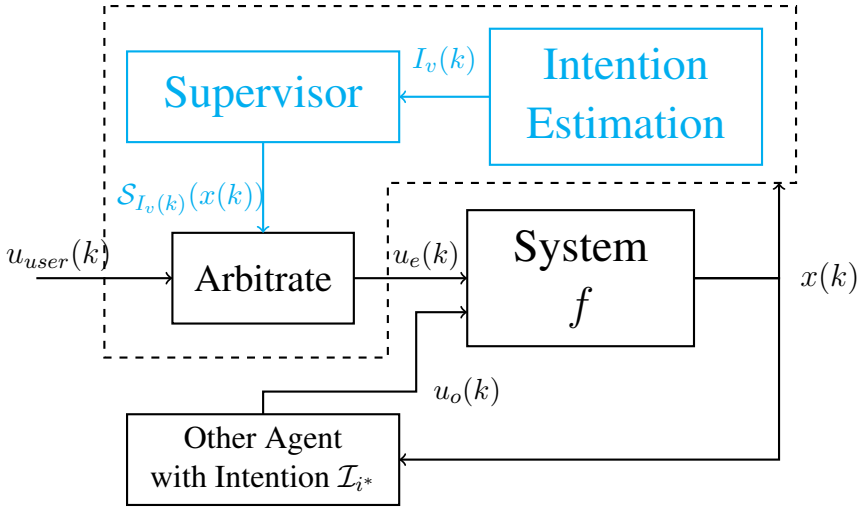


Figure 6.1: Guardian architecture proposed to solve Problem 4.

Problem 4 can be solved using existing methods such as [111]. However, as previously mentioned, uncertainty in the external input u_o is larger from the perspective of the ego agent since the intention of other agents is unbeknownst to the ego agent a priori. As a result, the supervisor $\mathcal{S}_{\mathcal{I}}$ must be designed so that it would guarantee safety for any intention model, which is conservative and not desirable. In reality, the ego agent could observe the other agents and decrease the uncertainty by invalidating intention models that are not consistent with the observed external inputs. Inspired by this observation, we propose a less conservative *guardian architecture*, which is illustrated in Figure 6.1, to solve Problem 4, that consists of a *library of supervisor modules* and an *intention estimation module*:

Definition 16. *An intention estimation module*

$$\mathcal{E} : (\mathcal{X} \times \mathcal{U}_e)^* \mapsto \mathcal{P}_{\geq 1}(\mathcal{I})$$

maps any state-ego input trajectory $\mathbf{xu_e^t} = \{(x(0), u_e(0)), \ldots, (x(t), u_e(t))\}$*, to a non-empty subset* $\mathcal{I}_v(t+1) = \mathcal{E}(\mathbf{xu_e^t}) \subseteq \mathcal{I}$ *of* ***valid intentions*** *such that there exist an external control input* $u_o(k)$ *and disturbance* $w(k)$ *that satisfy the following for all* $k \in \{0, \ldots, t\}$*:*

$$\begin{aligned} x(k+1) &= f\left(x(k), [u_e(k) \;\; u_o(k)], w(k)\right), \text{ and} \\ u_o(k) &\in I_i(x(k)) \text{ for all } I_i \in \mathcal{I}_v(t+1). \end{aligned} \tag{6.2}$$

An estimation module indicates the set of intention models that are *valid* by invalidating the intentions that are inconsistent with a given state-input pair. Since the true intention I_{i^*} of the other agents is assumed to be constant over time, it is always included in the set of valid intentions, i.e., $I_{i^*} \in \mathcal{E}(\mathbf{xu_e}), \forall \mathbf{xu_e} \in (\mathcal{X} \times \mathcal{U}_e)^*$. Note that, lengthening the state-input pair can only refine

the set of valid intentions, thus, intention estimation over time is a monotonically non-increasing set for a system.

Given an instance of Problem 4, a more permissive supervisor can be designed by leveraging the information gained from such an intention estimation module. To do so, we compute a library of supervisors $\{\mathcal{S}_{\mathcal{I}}, \mathcal{S}_{I_1}, \mathcal{S}_{I_2}, \ldots, \mathcal{S}_{I_n}\}$[1]. As the notation indicates, we design a supervisor $\mathcal{S}_{I_i}$ for each possible intention model I_i, together with an intention-agnostic supervisor $\mathcal{S}_{\mathcal{I}}$. During run-time, we switch between the supervisors, depending on the output of the intention estimation module $\mathcal{E}$. This approach enables us to change the level of permissiveness depending on the observations, while still guaranteeing safety. That is, we use the supervisor module $\mathcal{S}_{\mathcal{I}}$ when the true intention of the other agents is not yet known, and guarantee that the system remains in the safe set. Once the true intention I_{i^*} is revealed by the estimation module $\mathcal{E}$, we switch to the corresponding supervisor $\mathcal{S}_{I_{i^*}}$, which is more permissive. As a result, the overall architecture is less conservative.

6.3 The Scenario and System Models

To illustrate the concepts that are presented in this chapter, we choose a simple autonomous driving scenario and explain the solution method referring to this scenario. However, the concepts we propose in this chapter apply to the general framework explained in Section 6.2.

Imagine two vehicles moving on a straight road with two lanes as illustrated in Fig 6.2. One of these vehicles, the *ego vehicle*, is controllable through u_e and can move both in lateral and longitudinal directions. The other vehicle is called the *lead vehicle* and its longitudinal motion is controlled by a fixed intention model chosen from a set of intention models. Intention models are assumed to react to the ego vehicle when the distance between the cars is less than some threshold. As stated earlier, while this set of intention models is known to the ego vehicle, the specific intention model that controls the lead vehicle is not. We assume that the lead vehicle has no lateral motion and always drives along the center of the right lane. The safety requirement for the ego vehicle is to keep a minimum safe distance between the vehicles, in both the longitudinal and the lateral directions.

We now provide dynamics that captures the aforementioned scenario and formally define the safety requirements.

[1] An even more permissive design can be achieved if we compute a supervisor for each subset of intentions, i.e., compute $\mathcal{S}_{\mathcal{I}_v}$ for each $\mathcal{I}_v \in \mathcal{P}_{\geq 1}(\mathcal{I})$. However, such an approach would be computationally more expensive as a trade-off.

6.3.1 Dynamics

The vehicles are treated as point masses, and their motion is modeled as follows:

$$\begin{aligned} v_{e,x}(k+1) &= v_{e,x}(k) + (a_{e,x}(k) - b_e v_{e,x}(k))\Delta t + w_{e,x}(k)\Delta t, \\ y_e(k+1) &= y_e(k) + v_{e,y}(k)\Delta t + w_{e,y}(k)\Delta t, \\ v_{L,x}(k+1) &= v_{L,x}(k) + (a_{L,x}(k) - b_L v_{L,x}(k))\Delta t + w_{L,x}(k)\Delta t, \end{aligned} \tag{6.3}$$

where Δt $(= 0.1)$ is the sampling time, $v_{e,x}$ is the longitudinal velocity of the ego vehicle, y_e is the lateral displacement of the ego vehicle with respect to the center of the right lane, and $v_{L,x}$ represents the longitudinal velocity of the lead vehicle. The ego vehicle is controlled through its longitudinal acceleration $a_{e,x}$ and lateral velocity $v_{e,y}$. The longitudinal acceleration of the lead vehicle, $a_{L,x}$, depends on the intention and is treated as external disturbance. Terms b_e $(= 0.1)$ and b_L $(= 0.1)$ are drag coefficients and $w_{e,x}(k) \in [-.15, .15]$, $w_{e,y}(k) \in [-.09, .09]$ and $w_{L,x}(k) \in [-.05, .05]$ are process noises. The relative longitudinal distance between the two vehicles is denoted by h and evolves according to the following:

$$h(k+1) = h(k) + (v_{L,x}(k) - v_{e,x}(k))\Delta t. \tag{6.4}$$

As indicated by (6.4), positive values for h imply that the ego vehicle is behind the lead vehicle.

We now define the vectors $x(k) = [v_{e,x}(k),\ y_e(k),\ h(k),\ v_{L,x}(k)]^\top$, $u_e(k) = [a_{e,x}(k),\ v_{e,y}(k)]^\top$, $u_o(k) = [a_{L,x}(k)]$, $u(k) = [u_e(k),\ u_o(k)]^\top$, $w = [w_{e,x}(k), w_{e,y}(k), w_{L,x}(k)]^\top$, and combine (6.3) and (6.4) in to the form (2.7), where $\mathcal{X} = [v_{e,x}^{min}, v_{e,x}^{max}] \times [y_e^{min}, y_e^{max}] \times \mathbb{R} \times [v_{L,x}^{min}, v_{L,x}^{max}]$.

6.3.2 Intention Models

We consider two driver intentions, denoted by $\mathcal{I} \in \{I_a, I_c\}$, corresponding to Aggressive and Cautious drivers[2]. Here, these drivers react to the ego vehicle only when it is close enough, that is, when the absolute value of the longitudinal distance is less than some threshold. This area is called the *reaction zone* and is illustrated in Fig. 6.2. When the ego vehicle is inside the reaction zone, the external input u_o is determined by an affine state-feedback policy; otherwise only a bound on the velocity is imposed in the choice of u_o. Having the reaction zone captures two properties: (i) since intentions are feedback policies in our setup, it is reasonable to assume feedback occurs when the vehicles are in the vicinity of each other, (ii) fixed intention assumption is automatically relaxed to intention being unchanged only within the reaction zone as outside the reaction zone the

[2]We choose two intentions to clearly illustrate these concepts and stress to the reader that our framework is general enough to incorporate as many intention models as available.

Table 6.1: Parameters in intention models

$a_{L,x}^{max}$	$3m/s^2$	$a_{L,x}^{min}$	$3m/s^2$	w_{Δ}^{max}	0.1	w_{Δ}^{min}	0.1
$v_{L,x}^{min}$	$0m/s$	$v_{L,x}^{max}$	$33.5m/s$	K_{des}	1	K_A	[1 0 0 -1]
K_C	[0 -0.1 0.1 -0.01]	k_c	0.01	$v_{L,x}^{\text{des}}$	30m	h_r	60 m
h_{min}	10m	$v_{e,x}^{min}$	$16m/s$	$v_{e,x}^{max}$	$36m/s$	y_e^{min}	-0.9m
y_e^{max}	2.7m						

assumptions on all vehicles are the same. In addition to the acceleration bounds captured by $\mathcal{U}_o$, we assume the lead car velocity is bounded by $v_{L,x}(k) \in [v_{L,x}^{min}, v_{L,x}^{max}]$. One thing to note is that an affine state-feedback might lead to violation of the assumed acceleration and velocity bounds. These bounds mimic the physical limitations of the vehicles, thus, it is assumed not possible to exceed them. Thus, external input u_o is saturated when needed. The resulting dynamics for each intention model can be represented as a PWA system as described by Table 6.1 with input bounds given by $\mathcal{U}_e = [-3, 3] \times [-1.8, 1.8]$ and $\mathcal{U}_o = [-3, 3]$.

6.3.2.1 Aggressive Driver

Tries to match the speed of the ego vehicle when the ego vehicle is inside the reaction zone, thus making it harder to overtake:

$$a_{L,x}(k) = \begin{cases} \max(\min(K_a x(k), \alpha_1), \alpha_2) + w_{\Delta}(k), & \text{if } |h(k)| \leq h_r, \\ \max(\min((v_{L,x}^{des} - v_{L,x}(k)), \alpha_1), \alpha_2) + w_{\Delta}(k), & \text{o.w.} \end{cases} \tag{6.5}$$

where

$$\begin{aligned} \alpha_1 &= \min\left(a_{L,x}^{max}, \frac{v_{L,x}^{max} - (1 - b_L \Delta t) v_{L,x}(k)}{\Delta t}\right) - w_{\Delta}^{max} - w_{L,x}^{max} \\ \alpha_2 &= \max\left(a_{L,x}^{min}, \frac{v_{L,x}^{min} - (1 - b_L \Delta t) v_{L,x}(k)}{\Delta t}\right) - w_{\Delta}^{min} - w_{L,x}^{min}. \end{aligned} \tag{6.6}$$

The $\min$ and $\max$ operations in (6.5) and (6.6) ensure that the acceleration and velocity bounds for the lead vehicle are always respected. Note that the action of the aggressive driver is non-deterministic due to the term $w_{\Delta}(k) \in [w_{\Delta}^{min}, w_{\Delta}^{max}]$, which captures the variability within each intention model. Due to $\min$ and $\max$ operators used, resulting dynamics $f_a = \{(f_a^j, D_a^j)\}_{j=1}^{9}$ is a PWA system with nine regions.

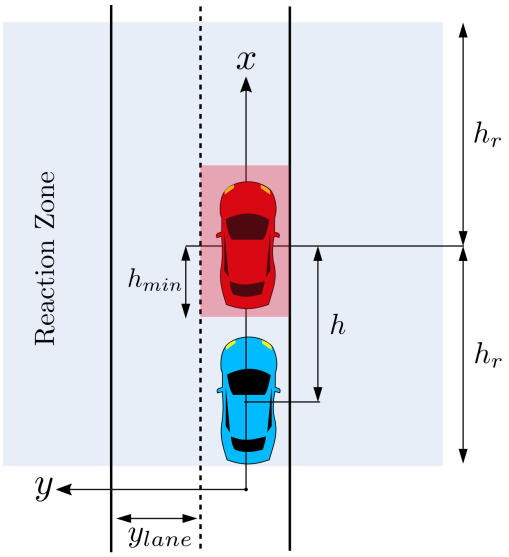


Figure 6.2: The red and blue vehicles represent the lead vehicle and ego vehicle, respectively. The red and blue boxes indicate the unsafe and the reaction zone, respectively.

6.3.2.2 Cautious Driver

Tends to maintain its desired speed and makes it easier for ego vehicle to change lane or overtake. The cautious driver is modeled as follows:

$$a_{L,x}(k) = \begin{cases} \max(\min(K_c x(k) + k_c v_{L,x}^{des}, \alpha_1), \alpha_2) + w_\Delta(k), & \text{if} |h(k)| \leq h_r, \\ \max(\min((v_{L,x}^{des} - v_{L,x}(k)), \alpha_1), \alpha_2) + w_\Delta(k), & \text{o.w.} \end{cases} \tag{6.7}$$

where α_1 and α_2 are defined as in (6.6). The resulting dynamics $f_c = \{(f_c^j, \mathcal{D}_c^j)\}_{j=1}^9$ is a PWA system with nine regions.

6.3.2.3 Bounded Velocity

When the intention of the lead vehicle is not known, we assume the worst case scenario and let $v_{L,x}(k)$ to change arbitrarily fast. That is, $v_{L,x}(k+1)$ can take any value between the lower and the upper bound, regardless of $v_{L,x}(k)$. By doing so, we capture the behavior of both intentions. We use this conservative model when the intention of the lead vehicle is not known.

6.3.3 Safety Requirements

The ego vehicle is required to keep a minimum distance between two vehicles at all times. In this case, we can represent the set $\mathcal{X}_{safe}$ of *safe* states as follows:

$$\mathcal{X}_{safe} \doteq \mathcal{X}_{safe}^1 \cap \mathcal{X}_{safe}^2 \cap \mathcal{X}_{safe}^3, \tag{6.8}$$

where $\mathcal{X}^1_{safe} \doteq \{x \in \mathcal{X} \mid |h| \geq h_{min} \text{ or } y_e \geq |y_e^{min}|\}$ capturing safe distance during takeover, $\mathcal{X}^2_{safe} \doteq \{x \in \mathcal{X} \mid y_e \in [y_e^{min}, y_e^{max}]\}$ capturing lane keeping constraints, and $\mathcal{X}^3_{safe} \doteq \{x \in \mathcal{X} \mid v_{e,x} \in [v_{e,x}^{min}, v_{e,x}^{max}]\}$ capturing the speed limits. Note that, the resulting set $\mathcal{X}_{safe}$ of safe states is not convex, but it can be represented as a union of polyhedra.

6.4 The Guardian for the Overtake Scenario

Together, a library of RCIS for each intention in 6.3.2 and an intention estimation module define the guardian for the overtake scenario. So, this section begins by discussing guarantees and methods for constructing a library of RCIS. Then, an intention estimation module is formally defined. Finally, we prove that integrating these two parts provides safety and is less conservative than previously considered models.

6.4.1 Library of RCIS

An RCIS can be constructed using any of the methods described in Section 2.5.1. Specifically, we leverage the inside-out algorithm of [111] to compute an RCIS for each intention model $I_j \in \mathcal{I}$. The reader can recall that the inside-out algorithm uses an initial RCIS and expands it to obtain a final RCIS. One fact that we can use to generate such an initial, simple RCIS is given as follows:

Proposition 20. *The set $\mathcal{C}_{left} \doteq \{x \in \mathcal{X} \mid y_e \in [0.9,\ 2.7]\}$ of states corresponding to the left lane is an RCIS for any intention.*

The proposition is stated without proof because the lead car cannot move laterally (i.e., it cannot change it's y position in the lane); thus, the proposition immediately follows from the model definition.

Given this proposition, one can apply the inside-out algorithm by setting the 'left lane' states as the initial RCIS, i.e., $\mathcal{C}_0 = \mathcal{C}_{left}$, for any of the intention models discussed in Section 6.3.2. A more involved, but helpful result that can be used to ease computation is:

Proposition 21. *Any set $\mathcal{C}_{bnd} \subseteq \mathcal{X}_{safe}$ that is a controlled invariant set for the bounded velocity model is also a controlled invariant set for the aggressive and the cautious driver intention models.*

Proof (sketch). While the acceleration of the lead vehicle $a_{L,x}$ has a specified bound for the aggressive and the cautious driver intention models, the bounded velocity model has no such bound on the lead vehicle's acceleration (i.e., the lead car may change its velocity arbitrarily fast). Thus, if it is possible to remain robustly safe in the bounded velocity model, then when the lead car's acceleration is more restricted than the bounded velocity model allows, it should be the case that the ego vehicle can remain safe in all states in $\mathcal{C}_{bnd}$. □

Thus, the previous two propositions can be used to synthesize a set of RCIS, corresponding to each of the intention models described in Section 6.3.2 $\{\mathcal{C}_{bnd}, \mathcal{C}_a, \mathcal{C}_c\}$. Specifically, one can use Proposition 20 to identify the left lane as the initial RCIS, i.e., set $\mathcal{C}_0 = \mathcal{C}_{left}$, and apply the inside-out algorithm for the bounded velocity model to obtain $\mathcal{C}_{bnd}$. After that, the resulting set $\mathcal{C}_{bnd}$ can be used as the initial RCIS for the inside-out algorithm according to Proposition 21, for each of the two intentions. Each of these RCISs induces a supervisor. For instance, for $i \in \{a, c\}$, we have $\mathcal{S}_{I_i}(x) = \{u_e \in \mathcal{U}_e \mid f_i(x, u, w) \in \mathcal{C}_i, \forall w \in \mathcal{W}, \forall u_o \in I_i(x)\}$. And, $\mathcal{S}_{\mathcal{I}}$ is defined similarly from $\mathcal{C}_{bnd}$. Moreover, these supervisors by construction satisfy the following:

Proposition 22. *$\mathcal{S}_{\mathcal{I}} \subseteq \mathcal{S}_{I_i}$ and $\mathcal{C}_{\mathcal{I}} \subseteq \mathcal{C}_{I_i}$ for $i \in \{a, c\}$.*

6.4.2 Intention Estimation

Intention estimation techniques can roughly be categorized into two categories: active [35, 46] and passive [83, 109] methods. The former assumes that the intention estimation method can modify the controller's commands. The latter, on the other hand, assumes that the intention estimation module cannot modify control signals and must perform the discrimination operation using the observations gathered by the sensors. Our guardian architecture uses a passive intention estimation scheme to allow maximal permissiveness and to avoid violation of any safe input constraints.

Given a state-input trajectory $\mathbf{xu_e^t} = \{(x(0), u_e(0)), \ldots, (x(t), u_e(t))\}$ and two intention models $\mathcal{I} = \{I_a, I_c\}$ as in Section 6.3.2, intention estimation aims to determine whether or not the state-input trajectory is consistent with model $i \in \{a, c\}$. This problem can be posed as a linear program at each time t, similar to [65]:

$$\begin{aligned} \text{find} \quad & \{u_o(k), w(k)\}_{k=\max(t-N,0)}^{t-1} \\ \text{s.t.} \quad & \text{for all } k \in \{\max(t-N,0), \ldots, t-1\} \\ & x(k+1) = f_i^j(x(k), u(k), w(k)) \ \text{ if } x(k) \in D_i^j, \\ & u_o(k) \in I_i(x(k)) \text{ and } w(k) \in \mathcal{W} \end{aligned} \tag{LP$_i^t$}$$

where N is a horizon to keep the estimator of finite memory. Note that, infeasibility of LP$_i^t$ implies

that the intention model is not I_i. Therefore, the estimator $\mathcal{E}$ is defined as:

$$\mathcal{E}(\mathbf{xu_e^t}) = \begin{cases} I_a, & \text{if } \mathcal{E}(\mathbf{xu_e^{t-1}}) = I_a \\ & \quad \text{or LP}_c^t \text{ is infeasible,} \\ I_c, & \text{if } \mathcal{E}(\mathbf{xu_e^{t-1}}) = I_c \\ & \quad \text{or LP}_a^t \text{ is infeasible,} \\ \mathcal{I}, & \text{otherwise.} \end{cases} \tag{6.9}$$

6.4.3 Putting things together

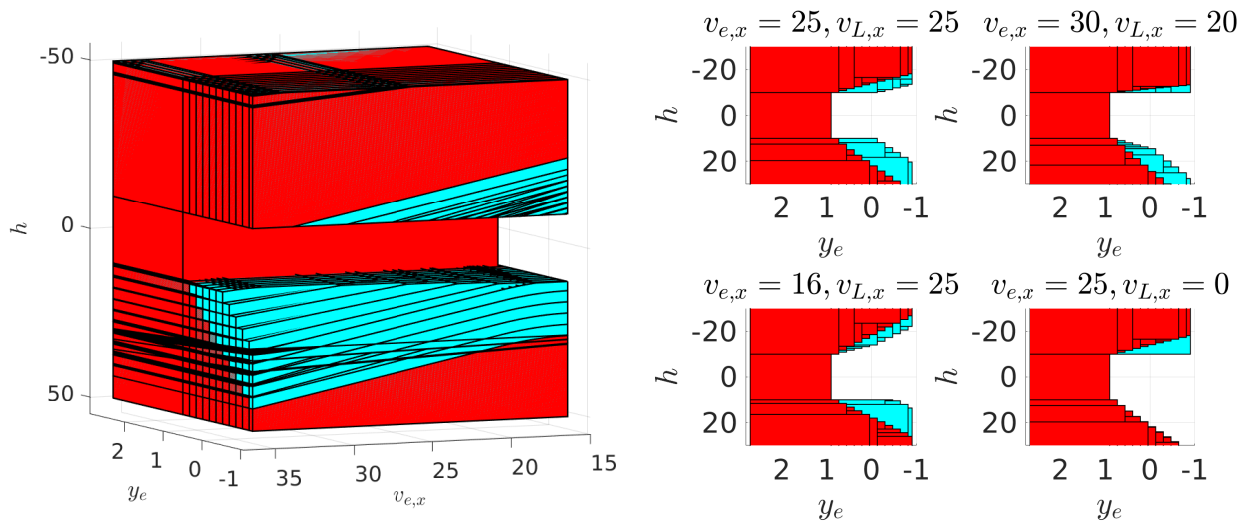
Having designed a library of RCIS and the intention estimation module, at run-time, we initialize the estimated intention for the intention-aware supervisor as the bounded velocity model, i.e., $\mathcal{I}_v(0) = \mathcal{I}$. As the intention estimation model $\mathcal{E}$ refines the valid intention models $\mathcal{I}_v$ by collecting data, the intention-aware supervisor is updated accordingly.

Theorem 4. *Assume that the intention of the other vehicle is not changing with time (i.e., I_{i^*} is constant for the driving scenario) and $I_{i^*} \in \mathcal{I} = \{I_a, I_c\}$. If $x(0) \in \mathcal{C}_{bnd}$ and $u_e(t) \in \mathcal{S}_{\mathcal{I}_v(t)}(x(t))$ for all t where $\mathcal{I}_v(t) = \mathcal{E}(\mathbf{xu_e^{t-1}})$, then we have $x(t) \in \mathcal{X}_{safe}$ for all t.*

Proof. First note that the linear program (LP_i^t) will always be feasible for $i = i^*$ as we assume I_{i^*} is constant over time. Therefore, $I_{i^*} \in \mathcal{I}_v(t)$ for all t. The intention estimation is initialized with $\mathcal{I}$. By construction, $\mathcal{S}_{\mathcal{I}}(x(0)) \neq \emptyset$ for all $x \in \mathcal{C}_{bnd}$. Now, assume that the intention estimation module never detects the correct intention (i.e., $\mathcal{I}_v(t) = \mathcal{I}$ for all t). Since $\mathcal{S}_{\mathcal{I}}(x(0)) \neq \emptyset$, it follows from Def. 15 by induction that $\mathcal{S}_{\mathcal{I}_v(t)}(x(t)) \neq \emptyset$ and $x(t) \in \mathcal{C}_{bnd} \subseteq \mathcal{X}_{safe}$ for all t. Now, assume that intention estimation module eventually reveals the true intention I_{i^*}, i.e., there exists a t^* such that $\mathcal{I}_v(t^*) = I_{i^*}$. We know that the state of the system is safe ($x(t) \in \mathcal{C}_{bnd} \subseteq \mathcal{X}_{safe}$) for $t < t^*$ by using $\mathcal{S}_{\mathcal{I}}$. Moreover, by Proposition 22, at time t^*, $\mathcal{S}_{I_{i^*}}(x(t^*)) \supseteq \mathcal{S}_{\mathcal{I}}(x(t^*)) \neq \emptyset$ and $x(t^*) \in \mathcal{C}_{bnd} \subseteq \mathcal{C}_{i^*}$. By Eq. (6.9) and the assumption on constant intention, we will have $\mathcal{I}_v(t) = I_{i^*}$ for all $t \geq t^*$. Now, again, it follows from Def. 15 by induction that $\mathcal{S}_{\mathcal{I}_v(t)}(x(t)) \neq \emptyset$ and $x(t) \in \mathcal{C}_{i^*} \subseteq \mathcal{X}_{safe}$ for all $t \geq t^*$. □

6.5 Results

In this section, we discuss the results of the proposed solution to Problem 4 for the driving scenario presented in Section 6.3. We briefly describe the tools and methods used to implement the invariant set algorithms. We then illustrate the intuitive conclusions that can be made about the RCIS and safe (admissible) input sets of various estimated intentions.



(a) Projection of the invariant set onto $(v_{e,x}, y_e, h)$ space

(b) Sliced invariant sets given the $v_{e,x}$ and $v_{L,x}$ (in m/s)

Figure 6.3: The invariant sets for the bounded velocity model (red) and the model of the cautious driver intention (red+blue, the result after 5 iterations).

6.5.1 Implementation and Experimental Setup

We use the inside-out method described in Section 2.5.1 to compute RCIS and safe input sets. We use polyhedra (or union of polyhedra) representation of sets in our algorithm, since it forms a closed class of objects under set operations such as intersection and projection. The code is implemented on top of the Multi-Parametric Toolbox 3.0 (MPT3) [70], a MATLAB toolbox with efficient implementations of polyhedra manipulations. The system dynamics, intention models and the safety requirements are as stated in Section 6.3.

6.5.2 RCIS Computation Results and Discussion

We first compute an RCIS for the bounded velocity model. The seed set for the inside-out algorithm is chosen as the left lane, i.e., $\mathcal{C}_0 = \mathcal{C}_{left}$, which is shown to be robust controlled invariant in Proposition 20. The algorithm converges in 12 iterations and the resulting RCIS is shown as the red regions in Figures 6.3a and 6.3b.

Due to Proposition 21, RCIS for the bounded velocity model is also robust controlled invariant for the other intentions. Thus, we initialize the inside-out algorithm with this new seed in the following computations. The resulting set after 5 iterations for the cautious driver intention model is shown as the union of the red and blue regions in Figures 6.3a and 6.3b. The blue region indicates the difference between the RCIS of the cautious driver and the bounded velocity model. The results show that, by estimating the intention model, we indeed have a larger invariant set.

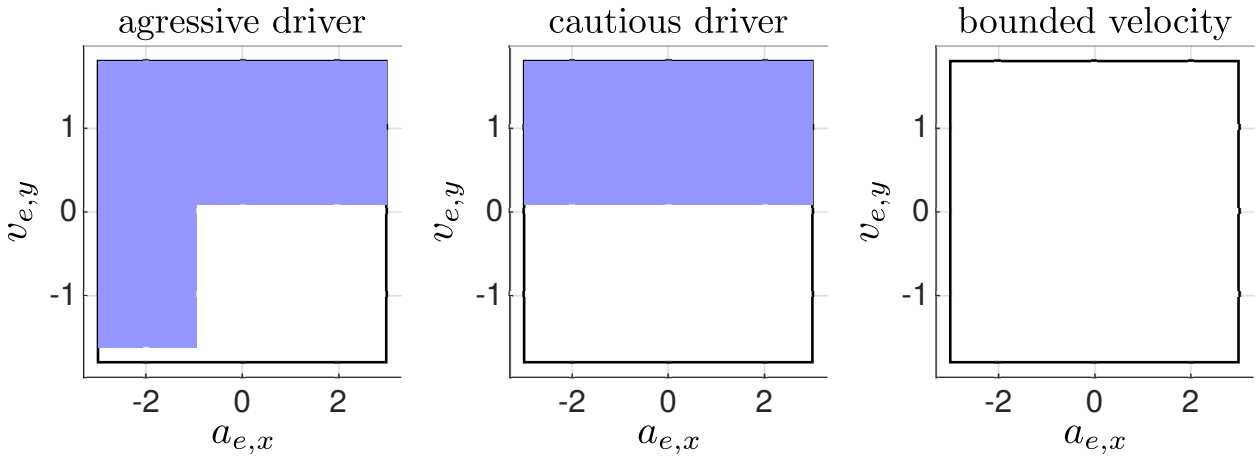


Figure 6.4: Safe inputs (blue regions) at state $[25,\ -0.297,\ 16.52,\ 20]^\top$ for aggressive driver intention (left), cautious driver intention (middle) and bounded velocity model (right).

On the other hand, RCIS obtained for the aggressive intention is almost visually indistinguishable with the invariant set for the cautious intention, but as can be shown in Figure 6.4, their sets of admissible inputs corresponding to the same state can be different.

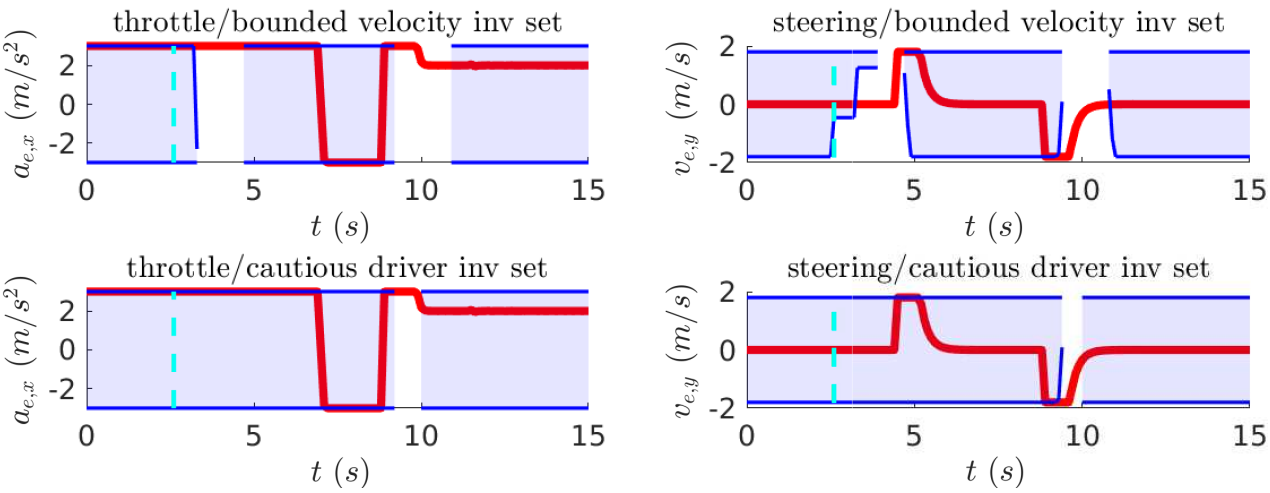
Note that, as shown in Figure 6.4, the safe input set can be non-convex. In that case, the projection to each dimension can be done in an order, according to a user defined priority. For example, speed change may be perceived as less "invasive" compared to a steering angle change from the human user perspective. In this case, projection onto the throttle input space may be preferred over the projection onto the steering input space.

6.5.3 Overtaking Simulation

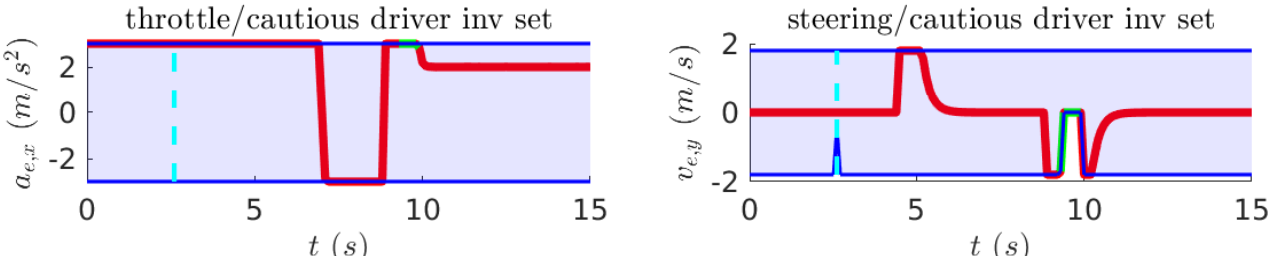
We perform an overtaking simulation in MATLAB to show how the ego car and the lead car behave with and without the supervisor, with a baseline switched MPC controller for the ego car that is chosen to mimic a human driver that undertakes the overtaking task. In the supervised case, the supervisor is implemented using the controlled invariant sets obtained by our proposed algorithm. On the other hand, the lead car behaves according to one of the two intentions. To view the simulation videos, please refer to our YouTube channel [3].

Figure 6.5a shows the MPC control inputs over time in the simulations with no supervision for the case where the lead car driver is cautious. The red lines show the MPC inputs and the blue shadow shows the safe range of throttle/steering inputs (obtained by slicing the safe input set at each time) given the user-selected steering/throttle inputs. The region without blue shadow corresponds to the time when the ego car is out of the invariant set, since no supervision is applied. In Figure 6.5a, the blue shadow in the second row covers more time steps than the first row,

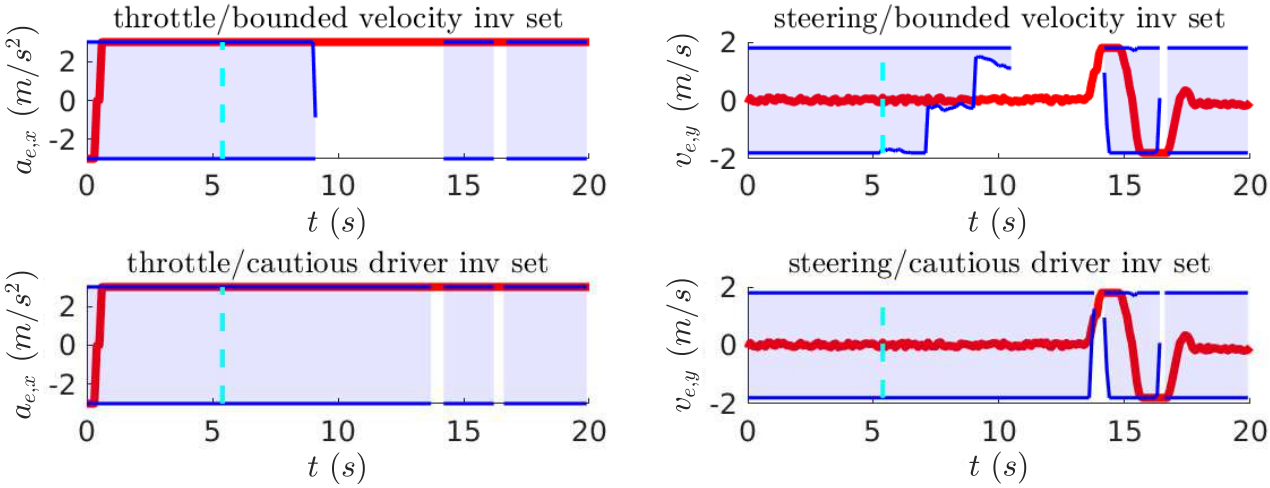
[3] https://www.youtube.com/playlist?list=PLTwM5p2COdg3AfXlSa6r_I0t0FnsU8vZF



(a) MPC without supervision



(b) MPC with supervision



(c) Human driver without supervision

Figure 6.5: The control inputs (red lines) of the ego vehicle over time (in seconds) for the following scenarios with and without supervision: ego car tailgates the lead car for a few seconds and then overtakes. The ego car in (a), (b) is controlled by an MPC controller, but in (c) is controlled by a human driver using the vehicle simulator in Figure 6.6. The lead car has cautious intention. The blue lines and shadow label the range of safe inputs given by the invariant sets. The cyan dash line labels the time when the intention estimation gives the correct intention. The green line in (b) labels the time when the ego car's inputs are overridden by the supervisor. The safe input ranges in the first and second rows in (a), (c) are computed with respect to the bounded velocity model and the cautious driver intention model respectively.

which implies that the invariant set for the cautious driver intention contains more states than the invariant set for the bounded velocity model. Therefore, once the intention of the lead vehicle is discovered (shown by cyan vertical dashed lines), the supervisor will behave less conservatively (i.e., will allow more user-selected inputs) by switching to the supervisor for the estimated intention. This is indeed the case, as can be seen in Figure 6.5b, where the intention estimation and the guardian/supervisor are engaged.

In the YouTube video list, Simulation 1 shows the animation that compares the results in Figure 6.5a and 6.5b. The same scenario with the aggressive intention is shown in Simulation 2. In addition, in the videos of Simulations 3 and 4, MPC is tuned to mimic a safe driver and a "*bad*" driver (more likely to crash with the lead car), respectively. Simulation 3 shows how such a "bad" driver crashes into the lead car in this scenario, but with supervision the driver is prevented from causing a crash. Furthermore, experimental results in Simulation 4 suggest that if the ego driver is already very careful, e.g., always keeping a safe distance with the lead car, the supervisor rarely needs to override.

6.5.4 Results from Driving Simulator

We also collected data using a driving simulator, where a human-driver is asked to perform an overtaking maneuver as described in the previous subsection. The dynamics are implemented in MATLAB/Simulink which interfaces with Unreal Engine 4 for visualization. The hardware used is a Logitech Driving Force G920 Racing Wheel for human control inputs (steering and acceleration). Fig. 6.6 shows the setup of the simulator. Fig. 6.5c shows the data from human-driver overlaid with the guardians assessment of its safety. As can be seen in the figure, the estimation of the intention significantly reduces the times human input needs to be overridden to guarantee safety.

(a) A photo of the simulator setup.

(b) Chase camera view from the simulator environment.

(c) Onboard camera view from the simulator environment.

Figure 6.6: Driving Simulator

CHAPTER 7

The Inter-Triggering Hybrid Automaton

7.1 Introduction

Recall that, in this dissertation, information uncertainty refers to the uncertainty which occurs when a single agent (or robot) must make a decision as part of a team, but does not directly observe the *entire* team. The single agent only has understanding about its individual state and the states of its neighboring teammates. Having a local understanding means that it can be difficult or impossible for the team to coordinate to optimally achieve a task like safety.

For example, consider a collection of cars on the highway. While the collection of vehicles might share the goal of traveling down the highway at the speed limit, a single car on the highway may need to slow down to take an exit or speed up to pass a vehicle in front of it. When the single vehicle makes these decisions it typically shares information about what it's doing via indicator lights, but this information is only visible to nearby vehicles. Coordination in this context is thus very difficult because little communication is shared about goals (e.g., exiting the highway) and the communication is through a very limited channel (i.e., the indicator light). We shall return to this example later on in this chapter.

Achieving a safety task using local information typically requires deep knowledge of other agents on the team, but this chapter presents a method that allows for safety guarantees to be made for the team using abstract, simple notions of interactions between agents, encoded with the idea of responsibility-sensitive safety. This chapter defines this new notion of safety and then discusses how it can be guaranteed with RCIS-based controllers. It also describes how responsibility-sensitive safety can be modified to incorporate different levels of perception of the world around an agent as well as what amount of responsibility the team would like to have for each team member.

7.2 A modeling formalism for interacting systems

We introduce *inter-triggering hybrid automata* (ITHA), a hybrid modeling formalism for collections of discrete-time hybrid systems with a special form of interaction between them. In particular, these interactions are such that they induce jumps or resets on the state evolution of individual agents.

Definition 17. *An* inter-triggering hybrid automaton *is a collection $\{\mathcal{H}_i\}_{i\in\mathcal{I}}$ of systems together with a function $\rho = (\rho_1, \ldots, \rho_{|\mathcal{I}|})$, which we refer to as a resolution function, with each $\mathcal{H}_i$, i.e., agent i, being a hybrid automaton of the form $\mathcal{H}_i = \langle \Sigma_i, \mathcal{T}_i, R_i \rangle$, where:*

- *$\Sigma_i = \langle \mathcal{X}_i, \mathcal{U}_i, \mathcal{D}_i, f_i \rangle$ are the individual dynamics for agent i;*
- *$\mathcal{T}_i$ is the set of triggering actions of agent i, including a null triggering action $\epsilon \in \mathcal{T}_i$ that indicates that agent i is not triggering a reset on any other agent;*
- *$R_i : \mathbb{N} \times \mathcal{X}_i \times \mathcal{U}_i \times 2^{\mathcal{I}} \to 2^{\mathcal{X}_i}$ is the (potentially time-varying) reset map for agent i[1];*

and where each $\rho_i : \mathbb{N} \times \mathcal{T}_1 \times \cdots \times \mathcal{T}_{|\mathcal{I}|} \to 2^{\mathcal{I}}$ is the resolution function of agent i that takes the triggering inputs of the entire collection and determines the set of agents that trigger a reset on agent i.

For notational simplicity when referring to an ITHA, we will omit the resolution function and simply say "an ITHA $\{\mathcal{H}_i\}_{i\in\mathcal{I}}$". We assume the resolution function ρ satisfies the following property, which essentially says if agent i is using a null triggering action at a given time, it will not appear in the output of the resolution function of any agents at that time.

Assumption 2. *For all i, for all t such that $\tau_i(t) = \epsilon$, we have $i \notin \rho_j(t, \ldots, \tau_i(t), \ldots)$ for all j, which implies, as a special case, $\rho_i(t, \epsilon, \ldots, \epsilon) = \emptyset$.*

Definition 18 (Execution of an ITHA)**.** *Given sequences of control inputs $\mathbf{u}_i = u_i(0), u_i(1) \ldots$ and triggering inputs $\boldsymbol{\tau}_i = \tau_i(0), \tau_i(1) \ldots$ for each agent i, an* execution *of $\{\mathcal{H}_i\}_{i\in\mathcal{I}}$ is a collection of sequences $\{\mathbf{e}_i\}_{i\in\mathcal{I}}$, each a sequence of alternating states and actions $\mathbf{e}_i = x_i(0), u_i(0), \tau_i(0), x_i(1), u_i(1), \tau_i(1), \ldots$ such that:*

$$x_i(0) \in \mathcal{X}_i \quad \forall i \in \mathcal{I} \tag{7.1a}$$

$$(u_i(t), \tau_i(t)) \in \mathcal{U}_i \times \mathcal{T}_i \quad \forall i \in \mathcal{I}, \forall t \geq 0 \tag{7.1b}$$

[1] With a slight abuse of notation, the last argument of the reset map is shown as an index set, but the actual reset value depends on the state, input, and triggering action of the agents in that set.

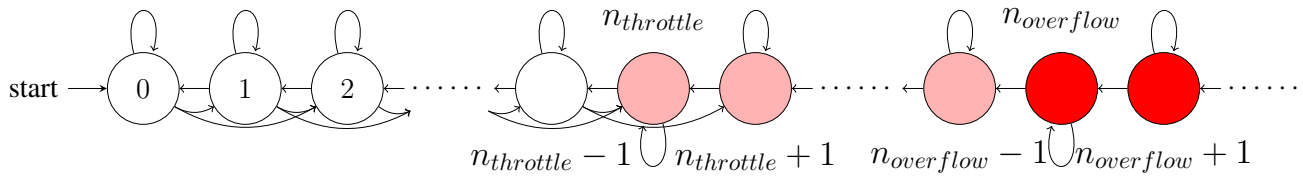


Figure 7.1: The individual dynamics for a processor in the collection specified in Example 4. The processor is unable to accept as many jobs when $x_i(t) \geq n_{throttle}$ and it experiences a stack overflow (i.e. it fails) if $x_i(t) \geq n_{overflow}$.

$$x_i(t+1) \in \begin{cases} f_i(x_i(t), u_i(t), \mathcal{D}_i) & \text{if } \rho_i(t) = \emptyset, \\ R_i(t, x_i(t), u_i(t), \rho_i(t)) & \text{otherwise.} \end{cases} \tag{7.1c}$$

where $\rho_i(t) \triangleq \rho_i(t, \tau_1(t), \tau_2(t), ..., \tau_{|\mathcal{I}|}(t))$.

For an element $\mathbf{e}_i$ of an execution, we denote the corresponding state trajectory by $\mathbf{x}_i = x_i(0), x_i(1), \ldots$. We consider problems related to safety of an execution of an ITHA, where we require the state trajectory of each agent $\mathcal{H}_i$ to remain in a safe set $\mathcal{X}_{i,safe} \subseteq \mathcal{X}_i$ for all times. We use an execution $\{\mathbf{e}_i\}_{i\in\mathcal{I}}$ remaining in a collection of sets $\{\mathcal{X}_{i,safe}\}_{i\in\mathcal{I}}$ inter-changeably with the corresponding state trajectories $\{\mathbf{x}_i\}_{i\in\mathcal{I}}$ remaining in the same collection.

To make the definition of inter-triggering hybrid automaton concrete, we present two examples used throughout the chapter.

Example 4 (Parallel Processors on a Server Farm)**.** *A collection of processors in a server farm can be treated as a collection of agents where each agent's state is the number of jobs it has left to compute. In other words, agent i's state $x_i \in \mathbb{N}$, where there is a limit of jobs, $n_{overflow}$, over which the processor will create a stack overflow and fail. External jobs d_i for processor i are passed into the server according to a protocol that blocks new jobs from coming in if $x_i \geq n_{throttle}$ where $n_{throttle} < n_{overflow}$ and the processor always can take the action to address a job in its queue or do nothing. Thus, the individual dynamics can be visualized as shown in Fig. 7.1. In addition, the processors can be recruited by other processors according to a directed graph $\mathcal{G}$ that indicates which processors can send jobs to which other processors, i.e. processor i can recruit processor j if $(i,j) \in \mathcal{E}$. An example of such a graph is shown in Fig. 2.1. This scenario can be modeled with the representation $\{\mathcal{H}_i\}_{i\in\mathcal{I}}$, where each agent $\mathcal{H}_i = \langle \Sigma_i, \mathcal{T}_i, R_i \rangle$ is composed of the following parts:*

- *Dynamics Σ_i where $\mathcal{X}_i = \mathbb{N}$ is the queue of jobs to be done by agent i; $\mathcal{U}_i = \{0, -1\}$ represents processor i's choice to do nothing (i.e. $u_i(t) = 0$) or to address one of the jobs in its queue (i.e. $u_i(t) = -1$); $\mathcal{D}_i \in \{0, 1, 2\}$ represents the number of external jobs passed*

into processor i, and f_i, given by

$$f_i(x_i, u_i, d_i) = \begin{cases} x_i + u_i & \text{if } x_i \geq n_{throttle} \\ x_i + u_i + d_i & \text{otherwise}, \end{cases}$$

describes how the queue of jobs is changing for agent i in the absence of it being recruited;

- *$\mathcal{T}_i = 2^{out_{\mathcal{G}}(i)}$ represents the possible sets of agents that agent i recruits to help it with its queue, with null element $\epsilon = \emptyset$, according to its outgoing edges in $\mathcal{G}$;*

- *Reset map R_i describes how the queue for agent i changes if it was recruited by or recruited another agent. To each agent in the recruit set (i.e. $\forall j \in \tau_i(t) \in \mathcal{T}_i$), agent i sends 1 job from its queue to that processor:*

$$R_i(t, x_i(t), u_i(t), S) = \begin{cases} 0 & \text{if } s_i(t) \leq 0, \\ s_i(t) & \text{otherwise}, \end{cases} \tag{7.2}$$

where $s_i(t) = x_i(t) - |\tau_i(t)| + u_i(t) + |S| + d_i(t)$, where S is the set of agents recruiting agent i.

Furthermore, we can write the i^{th} component of the resolution function $\rho_i(t) = \{j \in \mathcal{I} \mid i \in \tau_j(t)\}$; that is, the set of agents triggering agent i at time t is the set of agents which contain i in its triggering action at time t. Concretely, for agent 1 in Fig. 2.1, $\rho_1(t) \subseteq \{2, 4\}$, for all t. If $\tau_2(t) = \epsilon$ and $\tau_4(t) = \{1, 6\}$, then $\rho_1(t) = \{4\}$, regardless of the triggering actions of the remaining agents.

Note that each processor can avoid the overflow states indefinitely if it is within a robust control invariant set that is completely contained in the safe set $\{x \in \mathbb{N} \mid x < n_{overflow}\}$. Under the individual dynamics, the maximal control invariant set in the safe set (i.e. $\mathcal{C} \subseteq \{x \in \mathbb{N} \mid x < n_{overflow}\}$) can be quickly shown to be $\mathcal{C} = \{x \in \mathbb{N} \mid x \leq n_{overflow} - 1\}$. Depending on the objectives of the processors, i.e. maximizing throughput, each processor may need to trigger other agents, and without adequate precaution the triggering can reset the states of some processors above $n_{overflow}$, leading to unsafe behavior.

Example 5 (Highway Driving)**.** *Consider a collection of vehicles travelling in the same direction on a highway (see Fig. 7.2). This collection can be represented by an inter-triggering hybrid automaton $\{\mathcal{H}_i\}_{i \in \mathcal{I}}$ where each agent $\mathcal{H}_i = \langle \Sigma_i, \mathcal{T}_i, R_i \rangle$ is composed of:*

- *Dynamics Σ_i with $\mathcal{X}_i = [0, v_{max}] \times [0, \infty) \times [0, v_{max}]$, where state $x_i = [v_i, h_i, v_i^L]^\intercal$ contains v_i (velocity of current agent i, henceforth referred to as the ego vehicle), h_i (headway between this agent and the nearest car in front of it on the same lane, henceforth referred*

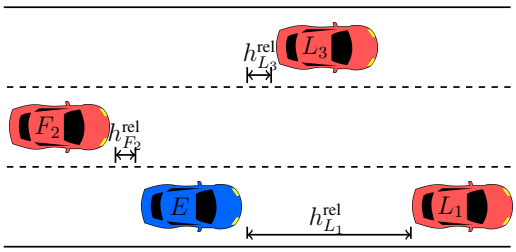


Figure 7.2: A collection of vehicles on the highway, as described in Example 5. The ego vehicle E is marked in blue, and longitudinal distances between the ego vehicle and car i are marked as h_i^{rel}.

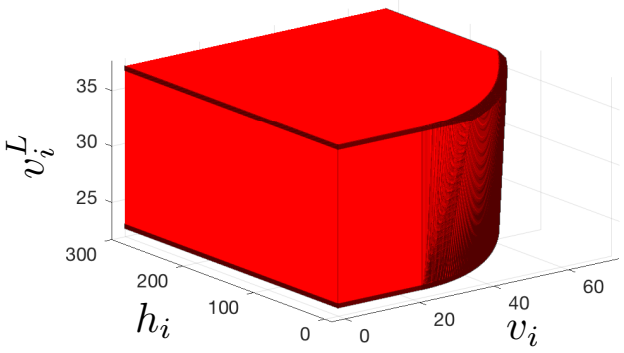


Figure 7.3: The controlled invariant set for the car-following system defined in (7.3) for parameters used in [111, 149].

to as the lead vehicle), and v_i^L (the velocity of the lead vehicle), $\mathcal{U}_i$ is the set of allowed inputs, with input $u_i = a_i$ being the acceleration of the ego car, $\mathcal{D}_i$ is the set of allowable disturbances, with disturbance $d_i = a_i^L$ being the acceleration of the lead vehicle, and the system dynamics $f_i : \mathcal{X}_i \times \mathcal{U}_i \times \mathcal{D}_i \to \mathcal{X}_i$ is such that

$$f_i(x_i, u_i, d_i) = \begin{bmatrix} 1 & 0 & 0 \\ -\Delta t & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} x_i + \begin{bmatrix} \Delta t \\ 0 \\ 0 \end{bmatrix} u_i + \begin{bmatrix} 0 \\ 0 \\ \Delta t \end{bmatrix} d_i \tag{7.3}$$

where Δt is the sampling time;

- *$\mathcal{T}_i = \{\text{stay}, \text{left}, \text{right}\}$ is the set of possible lane change decisions, with null element $\epsilon = \text{stay}$;*

- *A reset $R_i(t, x_i(t), u_i(t), \rho_i(t))$ is triggered on agent i if one of the following happens (1) $\tau_i(t) \neq \epsilon$, (2) if their lead car is j and $\tau_j(t) \neq \epsilon$ and/or (3) a car becomes the current agent's lead car (i.e. lead car was j at time $t-1$ but the lead car becomes $k \neq j$ at time t). The value of the reset depends on the triggering actions of all agents $\rho_i(t)$, which can possibly affect agent i at time t. This determines the new lead car and hence the new values of h_i and v_i^L, while v_i evolves according to individual dynamics in* (7.3)*. A more precise discussion of the reset map is considered in Appendix B.*

Controlled invariant sets for the system defined in (7.3) *can be computed using polyhedral set*

computation methods such as those discussed in [111, 149]. An example of such a set is shown in Fig. 7.3.

To further illustrate how the reset map is defined for highway driving, the value of R_i will be explained for some vehicles in Fig. 7.2. First, suppose that the current time is t and vehicles E and F_2 have states $x_E = [v_E, h_E, v_E^L]^\intercal$ and $x_{F_2} = [v_{F_2}, h_{F_2}, v_{F_2}^L]^\intercal$, respectively, and apply control inputs u_E and u_{F_2}. Suppose that the ego vehicle makes a left lane change at time t; this action triggers a reset of its own continuous state as well as that of F_2 (leading to changes in headway and lead car velocity for both vehicles). Formally, if $\tau_E(t) = \{\textit{left}\}$, then

$$R_E(t, x_E, u_E, \{E\}) = \begin{bmatrix} v_E + \Delta t u_E \\ \infty \\ v_{max} \end{bmatrix}$$

and

$$R_{F_2}(t, x_{F_2}, u_{F_2}, \{E\}) = \begin{bmatrix} v_{F_2} + \Delta t u_{F_2} \\ h_{F_2}^{rel} + (v_E - v_{F_2})\Delta t \\ v_E + \Delta t u_E \end{bmatrix}$$

where we assume by convention that resetting an agent to have no lead car results in a headway of ∞ and a lead car velocity of v_{max}.

As concrete examples, we overview a subset of possible values that the resolution function may take in the highway setting. We can write $\rho_E(t, \tau_E(t) = \textit{left}, \tau_{L_1}(t) = \epsilon, \tau_{F_2}(t) = \epsilon, \tau_{L_3}(t) = \textit{right}) = \{E, L_3\}$, as both the ego vehicle and vehicle L_3 changing lanes resets the continuous state of the ego vehicle. Likewise, we can write $\rho_E(t, \tau_E(t) = \epsilon, \tau_{L_1}(t) = \epsilon, \tau_{F_2}(t) = \textit{right}, \tau_{L_3}(t) = \epsilon) = \emptyset$, as F_2 will not trigger a state reset on the ego vehicle. Vehicle L_1 can also trigger a reset on the state of the ego vehicle by changing lanes, i.e. $\rho_E(t, \tau_E(t) = \epsilon, \tau_{L_1}(t) = \textit{left}, \tau_{F_2}(t) = \epsilon, \tau_{L_3}(t) = \textit{right}) = \{L_1\}$. A final, more complicated case occurs when vehicles E and L_1 both make a left lane change, while L_3 makes a right lane change at the same time: $\rho_E(t, \tau_E(t) = \textit{left}, \tau_{L_1}(t) = \textit{left}, \tau_{F_2}(t) = \epsilon, \tau_{L_3}(t) = \textit{right}) = \{E, L_3\}$; in this case, L_1 does not end up factoring into the resolution function as L_3 becomes the ego vehicle's lead car instead.

7.3 Compositional Safety Rules

In general, not all executions of an inter-triggering hybrid automaton are safe. To render the executions safe, control policies may need to restrict the possible control inputs and triggering actions of individual agents. In this section, we develop sufficient conditions on local control policies that collectively guarantee safety. To do this, robust controlled invariant sets are found for the

inter-triggering hybrid automaton's individual dynamics and then responsibility-sensitive safe controllers are defined with respect to these individual invariant sets. This section shows that safety can be guaranteed when all agents in the collection use such responsibility-sensitive safe controllers and then discusses how conservativeness can be further reduced via a communication scheme.

7.3.1 Control Policies and Safety Control Problem for ITHA

At run-time each agent i picks its control inputs $u_i(t)$ and triggering actions $\tau_i(t)$ based on the information available to it by time t. Formally, for a given set $\mathcal{Y}_i$ of possible observations of agent i, a memoryless local controller (or, control policy) is a function $\gamma_i : \mathcal{Y}_i \to \mathcal{U}_i \times \mathcal{T}_i$. Similarly, a local controller with memory is a function $\gamma_i : \mathcal{Y}_i^+ \to \mathcal{U}_i \times \mathcal{T}_i$, where the superscript $+$ denotes finite non-zero repetition. If agent i's decisions only depend on its own state or the state of all agents, we have $\mathcal{Y}_i = \mathcal{X}_i$ or $\mathcal{Y}_i = \mathcal{X}_1 \times \ldots \times \mathcal{X}_{|\mathcal{I}|}$, respectively. Also, if an agent can access a (potentially time-varying) subset of other agents' states, we have $\mathcal{Y}_i = \bigcup_{\mathcal{I}' \subset \mathcal{I}} \{\mathcal{X}_j\}_{j \in \mathcal{I}'}$. In addition to states, $\mathcal{Y}_i$ can incorporate observations of actions of the other agents, which would be relevant when introducing the communication scheme in section 7.3.3.2.

Definition 19 (Controlled Execution of an ITHA)**.** *Given a collection of controllers* $\{\gamma_i\}_{i \in \mathcal{I}}$*,* $\{\gamma_i\}_{i \in \mathcal{I}}$-controlled executions of $\{\mathcal{H}_i\}_{i \in \mathcal{I}}$ *are the set of executions where control inputs* $\mathbf{u}_i$ *and triggering inputs* $\boldsymbol{\tau}_i$ *are produced according to the function* γ_i *for all* i*.*

Given a collection of local safe sets $\{\mathcal{X}_{i,safe}\}_{i \in \mathcal{I}}$ and information $\mathcal{Y}_i$ available to each agent, synthesizing local controllers for each agent to guarantee global safety is a distributed synthesis problem [118]. Verifying existence of such controllers is known to be undecidable in general even when the sets $\mathcal{X}_i$, $\mathcal{U}_i$, $\mathcal{D}_i$ are finite [52, 33]. Any architecture defining the information flow in a distributed synthesis problem can be captured by choosing some $\mathcal{I}_i'$ and setting $\mathcal{Y}_i = \{\mathcal{X}_j\}_{j \in \mathcal{I}_i'}$, therefore synthesis in the ITHA setting cannot be easier. Given this hardness result, we instead search for sufficient conditions on local controllers under which global safety is guaranteed. These conditions can be checked locally and instantaneously in time. Moreover, instead of working with a fixed observation structure, we will deduce the sets $\mathcal{Y}_i$ each local agent should have access to in order to be able to comply with the conditions.

7.3.2 Responsibility-Sensitive Safety

Responsibility-sensitive safety consists of two rules. The first rule handles safety of the individual dynamics and the second rule handles safety during triggering interactions. Consider the first rule:

Definition 20 (Self-Safety)**.** *A control policy γ_i renders agent $\mathcal{H}_i$* self-safe *on a set $\mathcal{X}_{i,c} \subseteq \mathcal{X}_i$ if for all states in $\mathcal{X}_{i,c}$, the control input guarantees $\mathcal{H}_i$'s own safety assuming a reset will not happen in the next step. In math, for all t, if $x_i(t) \in \mathcal{X}_{i,c}$, then $u_i(t)$ produced by γ_i is such that*

$$f_i(x_i(t), u_i(t), \mathcal{D}_i) \subseteq \mathcal{X}_{i,c}. \tag{7.4}$$

It is clear from (2.9) that for the existence of a self-safe controller on $\mathcal{X}_{i,c}$, $\mathcal{X}_{i,c}$ should be a robust controlled invariant set. Moreover, the controller would only need information on the agent's own state to be in $\mathcal{Y}_i$. Though even when $\mathcal{X}_{i,c}$'s are robust controlled invariant sets for Σ_i's, adopting a controller that renders $\mathcal{X}_{i,c}$ invariant, an agent $\mathcal{H}_i$ cannot be guaranteed to remain in $\mathcal{X}_{i,c}$ because its state trajectories depend on both f_i and R_i. To incorporate the potential resets of agent $\mathcal{H}_i$ into an invariance condition, any agent j contributing to a reset on agent i, i.e., $j \in \rho_i(t)$, should somehow make guarantees about R_i on the same set. While it may be problematic to expect agents to know ρ in a distributed setting, an over-approximation of the value of ρ at each time step, as defined next, can be obtained locally in many practical scenarios.

Definition 21 (Resolution Over-approximation)**.** *A function $\hat{\rho}_i : \mathbb{N} \times \mathcal{T}_1 \times \mathcal{T}_2 \times \cdots \times \mathcal{T}_{|\mathcal{I}|} \to 2^{2^{\mathcal{I}}}$ is an* over-approximation *of the i^{th} component of the resolution function if and only if:*

$$\rho_i(t, \tau_1', \tau_2', \cdots, \tau_{|\mathcal{I}|}') \in \hat{\rho}_i(t, \tau_1', \tau_2', \cdots, \tau_{|\mathcal{I}|}').$$

for all values of t and all $\tau_i' \in \mathcal{T}_i$ for all i. Similarly, we say $\hat{\rho}$ is an over-approximation of the resolution function ρ, denoted as $\hat{\rho} \supseteq \rho$, if and only if each $\hat{\rho}_i$ is an over-approximation of corresponding ρ_i.

For notation convenience, when the triggering action arguments of $\hat{\rho}$ are clear from the context or are irrelevant, we simply write $\hat{\rho}_i(t)$. We discuss how resolution over-approximations can be obtained locally by each agent in Section 7.3.3. In general, different agents j might have different resolution over-approximations $\hat{\rho}^{(j)} \supseteq \rho$ depending on their local information. With this in mind, to enable safety through resets, we define a responsibility rule that uses such over-approximations.

Definition 22 ($\hat{\rho}$-Responsibility)**.** *Given an over-approximation $\hat{\rho}$ of the resolution function and a collection $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ of sets, a controller γ_j renders an agent $\mathcal{H}_j$ $\hat{\rho}$-*responsible *with respect to the sets $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ if, when agent j triggers a reset on other agents, agent j's triggering action does not lead to safety violations for any other agent that it could induce a reset on according to $\hat{\rho}$, possibly including itself. In math, the controller γ_j renders $\mathcal{H}_j$ $\hat{\rho}$-*responsible*, if for all t, $\tau_j(t)$ and $u_j(t)$ produced by the controller are such that if $\tau_j(t) \neq \epsilon$ and $x_i(t) \in \mathcal{X}_{i,c}$ $\forall i \in \mathcal{I}$, then for all $i \in \mathcal{I}$*

and $S \in \hat{\rho}_i(t)$ with $j \in S$ we have:

$$\begin{cases} R_i(t, x_i(t), \mathcal{U}_i, S) \subseteq \mathcal{X}_{i,c} & \text{if } i \neq j, \text{and} \\ R_i(t, x_i(t), u_i(t), S) \subseteq \mathcal{X}_{i,c} & \text{if } i = j. \end{cases} \tag{7.5}$$

We use controller being $\hat{\rho}$-responsible (or, self-safe), agent being $\hat{\rho}$-responsible (or, self-safe) and controller rendering an agent $\hat{\rho}$-responsible (or, self-safe), interchangeably. With all of the above the following theorem can be stated, which provides a recursive safety guarantee.

Theorem 5. *Consider an inter-triggering hybrid automaton $\{\mathcal{H}_i\}_{i\in\mathcal{I}}$, an accompanying collection of sets $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ that are robustly controlled invariant for respective Σ_i's in their respective safe sets $\{\mathcal{X}_{i,safe}\}_{i\in\mathcal{I}}$ and a collection $\{\hat{\rho}^{(i)}\}_{i\in\mathcal{I}}$ of resolution over-approximations. Then,*

1. *there exists local controllers γ_i for each agent $\mathcal{H}_i$ that render them self-safe and $\hat{\rho}^{(i)}$-responsible with respect to the sets $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ and*

2. *if each agent uses a controller γ_i that renders itself self-safe and $\hat{\rho}^{(i)}$-responsible with respect to the sets $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$, the state trajectories corresponding to any $\{\gamma_i\}_{i\in\mathcal{I}}$-controlled execution of $\{\mathcal{H}_i\}_{i\in\mathcal{I}}$ beginning in $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ always remain within these sets.*

Proof. To prove statement (1), consider, for each $\mathcal{H}_i$, a controller that produces an input $u_i(t) \in \{u \mid f(x_i(t), u, \mathcal{D}) \subseteq \mathcal{X}_{i,c}\}$ and the triggering action $\tau_i(t) = \epsilon$ for all time t for which $x_i(t) \in \mathcal{X}_{i,c}$. With the triggering action $\tau_i(t) = \epsilon$, the controller γ_i trivially satisfies the definition of $\hat{\rho}$-responsibility. Also, $\mathcal{X}_{i,c}$ being a robust controlled invariant set guarantees $u_i(t)$ exists whenever $x_i(t) \in \mathcal{X}_{i,c}$ and with this $u_i(t)$ the controller satisfies (7.4).

To show statement (2), we use induction on time. In the base case ($t = 0$), by assumption, all agents satisfy $x_i(0) \in \mathcal{X}_{i,c}$. Assume at time $t = k$, each agent's state $x_i(k)$ is in its corresponding set $\mathcal{X}_{i,c}$. The controller γ_i either produces (i) $\tau_i(k) = \epsilon$ or (ii) $\tau_i(k) \neq \epsilon$.

First, consider case (i). Since controller γ_i renders $\mathcal{H}_i$ self-safe, $u_i'(k) \in \mathcal{U}_i$ produced by it satisfies (7.4). With this choice of $u_i'(k)$, there are two possibilities for state evolution. If $\rho_i(k) = \emptyset$, the state x_i evolves with the first line of Eq. (7.1c) and we have $x_i(k+1) \in \mathcal{X}_{i,c}$ by (7.4). If $\rho_i(k) \neq \emptyset$, state x_i evolves with the second line of Eq. (7.1c), that is, $x_i(k+1) \in R_i(k, x_i(k), u_i'(k), \rho_i(k))$. Let $j \in \rho_i(k) \in \hat{\rho}_i^{(j)}(k)$. By Assumption 2, $\tau_j(k) \neq \epsilon$. By agent j being $\hat{\rho}^{(j)}$-responsible with $\tau_j(k) \neq \epsilon$, for any $S \in \hat{\rho}_i^{(j)}(k)$ with $j \in S$, and, in particular for $S = \rho_i(k)$, the first line of (7.5) is satisfied. Since $R_i(k, x_i(k), u_i'(k), \rho_i(k)) \subseteq R_i(k, x_i(k), \mathcal{U}_i, \rho_i(k))$ and $j \in \rho_i(k)$ was arbitrary, $x_i(k+1) \in \mathcal{X}_{i,c}$ follows.

Now, consider case (ii). By assumption, the controller γ_i produces $\tau_i^*(k) \neq \epsilon$ and $u_i^*(k)$ such that both conditions in (7.5) and condition (7.4) are satisfied. Then, if $\rho_i(k) = \emptyset$, the state x_i

evolves with the first line of Eq. (7.1c) and we have $x_i(k+1) \in \mathcal{X}_{i,c}$ by (7.4). If $\rho_i(k) \neq \emptyset$, state x_i evolves with the second line of Eq. (7.1c), that is, $x_i(k+1) \in R_i(k, x_i(k), u_i^*(k), \rho_i(k))$. Let $j \in \rho_i(k) \in \hat{\rho}_i^{(j)}(k)$. If $j \neq i$, the reasoning in case (i) above holds. If $j = i \in \rho_i(k)$, by assumption, $u_i^*(k)$ also satisfies the second line of (7.5) for any $S \in \hat{\rho}_i^{(j)}(k)$ with $i \in S$, and in particular for $S = \rho_i(k)$. Therefore, $x_i(k+1) \in R_i(k, x_i(k), u_i^*(k), \rho_i(k)) \subseteq \mathcal{X}_{i,c}$. □

This theorem essentially says for ITHA, existence of controlled invariant sets for individual dynamics is a sufficient condition for ensuring global safety. However, this is not a necessary condition and our results do not apply to the cases where the only way to ensure safety is via triggering. The next result relates the self-safety and responsibility conditions to "not being at fault" as in [141] in the sense that if an agent's control policy is self-safe and $\hat{\rho}$-responsible, there exists controllers for the remaining agents such that the overall system stays safe.

Corollary 1. *Consider an inter-triggering hybrid automaton $\{\mathcal{H}_i\}_{i\in\mathcal{I}}$, an accompanying collection of sets $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ that are robustly controlled invariant for respective Σ_i's in their respective safe sets $\{\mathcal{X}_{i,safe}\}_{i\in\mathcal{I}}$ and a collection $\{\hat{\rho}^{(i)}\}_{i\in\mathcal{I}}$ of resolution over-approximations. If some subset $\{\mathcal{H}_j\}_{j\in\mathcal{I}'}$ with $\mathcal{I}' \subset \mathcal{I}$ of agents have controllers γ_j which are both self-safe and $\hat{\rho}^{(j)}$-responsible on the sets $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$, then there exists controllers γ_i for all of the other agents $\{\mathcal{H}_i\}_{i\in\mathcal{I}\setminus\mathcal{I}'}$ such that the state trajectories corresponding to any $\{\gamma_i\}_{i\in\mathcal{I}}$-controlled execution of $\{\mathcal{H}_i\}_{i\in\mathcal{I}}$ beginning in $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ always remain within these sets.*

Proof. For each $i \in \mathcal{I} \setminus \mathcal{I}'$, consider the controller γ_i that produces an input $u_i(t) \in \{u \mid f(x_i(t), u, \mathcal{D}) \subseteq \mathcal{X}_{i,c}\}$, which exists by $\mathcal{X}_{i,c}$ being robust controlled invariant, and triggering action $\tau_i(t) = \epsilon$ for all time t for which $x_i(t) \in \mathcal{X}_{i,c}$. As shown in the proof of Theorem 5 statement (1), such γ_i is self-safe and $\hat{\rho}^{(i)}$-responsible on $\mathcal{X}_{i,c}$. Since γ_j for $j \in \mathcal{I}'$ are given to be self-safe and $\hat{\rho}^{(j)}$-responsible on $\mathcal{X}_{j,c}$, with the above choice of controllers for agents in $\mathcal{I} \setminus \mathcal{I}'$, all the controllers are self-safe and $\hat{\rho}$-responsible, which by statement (2) of Theorem 5 ensures safety of the executions. □

One can try to verify self-safety and responsibility for given sets $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$, controllers γ_i and resolution over approximations $\hat{\rho}$. Conditions (7.4) and (7.5) can also be used to synthesize controllers that render an ITHA self-safe and $\hat{\rho}$-responsible or to supervise existing controllers at run-time. The latter two are the use cases we demonstrate in Section 7.4 using robust controlled invariant sets for $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$. Given $\hat{\rho}$, the basic idea is to construct the set of all triggering actions and control inputs that together satisfy conditions (7.4) and (7.5). This set is always non-empty when $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ are robust controlled invariant sets and it can be constructed at run-time. Then, for synthesis, a pair (u_j, τ_j) is picked from this set and for supervision, we check if the controller's u_j, τ_j is in this set or not. A few comments are in order as to what information, in general, is

needed to construct this set, which also prescribes what observations should be included in $\mathcal{Y}_j$ to implement a controller γ_j constructed this way. In general, the states of all agents i, for which j appears in the sets in $\hat{\rho}_i(t)$ should be included in $\mathcal{Y}_j$. However, we note that the reset maps together with the collection $\{\mathcal{X}_{i,c}\}_{i\in\mathcal{I}}$ of sets in practice have more structure that can simplify checking for $\hat{\rho}$-reponsibility or the amount of observations needed. For instance, for the processor example, for all $S \subseteq \mathcal{I}$, $R_i(\cdot,\cdot,\cdot,S) \subseteq \mathcal{X}_{i,c}$ implies for all S' with $|S'| \leq |S|$, $R_i(\cdot,\cdot,\cdot,S') \subseteq \mathcal{X}_{i,c}$. In words, if the processor i is safe when recruited by a number of other processors, it will be safe when recruited by a smaller number of processors. This implies that it is enough to check the condition (7.5) only for the largest cardinality S containing j instead of all such sets. Similarly, for the highway example, for all $S \subseteq \mathcal{I}$, there is an $S^*_{t,i} \subseteq S$ such that $R_i(t,\cdot,\cdot,S^*_{t,i}) \subseteq \mathcal{X}_{i,c}$ implies $R_i(t,\cdot,\cdot,\tilde{S}) \subseteq \mathcal{X}_{i,c}$ for all non-empty $\tilde{S} \subseteq S$. This is because there is a "worst-case" lane switching leading to a "worst-case" reset. In a sense, it does not matter what switching actions an arbitrary agent takes; only agents close to agent j matter. Thus, an ego vehicle can reason over the set of agents switching lanes nearest to itself while still being able to guarantee safety. It is also worth remarking that when either R_i or $\mathcal{X}_{i,c}$ is not known exactly, an over-approximation of R_i and an under-approximation of $\mathcal{X}_{i,c}$ can be used in (7.5) while still guaranteeing overall safety of the ITHA per Theorem 5. Moreover, as discussed in the next section, $\hat{\rho}^{(i)}$ can be constructed on the fly, meaning $\hat{\rho}^{(i)}$ is only known up to $\hat{\rho}^{(i)}(t)$ at a given time t but this is enough to construct a controller that is self-safe and $\hat{\rho}^{(i)}$-responsible at time t.

7.3.3 Finding Resolution Over-Approximations

Both complexity and conservativeness can be exacerbated if the over-approximation $\hat{\rho}^{(i)}$ is "far" from the true ρ. The task of identifying proper over-approximations of ρ is thus a vitally important one. Insights into the structure of the problem can be used to generate good over-approximations.

We start this section with a relatively easy to compute, yet possibly conservative, over-approximation. Then, we define an order between agents through which they can communicate and substantially reduce conservatism. Along the way, we also discuss how these over-approximations look for our running examples.

7.3.3.1 Trivial Over-Approximations:

If we assume each agent knows the resolution function ρ, a trivial over-approximation of the resolution function can be locally computed at each time step by considering every possible choice of triggering inputs for all other agents. In math, agent i computes the j^{th} component of a trivial over-approximation $\rho^{(i)}_j$ as:

$$\hat{\rho}_j^{(i)}(t) = \hat{\rho}_j^{(i)}(t, \tau_i(t)) = \left\{ \rho' \in 2^{\mathcal{I}} \;\middle|\; \begin{array}{l} \exists \tau_k \in \mathcal{T}_k \;\; \forall k \in \mathcal{I} \setminus \{j\} : \\ \rho' = \rho_j(t, \tau_1, \ldots, \tau_i(t), \ldots, \tau_{|\mathcal{I}|}) \end{array} \right\} \tag{7.6}$$

It can be easily shown that $\hat{\rho}^{(i)}$, components of which are constructed as above is an over-approximation of ρ.

Example 6 (Cont'd Example 4)**.** *For the server farm in Fig. 2.1, we revisit the case where $\tau_2(t) = \epsilon$ and $\tau_4(t) = \{1, 6\}$, which leads to $\rho_1(t) = \{4\}$, regardless of the triggering actions of the remaining agents. Using the trivial over-approximation, processor 1 has $\hat{\rho}_1^{(1)}(t) = \{\emptyset, \{2\}, \{4\}, \{2, 4\}\}$, processor 2 has $\hat{\rho}_1^{(2)}(t) = \{\emptyset, \{4\}\}$ and processor 4 has $\hat{\rho}_1^{(4)}(t) = \{\{4\}, \{2, 4\}\}$. Note that no estimates depend on $\tau_1(t)$, as agent 1 cannot recruit itself.*

Example 7 (Cont'd Example 5)**.** *Consider the trivial over-approximation from the perspective of the ego agent, in the case where $\tau_E(t) = \epsilon$: the only possible resets depend on if L_1 does or does not trigger a reset on E; that is, $\hat{\rho}_E^{(E)}(t) = \{\emptyset, \{L_1\}\}$. If instead $\tau_E(t) =$ left, it is more complicated: $\hat{\rho}_E^{(E)}(t) = \{\{E\}, \{E, L_3\}, \{E, L_1\}\}$. The first case occurs if neither L_1 nor L_3 changes to the center lane simultaneously, the second case occurs if L_3 changes to the center lane, regardless of the triggering action of L_1, and the last case occurs if L_1 makes a lane change and L_3 does not.*

7.3.3.2 Ordered Actions

In some situations, agents in an inter-triggering hybrid automaton $\{\mathcal{H}_i\}_{i \in \mathcal{I}}$ can communicate their planned actions with one another. If such communication is done in an "orderly" manner, it can allow agents to obtain much refined over-approximations as they no longer need to consider all possible actions of the other agents.

We again assume each agent knows the resolution function ρ. Moreover, we assume at each time that there is a total order $\succeq_t$ among the agents that all agents know and use to communicate their planned triggering inputs.[2]

Given such an order, we propose Algorithm 1 for each agent to construct their resolution over-approximation at each time t. With abuse of notation and without loss of generality, we assume that the automaton $\{\mathcal{H}_i\}_{i \in \mathcal{I}}$ is (re)ordered/(re)indexed (by keeping track of their associated over-estimates computed so far) at each time t so that $\{\mathcal{H}_i\}_{i \in \mathcal{I}} = \{\mathcal{H}_1, \mathcal{H}_2, \cdots, \mathcal{H}_{|\mathcal{I}|}\}$ and the index of each agent represents its ranking according to $\succeq_t$. Then, Algorithm 1 is called starting with $\mathcal{H}_1$ to compute resolution over-approximations $\hat{\rho}^{(1)}$ and to choose a triggering action $\tau_1(t)$ for which a

[2]The assumption of $\succeq_t$ being a total order can be relaxed. In particular, agents can still get an over-approximation for a class of partial orders $\succeq_t$ for which the Hasse diagram of the partially ordered set $(\{\mathcal{H}_i\}_{i \in \mathcal{I}}, \succeq_t)$ is a rooted forest at each time, i.e., an agent does not receive information from two incomparable agents at a given time.

self-safe and $\hat{\rho}^{(1)}$-responsible control input $u_1(t)$ exists. Then, $\tau_1(t)$ is shared with the next agent and agent $\mathcal{H}_2$ calls the algorithm with $\{\tau_1(t)\}$, and so on, until all agents compute their triggering actions for time t. Then these actions are executed and time progresses.

Algorithm 1: Resolution over-approximation construction with ordered actions

Result: $\hat{\rho}^{(j)}(t, \tau_j)$, $\tau_j(t)$
Input: $\mathcal{H}_j$, $\{\tau_i(t)\}_{i=1}^{j-1}$, $\{x_i(t)\}_{i\in\mathcal{I}}$
$\widetilde{\mathcal{T}}_j \leftarrow \emptyset$
for $\tau_j \in \mathcal{T}_j$ **do**
  $\hat{\rho}^{(j)}(t, \tau_j) \leftarrow \emptyset$
  for $(\tau_{j+1}, \cdots, \tau_{|\mathcal{I}|}) \in \mathcal{T}_{j+1} \times \cdots \times \mathcal{T}_{|\mathcal{I}|}$ **do**
    $S \leftarrow \rho(t, \tau_1(t), \tau_2(t), \cdots, \tau_{j-1}(t), \tau_j, \cdots, \tau_{|\mathcal{I}|})$
    if $\forall u_j \in \mathcal{U}_j,\ R_j(t, x_j(t), u_j, S) \notin \mathcal{X}_{j,c}$ **then**
      continue;
    end
    if $\exists i \neq j$ *s.t.* $R_i(t, x_i(t), \mathcal{U}_i, S) \notin \mathcal{X}_{i,c}$ **then**
      continue;
    end
    $\hat{\rho}^{(j)}(t, \tau_j) \leftarrow \hat{\rho}^{(j)}(\tau_j) \cup \{S\}$
    $\widetilde{\mathcal{T}}_j \leftarrow \widetilde{\mathcal{T}}_j \cup \{\tau_j\}$;
  end
end
$\tau_j(t) \in \widetilde{\mathcal{T}}_j$

This scheme, and particularly the method for constructing $\hat{\rho} = \{\hat{\rho}_i\}_{i\in\mathcal{I}}$ can be shown to produce an over-approximation.

Lemma 10. *Calling Algorithm 1 at each time step according to order $\succeq_t$ produces functions $\hat{\rho}^{(j)}$, each of which is an over-approximation of ρ at every time step.*

Proof. Note that $\rho(t) = \{\rho_i(t, \tau_1(t), \tau_2(t), \cdots, \tau_{|\mathcal{I}|}(t))\}_{i\in\mathcal{I}}$. Thus, when $\{\tau_i(t)\}_{i\in\mathcal{I}}$ is completely known, we explicitly know $\rho(t)$. At the $|\mathcal{I}|^{th}$ call of Algorithm 1 at time t, after $\tau_{|\mathcal{I}|}(t)$ is chosen then $\{\tau_i(t)\}_{i\in\mathcal{I}}$ is completely known. This indicates that $\rho(t) \in \hat{\rho}^{(|\mathcal{I}|)}(t)$ for all t.

Now, consider an arbitrary call k of the Algorithm 1 at time t. By definition, $\hat{\rho}_i^{(k)}(t)$ contains all resolutions $\rho_i(t, \tau_1(t), \tau_2(t), \cdots, \tau_{k-1}(t), \tau'_k, \tau'_{k+1}, \cdots, \tau'_{|\mathcal{I}|})$ where $\tau'_k, \tau'_{k+1}, \cdots$ are arbitrarily chosen. Similarly, $\hat{\rho}_i^{(k-1)}$ contains all resolutions $\rho_i(t, \tau_1(t), \tau_2(t), \cdots, \tau_{k-2}(t), \tau'_{k-1}, \tau'_k, \tau'_{k+1}, \cdots, \tau'_{|\mathcal{I}|})$ where $\tau'_{k-1}, \tau'_k, \cdots$ are arbitrarily chosen. By observation one can see that $\hat{\rho}_i^{(k-1)} \supseteq \hat{\rho}_i^{(k)}$. Therefore, by induction $\rho_i(t) \in \hat{\rho}_i^{(k)}$ for any k and any i. Therefore, the functions $\hat{\rho}^{(j)}$ produced by Algorithm 1 are an over-approximation of ρ at every time step. □

Example 8 (Order in Highway Example)**.** *In the highway example, one can assume that any vehicle (e.g. vehicle E in Fig. 7.2) on the highway sees the actions of the vehicles in front of it (or on a slower lane when two agents are aligned) and can use those to inform its own lane change decisions. In this way, a time-varying ordering is implemented where $\mathcal{H}_i \succeq_t \mathcal{H}_j$ if and only if $\mathcal{H}_i$ is in front of $\mathcal{H}_j$ or they are aligned and $\mathcal{H}_j$ is on a slower lane compared to $\mathcal{H}_i$. This gives a total order across agents at each time t.*

For example, in Fig. 7.2 the ordering is $\mathcal{H}_{L_1} \succeq \mathcal{H}_{L_3} \succeq \mathcal{H}_E \succeq \mathcal{H}_{F_2}$. This choice is motivated by the intuition that the ego vehicle $\mathcal{H}_E$ is behind vehicles $\mathcal{H}_{L_1}$ and $\mathcal{H}_{L_3}$, so it can see their actions. As concrete examples of what Algorithm 1 outputs in this case, if $\tau_E(t) = \epsilon$, then $\hat{\rho}_E^{(E)}(t) = \{L_1\}$ when $\tau_{L_1}(t) \neq \epsilon$ and $\hat{\rho}_E^{(E)}(t) = \emptyset$ otherwise; that is, $\hat{\rho}_E^{(E)}(t)$ is not conservative, as it will observe the triggering action of L_1. Similarly, if $\tau_E(t) =$ left, $\hat{\rho}_E^{(E)}(t) = \{\rho_E(t)\}$, since E sees the triggering actions of both L_3 and L_1 and thus there is no conservativeness.

7.4 Experiments

We evaluate the flexibility and applicability of ITHA by using it to perform single-agent control in the highway driving scenario (Section 7.4.1) and multi-agent control in the parallel processing scenario (Section 7.4.2). Finally, we evaluate the conservativeness of the ITHA-based responsibility-sensitive safety rules on a real highway driving data-set (Section 7.4.3). Our software implementation is published at [1].

7.4.1 Single-agent control: highway driving

We demonstrate ITHA on the highway driving scenario, as described in Example 5, where only the ego vehicle is controlled and seeks to remain safe and responsible with respect to the uncontrolled vehicles. The ego vehicle E seeks to track a nominal velocity $v_{\text{nom}} = 15m/s$, formally solving the following receding horizon control problem at each time-step:

$$
\begin{aligned}
\min_{\substack{x_E, u_E, \\ \tau_E(t_0)}} \quad & \sum_{t=t_0+1}^{t_0+H} \|v(t) - v_{\text{nom}}\|_2^2 \\
\text{s.t.} \quad & x_E(t+1) = f_E(x_E(t), u_E(t), \tilde{d}(t)), && t = t_0+2, \ldots, \\
& && \quad t_0 + H - 1 \\
& x_E(t_0+1) \in \mathcal{X}_{E,c} \\
& x_E(t_0+1) = f_E(x_E(t_0), u_E(t_0), \tilde{d}(t_0)), && \text{if } \tau_E(t_0) = \epsilon \\
& x_E(t_0+1) = R_E^l(t_0, x_E(t_0), u_E(t_0), \bigcup\{\hat{\rho}_j(t_0)\}), && \text{if } \tau_E(t_0) = l \\
& R_j^l(t_0, x_j(t_0), u_j(t_0), \bigcup\{\hat{\rho}_j(t_0)\}) \in \mathcal{X}_{j,c}, \\
& \quad \forall u_j(t_0) \in \mathcal{U}_j, \forall j \in \hat{\rho}_{-E}(t_0), && \text{if } \tau_E(t_0) = l \\
& x_E(t_0+1) = R_E^r(t_0, x_E(t_0), u_E(t_0), \bigcup\{\hat{\rho}_j(t_0)\}), && \text{if } \tau_E(t_0) = r \\
& R_j^r(t_0, x_j(t_0), u_j(t_0), \bigcup\{\hat{\rho}_j(t_0)\}) \in \mathcal{X}_{j,c}, \\
& \quad \forall u_j(t_0) \in \mathcal{U}_j, \forall j \in \hat{\rho}_{-E}(t_0), && \text{if } \tau_E(t_0) = r
\end{aligned}
\tag{7.7}
$$

and executing $u_E(t_0)$, where the prediction horizon $H = 25$, the predicted disturbance $\tilde{d}(t) = 0$ if $t > t_0$ and $\tilde{d}(t) = -10$ if $t = t_0$, and the continuous dynamics $f_E(\cdot, \cdot, \cdot)$ are as in (7.3), where $\Delta t = 0.1$. Furthermore, $\mathcal{U}_i = [-10, 10]$ for all agents i, and we define $\hat{\rho}_{-E}(t) = \{i \in \mathcal{I} \mid \exists j \in \hat{\rho}_i(t), j \cap \{E\} \neq \emptyset\}$ as an over-approximation of the set of all agents that E can trigger at time t. We will shortly describe the specific $\hat{\rho}$ that we use in our experiments. Finally, we abuse notation to define $R_j^l(\cdot, \cdot, \cdot, \cdot)$ and $R_j^r(\cdot, \cdot, \cdot, \cdot)$ as functions which output the reset state upon making a left and right lane change, respectively.

To interpret (7.7), we note that the first constraint enforces the continuous dynamics from the second timestep onwards, and the second constraint enforces self-safety. The third constraint enforces the continuous dynamics at the first timestep if no triggering action is taken, while the fourth and sixth constraints enforce an appropriate state reset if the ego agent performs a left or right lane change, respectively. Finally, the fifth and seventh constraints enforce that all agents that are triggered by the ego vehicle's lane change action can remain safe by applying any control input.

Note that (7.7) can be represented as a mixed integer quadratic program, where $\tau_E(t_0) \in \{\epsilon, \text{left}, \text{right}\}$ can be modeled with an integer decision variable $z \in \{0, 1, 2\}$ used within a big-M formulation [24] to determine the lane change choice. To improve performance, (7.7) only seeks to enforce self-safety and responsibility at the first time-step (the system remains safe as only the input from first time-step is executed; hence, only safe actions are applied).

The uncontrolled vehicles are simulated using the Intelligent Driver Model (IDM) [154]:

$$\begin{aligned} x_i(t+1) = \;& x_i(t) + \Delta t v_i(t) \\ v_i(t+1) = \;& v_i(t) + \Delta t a \left(1 - \left(\frac{v_i(t)}{\tilde{v}_i}\right)^{\delta} - \right. \\ & \left. \left(\frac{s_0 + v_i(t) T_H + v_i(t)(v_i(t) - v_i^{\text{lead}}(t))}{2\sqrt{ab}}\right)^2\right) \end{aligned} \tag{7.8}$$

with parameters $\delta = 4, s_0 = 5, T_H = 1.5, a = b = 10$ and randomly sampled nominal velocities $\tilde{v}_i$. Here, $v_i^{\text{lead}}(t)$ refers to the velocity of agent i's lead car. We execute for 500 time-steps, solving (7.7) at each time-step; see Fig. 7.4 for a visualization of an example execution.

To illustrate the impact of different over-approximations of ρ on conservativeness, we compare control performance under these $\hat{\rho}$:

(A) The trivial over-approximation (7.6)

(B) The ordered over-approximation described in Algorithm 1, with the time-varying ordering as described in Example 8

A video showing the behavior of the $\hat{\rho}$-responsible ego vehicle when using the two different over-approximations is available here: https://youtu.be/a5IULWQYVzM. Under over-approximation (A), the ego vehicle travels 351.9 meters, while it travels 415.3 meters under over-approximation (B), averaged over 25 random initializations of the uncontrolled vehicles. In all simulations for both over-approximations, we did not experience any unsafe behavior, as guaranteed by the theory. We note that the performance improvement of the second over-approximation is a result of it being less conservative than the first. For instance, consider the example in Fig. 7.4. The simulation remains the same for both over-approximations up until time $t = 86$. At time $t = 87$, the ego agent is unable to make an advantageous left lane change to lane 3 under over-approximation (A), because if the car in lane 4 also changes to lane 3 simultaneously, it would lead to a safety violation. However, under over-approximation (B), the ego agent can safely make that lane change because the ordering implies that the car in lane 4 observes and should yield to the triggering action of the ego vehicle. A similar event occurs at time $t = 201$: under (A), the ego vehicle cannot make an advantageous switch to lane 4, because if the car in lane 5 is to simultaneously switch to lane 4, it would result in safety violations. By the end, the ego vehicle travels 161 meters further under (B) than under (A) (Fig. 7.4, bottom).

Finally, we note that while (B) outperforms (A) on average, (A) can still possibly outperform (B) for specific assignments of the uncontrolled vehicles. This occurs because (7.7) is restricted to a one-step plan for triggering actions, so the ego vehicle can make extra lane changes under (B)

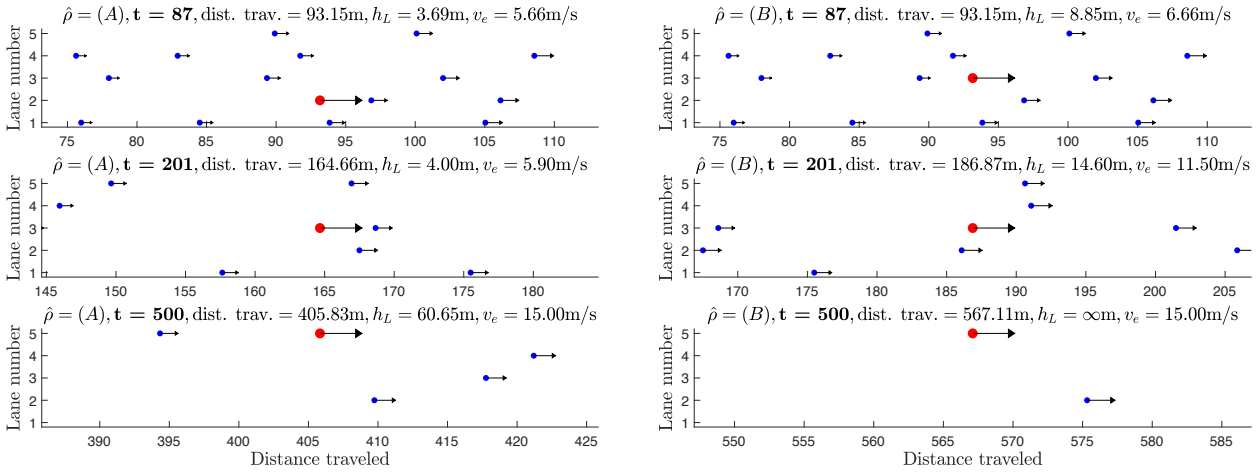


Figure 7.4: Highway driving example. *Red*: ego vehicle. *Blue*: uncontrolled vehicles. Arrow magnitudes are proportional to agent velocity. *Top row*: the ego vehicle can make an advantageous change to lane 3 under (B), but not under (A) due to a hypothetical simultaneous lane change from the lane 4 agent to lane 3. *Middle row*: the ego vehicle cannot make an advantageous lane change to lane 4 under (A) due to a hypothetical simultaneous lane change from the lane 5 agent to lane 4. *Bottom row*: At the end of the simulation, there is a large performance gap between using over-estimates (A) and (B).

that can cause it to get trapped behind a slow car without realizing that it can free itself using a long sequence of lane changes; planning triggering actions over a longer horizon would aid in escaping from these "local optima".

Overall, this experiment suggests we can use ITHA-based controllers to control an agent in a multi-agent environment with safety guarantees under limited communication, and that conservativeness of $\hat{\rho}$ can affect control performance.

7.4.2 Multi-agent control: parallel processors

We demonstrate ITHA on the parallel processor scenario, as decribed in Example 4, where we control all agents (processors). Our task is to maximize the number of accepted jobs over a finite horizon. At each time-step, we indirectly achieve this in a decentralized, receding-horizon fashion by computing control inputs and triggering inputs individually for each agent, which greedily minimize the number of remaining jobs for that agent. Formally, for each agent i, we solve the following integer program at each time-step t:

	Avg. no. accepted jobs	Avg. safety violations
ρ estimate (A)	268.6	0
ρ estimate (B)	262.2	147.64
ρ estimate (C)	302.72	0

Table 7.1: Parallel processor statistics, averaged over 25 runs.

$$\begin{aligned} \min_{u_i(t),\tau_i(t)} \quad & x_i(t+1) \\ \text{s.t.} \quad & x_i(t+1) = x_i(t) - u_i(t) + d_i(t) - \textstyle\sum_{j=1}^{|\mathcal{I}|} \tau_i^j(t) \\ & x_i(t+1) \in \mathcal{X}_{i,c} \\ & x_i(t+1) \geq 0 \\ & \tau_i^j(t) = 0,\ \forall j \notin out_{\mathcal{G}}(i) \\ & R_j(t, x_j(t), u_j(t), \textstyle\bigcup\{\hat{\rho}_j(t)\}) \in \mathcal{X}_{j,c}, \\ & \qquad \forall u_j(t) \in \mathcal{U}_j, \forall j : \tau_i^j(t) = 1 \end{aligned} \tag{7.9}$$

where $u_i(t) \in \{0,1\}$ and $\tau_i(t) \in \{0,1\}^{|\mathcal{I}|}$. Here, $|\mathcal{I}| = 10$, $n_{\text{throttle}} = 3$, and $n_{\text{overflow}} = 5$. We define an over-approximation of the set of agents that agent i can trigger at time t, $\hat{\rho}_{-i}(t)$, in the same way as in the highway example. Similar to the highway example, we will compare performance between three ρ estimates:

(A) The trivial over-approximation (7.6). Here, $\hat{\rho}_i(t) = 2^{\text{in}_{\mathcal{G}}(i)}$.

(B) An *under-approximation* $\rho_{-j}^{\text{bad}}(t) \subseteq \{i\}$, that is, when planning $\tau_i(t)$, agent i assumes no other agent will recruit j.

(C) The ordered over-approximation described in Algorithm 1, with a time-invariant priority order sorted by processor index, i.e. $\mathcal{H}_1 \succ \ldots \succ \mathcal{H}_{\mathcal{I}}$.

We simulate 25 runs, each over a horizon of 50 time-steps, and report the performance statistics in Table 7.1. In each run, we generate a random undirected connectivity graph $\mathcal{G}$, where an edge between agents i and j exists if a sample uniformly drawn from $[0,1]$ is greater than or equal to 0.1. Disturbances $d_i(t)$ are also generated randomly. Note that using the ρ estimate (A) leads to conservative performance, since there is no communication; thus, for agent i to recruit agent j, it must guarantee that agent j can remain safe if the rest of agent j's neighbors also trigger it. This overall leads to few recruitment actions, and thus many jobs are rejected. On the other hand, (B) is less conservative, but as it is an unsafe estimate of ρ (since it is an under-approximation), safety violations can occur, such as when many other processors recruit one processor during the same time-step, leading to the recruited processor exceeding n_{overflow}. As a side effect, the average number of accepted jobs is also lower under (B) since many processors are often over the throttle limit, limiting the number of incoming jobs. With the ordered contract, we can avoid this

mismanagement, more efficiently allocating jobs among the processors and preventing jobs from being unnecessarily rejected while remaining safe.

Overall, this experiment suggests that we can also use ITHA to control multiple agents in a decentralized fashion with safety guarantees, and that it is vital to select an appropriate ρ estimate to ensure safety and good performance.

7.4.3 Supervision: Evaluation of ITHA on data

Several frameworks for autonomous driving have sought to supervise a performance controller with a safety supervisor, which overrides when the performance controller may lead the system to an unsafe state. These supervisors often use invariant sets or control barrier functions to detect these safety violations. However, the usefulness of a safety supervisor is often dependent on its conservativeness, i.e. it should not unnecessarily override, as the jerkiness of changing controllers may annoy or frighten the user. To empirically demonstrate that the ITHA framework can serve as a high-quality safety supervisor that provides rigorous safety guarantees while remaining sufficiently permissive, we demonstrate that an ITHA-based supervisor achieves low override rates when supervising on a real world highway driving data-set [82].

The HighD data-set consists of trajectories of each driver's position with annotations, such as the vehicle lane and vehicle class (motorcycle, truck, or car), with data recorded at six different locations on the German Autobahn at various times of day. We use 110516 trajectories from the HighD data-set, containing a total of around 4×10^7 (x, u, d) data-point tuples. The state, input, and disturbance trajectories for each car in the data-set under the dynamics (7.3) (the v_i, h_i, v_i^L, and d_i trajectories) are generated as follows. v_i, h_i, and v_i^L are directly provided in the HighD data-set; we compute d_i via finite-differencing using the lead car velocities and $\Delta t = 0.04$ seconds, which is the provided time discretization of the data-set.

We hold out 20% of the data and use the remaining 80% as a "training set" to compute disturbance bounds. These disturbance bounds are used to compute invariant sets which are well-calibrated to the driving behavior observed in the data-set. Let the set of all disturbance trajectories in the training data-set for dynamics (7.3) be denoted $\Xi_d \doteq \{\xi_{d,i}\}_{i=1}^N$, where N is the number of trajectories in the data-set. While the data-set can be noisy, we do not perform any de-noising in this step, and instead process outliers when computing the disturbance bounds. Specifically, we process Ξ_d for outliers by only keeping the data between the 0.025- and 0.975-sample quantiles $\hat{d}_{0.025}$, $\hat{d}_{0.975}$; that is, we concatenate Ξ_d, sort the result in increasing order (i.e. obtain the order statistics of Ξ_d, denoted $\Xi_{d,(1)}, \Xi_{d,(2)}, \ldots$), and remove all disturbances belonging in the first 2.5 and last 2.5 percent of Ξ_d (this is possible, since d is scalar in (7.3)). Formally, we define the y-sample quantile, $y \in (0, 1)$, as $\hat{d}_y = \Xi_{d,(\lceil yN \rceil)}$, where N is the number of elements in Ξ_d. Let

Case	Override percentage
Self-safety	10.09%
Responsibility: Trivial $\hat{\rho}$	28.03%
Responsibility: Prioritized order $\hat{\rho}$	2.30%

Table 7.2: Override statistics for HighD data-set supervision.

this modified data-set be denoted $\hat{\Xi}_d \doteq \{d \in \Xi_d \mid d \in [\hat{d}_{0.025}, \hat{d}_{0.975}]\}$. We compute an invariant set assuming disturbances d satisfy $d \in \mathcal{D} = \mathcal{D}_i = [\hat{d}_{0.025}, \hat{d}_{0.975}]$, and use these invariant sets $\mathcal{X}_{i,c} = C_{\text{inv}}$ within an ITHA-based supervisor.

To quantify the conservativeness of using ITHA-based responsibility-sensitive safety rules to supervise highway driving, we calculate the number of times our supervisor overrides the human control input on the trajectories observed in the data-set. Specifically, we calculate the fraction of datapoints in which self-safety (as defined in Definition 20) and $\hat{\rho}$-responsibility (as defined in Definition 22) are violated; we denote this the *override rate*, reported in Table 7.2. A low override rate indicates that our supervisor will not frequently engage and is not excessively conservative, which is desirable since there are no crashes in the data-set. As mentioned, we compare between two different $\hat{\rho}$ over-approximations to evaluate responsibility: the first uses the trivial over-approximation (7.6), while the second uses the ordering contract where any agent j behind agent i in the direction of travel must yield to the triggering actions of agent i (see Example 8 for more details).

Analyzing the override percentages in Table 7.2, we observe that supervising self-safety with ITHA results in relatively low override percentages, while the ordered contract outperforms the trivial contract substantially. This is to be expected, since using the trivial over-approximation leads to overrides being counted if the ego car is changing lanes and there exists another car adjacent to the new lane with similar longitudinal position as the ego car. This is common behavior (i.e. many cars may simultaneously be at similar longitudinal positions on the highway in different lanes). Note that these override rates can be further improved by employing context-dependent invariant sets generated with disturbance bounds computed on different clusters of data (contexts), i.e. only on trajectories recorded in the fast lane, or only on trajectories recorded at rush hour. Further investigation of the impact of context-dependence on the conservativeness of ITHA-based supervisor rules is an interesting direction for future work.

Overall, this experiment suggests that an ITHA-based supervisor can obtain low override rates on a real driving dataset, indicating that driving data-sets can be used to calibrate an ITHA-based safety supervisor and that such supervisors are permissive enough to avoid excessive overrides (10% for self-safety and 2% for an appropriate responsibility contract) and act as a useful safety supervisor.

7.5 Discussion

In this section, we provide a few remarks on limitations and simple extensions of our framework:

- ITHA is appropriate in modeling systems whose individual dynamics are decoupled but have additional triggering actions for interaction. Our self-safety and responsibility rules utilize this structure to provide sufficient conditions for global safety. In comparison, existing compositional frameworks, such as [149, 101, 112, 134, 34, 138, 57, 88], that give sufficient conditions for global safety allow agents' dynamics to be coupled but do not allow for triggering actions.

- We note that our approach can be extended to guarantee safety for settings in which individual systems may have communication delays or sensor noise by leveraging recent advances in invariant set computation [64, 74, 170, 89] for systems with these imperfections.

- The increased complexity of our method over other responsibility-sensitive safety frameworks for driving (i.e. [42]) can be attributed in part to analyzing "second-order" triggers, i.e. reasoning about the set of agents which can have their feasible triggering set modified by the triggering action of another agent. In the two-lane highway driving setting, such behavior does not exist (which is what is considered in most existing responsibility-sensitive highway driving frameworks) since there are no "second-order" neighbors; however, to guarantee safety when there are more than two lanes of traffic, it is vital to consider second-order behavior.

- While we show communication and ordered triggering action selection can reduce conservativeness, it can be further reduced by communicating control inputs. We assume that agents evaluate responsibility using all inputs u that the triggered agents can apply (see Eq. (7.5)); however, if agents can communicate their state and input to the triggered agent, we can relax this "for all inputs" condition to "there exists an input" according to a similar priority.

- Finally, we note our framework allows for priorities between agents which are not fixed a priori and dynamic orders can be chosen to improve performance. For instance, in the processor example, agents with the most remaining jobs can be reassigned to have higher priority in recruiting other agents. So, there is a potential to employ distributed algorithms to select such dynamic orders.

CHAPTER 8

Conclusions

In this dissertation, we've discussed multiple methods for correctly designing adaptive controllers in a variety of settings.

In Chapter 3, we presented a method for designing controllers or estimators that adapt to missing data (e.g. packet drop events) events on their sensors. When a missing data event occurs to a dynamical system's sensor, the sensor is not able to measure the state of the system and instead produces a measurement that the controller recognizes as "faulty". This situation is important because, although many output feedback controllers make the assumption that sensor data is always available to the decision-maker (i.e., the controller), there are large classes of sensors for which a measurement is not always available. Such sensors include vision-based sensors which may occasionally lose sight of the object they are measuring or sensors which collect information over a network with lossy channels. The loss of measurements is sometimes governed by processes which we can prepare for a priori, which we encode into a missing data language. This missing data language can be incorporated into a controller design problem that guarantees the system satisfies reachability tasks (and thus any task that can be formulated with temporal logics like LTL). We use such guarantees to prove the safety of a motion plan for drones moving through a narrow channel as well as several other tasks in this chapter.

In Chapter 4, we presented a method for correctly designing adaptive controllers that can achieve tasks despite some initial uncertainty about their own dynamics. The system's dynamics are incredibly important for guaranteeing that tasks get completed because they govern how a control action at one time will guide the system closer to or further away from unsafe regions or task regions. Guaranteeing task satisfaction has traditionally been done using the theories of control barrier functions or contraction theory, but each of these methods requires that the control designer identify an often complex and high-dimensional function (the control barrier function, or the contraction metric) in order to apply them. Instead of relying on an oracle to provide such a function, our method can identify adaptive controllers using task information only. The approach relies on treating the reachable set of a linear system with a disturbance controller as a parameterized polytope. By considering these reachable sets as parameterized polytopes, we can use bilinear

constraints to make guarantees about whether or not a given controller will complete a task for any one of the unknown models in a finite set.

In Chapter 5, we presented a framework for correctly designing adaptive controllers with respect to a temporal logic formula. Temporal logic formulas are much more commonly applied and better understood than the parameter (the Exploration-Exploitation profile) defined in Chapter 4. The temporal logic formulas that we considered were written in KLTL and were defined over state- and model-based propositions. With these atomic propositions, we developed the semantics for KLTL formulae for a class of linear system with unknown parameters. These semantics were used to develop optimization conditions under which an adaptive controller would satisfy each part of the grammar individually. Several elements of the grammar could be combined as well and we shared how optimization constraints could be added to guarantee that the designed adaptive controllers satisfied these more complex tasks as well. The method for synthesizing adaptive controllers that guaranteed KLTL tasks were correctly achieved was then applied to two examples.

In Chapter 6, we presented a method for correctly designing adaptive controllers that can achieve safety tasks despite some initial uncertainty about their own dynamics. The key difference between this chapter and Chapter 4 is that the tasks we are interested in are safety tasks (so they require guarantees about all future times, not just for a finite time horizon as is done in Chapter 4). To guarantee safety of the system for all future times, a robust control invariant set is frequently employed. The robust control invariant set can be found for linear systems when the model is known, however in this setting the model is not immediately known at design time. The unknown parameters of the model are referred to as intentions in this chapter due to our choice of motivating example (a human-driven vehicle driving next to our computer-controlled vehicle). We combine a more conservative robust control invariant set with an intention estimation controller to design a controller that guarantees safety when the task is started, and can later learn the intention of the other car (and thus the model). When the model is learned, a less conservative invariant set can be used to satisfy the safety task.

In Chapter 7, we presented a new model for hybrid systems called the inter-triggering hybrid automaton. This model was developed as part of a team at the Özay group and is used to represent collections of interacting agents in the world (e.g. collections as disparate as a set of cars interacting on a highway or a set of server's sending packets to one another). For such collections of systems, it is difficult to verify the safety of their open-loop operation or to design closed-loop, decentralized control policies that can guarantee their safety because most methods use restrictive notions of safety or rely on tools that do not scale well with the number of agents in a collection. Our model allows for the problem of verification and control design to be simplified because it (i) allows for a less restrictive notion of safety that we call *responsibility-sensitive safety* and (ii) decomposes the problem in a way that eliminates several scalability issues. Responsibility-sensitive safety encodes

a notion of responsibility into safety which enforces that agents act in a way that doesn't directly cause others to be unsafe, but importantly does not require that agents reason about all possible control actions of other agents. The problem is decomposed so that each agent in the collection is only required to check the safety of its own states (self-safety) and some subset of control actions of the agents that it is responsible for (responsibility).

8.1 Future Work

In this dissertation, we successfully developed a method for synthesizing adaptive controllers that correctly satisfy temporal logic formulas. This improvement makes it much easier to apply our bilinear optimization framework for designing adaptive controllers to new tasks. Another improvement that we were interested in was improving the adaptive control tools in this work so that they could be applied to real robotic systems. This goal was attempted but a satisfactory approach was not completely developed in time to include in this dissertation. The two attempts towards this end can be summarized as:

- A guaranteed reachability pipeline for an uncertain linear system like (2.5), but where the set of feasible parameters Θ contains an infinite number of elements but is still compact, and
- An adaptive motion planning pipeline for uncertain nonlinear systems in continuous-time.

The first attempted solution was determined to be a polynomial optimization and requires a deeper understanding of semi-algebraic set containment before it can be completed. The second attempted solution was attempted using the theory of adaptive control Lyapunov functions (aCLFs) and adaptive control barrier functions (aCBFs) to guarantee safety of an adaptive controller while also achieving a reachability task. Unfortunately, generating adaptive controllers with this theory relies on the discovery of suitable aCLFs or aCBFs and there is no known systematic method for doing so. Methods for generating aCLFs and aCBFs from samples of the dynamics, previously observed trajectories, or other methods will be investigated to alleviate the difficulty of finding these functions. In addition, aCLF- and aCBF-based methods for task satisfaction require continuously differentiable control laws but the traditional quadratic programming-based, blended aCLF- and aCBF-based controllers do not always generate differentiable control laws. An investigation of how to guarantee differentiability of the controller will be done to determine if this guarantee can be made.

APPENDIX A

Missing Data Proofs

A.1 Proofs of Properties of Block Lower Triangular Matrices

Proof of Lemma 3

The first property is a trivial consequence of matrix addition, while the second property follows directly from multiplication of two block lower triangular matrices. Finally, the third property can be observed from the identity for partitioned matrix inversion of block lower triangular matrices. □

Proof of Proposition 1

Let $\bar{C}_0 \doteq \mathcal{BM}_j(\bar{C}^{(1)}) = \mathcal{BM}_j(\bar{C}^{(2)})$. We prove both the sufficient and necessary directions:

Sufficiency: Suppose that $\mathcal{BM}_j(F^{(1)}) = \mathcal{BM}_j(F^{(2)}) = \tilde{F}$. Using the fact that $\bar{C}^{(i)}$, S and $F^{(i)}$ are block lower triangular (and hence, $Q^{(i)}$ is also block lower triangular) for $i \in \{1, 2\}$ as well as Lemma 3, we have:

$$\begin{aligned}
&\mathcal{BM}_j(Q^{(1)}) \\
&= \mathcal{BM}_j[F^{(1)}(I - \bar{C}^{(1)}SF^{(1)})^{-1}] \\
&= \mathcal{BM}_j(F^{(1)})(\mathcal{BM}_j(I) - \mathcal{BM}_j(\bar{C}^{(1)})\mathcal{BM}_j(S)\mathcal{BM}_j(F^{(1)}))^{-1} \\
&= \tilde{F}(\mathcal{BM}_j(I) - \bar{C}_0\mathcal{BM}_j(S)\tilde{F})^{-1} \\
&= \mathcal{BM}_j(F^{(2)})(\mathcal{BM}_j(I) - \mathcal{BM}_j(\bar{C}^{(2)})\mathcal{BM}_j(S)\mathcal{BM}_j(F^{(2)}))^{-1} \\
&= \mathcal{BM}_j[F^{(2)}(I - \bar{C}^{(2)}SF^{(2)})^{-1}] \\
&= \mathcal{BM}_j(Q^{(2)}).
\end{aligned}$$

Necessity: Suppose that $\mathcal{BM}_j(Q^{(1)}) = \mathcal{BM}_j(Q^{(2)}) = \tilde{Q}$. First, we note the (strictly) block lower triangular properties of $\bar{C}^{(i)}$, S, $Q^{(i)}$ and $F^{(i)}$. It was shown in [147] that we can solve for $F^{(i)}$, for $i \in \{1, 2\}$ from (2.2) as:

$$F^{(i)} = (I + Q^{(i)}\bar{C}^{(i)}S)^{-1}Q^{(i)}.$$

Then, using the fact that $\bar{C}^{(i)}$, S, $Q^{(i)}$ and $F^{(i)}$ are block lower triangular for $i \in \{1, 2\}$ and Lemma 3, we find that:

$$
\begin{aligned}
&\mathcal{BM}_j(F^{(1)}) \\
&= \mathcal{BM}_j[(I + Q^{(1)}\bar{C}^{(1)}S)^{-1}Q^{(1)}] \\
&= (\mathcal{BM}_j(I) + \mathcal{BM}_j(Q^{(1)})\mathcal{BM}_j(\bar{C}^{(1)})\mathcal{BM}_j(S))^{-1}\mathcal{BM}_j(Q^{(1)}) \\
&= (\mathcal{BM}_j(I) + \tilde{Q}\bar{C}_0\mathcal{BM}_j(S))^{-1}\tilde{Q} \\
&= (\mathcal{BM}_j(I) + \mathcal{BM}_j(Q^{(2)})\mathcal{BM}_j(\bar{C}^{(2)})\mathcal{BM}_j(S))^{-1}\mathcal{BM}_j(Q^{(2)}) \\
&= \mathcal{BM}_j[(I + Q^{(2)}\bar{C}^{(2)}S)^{-1}Q^{(2)}] \\
&= \mathcal{BM}_j(F^{(2)}). \quad \square
\end{aligned}
$$

Proof of Proposition 2

The proof is similar to Proposition 1. First, consider the forward direction of the proof (sufficiency):

$$
\begin{aligned}
r^{(1)}_{1:jn} &= [(I + Q^{(1)}\bar{C}^{(1)}S)u^{(1)}_0]_{1:jn} \\
&= \mathcal{BM}_j(I + Q^{(1)}\bar{C}^{(1)}S)[u^{(1)}_0]_{1:jn} \\
&= (\mathcal{BM}_j(I) + \mathcal{BM}_j(Q^{(1)})\mathcal{BM}_j(\bar{C}^{(1)})\mathcal{BM}_j(S))[u^{(1)}_0]_{1:jn} \\
&= (\mathcal{BM}_j(I) + \mathcal{BM}_j(Q^{(2)})\mathcal{BM}_j(\bar{C}^{(2)})\mathcal{BM}_j(S))[u^{(2)}_0]_{1:jn} \\
&= \mathcal{BM}_j(I + Q^{(2)}\bar{C}^{(2)}S)[u^{(2)}_0]_{1:jn} \\
&= r^{(2)}_{1:jn},
\end{aligned}
$$

where we again applied Lemma 3 and the fact that $\bar{C}^{(i)}$, S and $Q^{(i)}$ are block lower triangular for $i \in \{1, 2\}$. The proof of the opposite direction (necessity) is similar. $\square$

A.2 Matrices and Sets for Estimator and Controller Synthesis Problems

For completeness' sake, we provide the corresponding matrices and sets for the estimator and controller synthesis problems in this section.

In the context of estimator synthesis for the system with missing data in (3.1) using the estimator structure in (3.2), the state estimation error system for the state estimation error given by $\xi(t) \doteq x(t) - \hat{x}(t)$ can be found to be of the form in (3.3) with $B_\xi \doteq I$, $u_\xi(t) \doteq u_e(t)$, $k_\xi \doteq 0$, $\mathcal{U}_\xi \doteq \mathbb{R}^m$, and the transformed output $y_\xi(t) \doteq y(t) - C\hat{x}(t)$ when $q(t) = 1$ and $y_\xi(t) \doteq \emptyset$ when $q(t) = 0$, where $\hat{x}(t)$ is known signal that can computed using (3.2).

On the other hand, the controller synthesis problem for the system with missing data in (3.1) is one with the system dynamics of the form in (3.3) with $B_\xi \doteq B$, $k_\xi \doteq k$, $u_\xi(t) \doteq u(t)$,

$y_\xi(t) \doteq y(t)$ and $\xi(t) \doteq x(t)$, as well as $\mathcal{U}_\xi \doteq \mathcal{U}$. Moreover, for the tracking control problem with a given desired trajectory $x_d(t_0), x_d(t_0+1), \ldots, x_d(t_0+T)$ and associated desired inputs $u_d(t_0), u_d(t_0+1), \ldots, u_d(t_0+T-1)$, the corresponding system dynamics takes the form in (3.3) with $B_\xi \doteq B$, $k_\xi \doteq 0$, $u_\xi(t) \doteq u(t) - u_d(t)$, $y_\xi(t) \doteq y(t) - Cx_d(t)$ and $\xi(t) \doteq x(t) - x_d(t)$, as well as $\mathcal{U}_\xi \doteq \{u_\xi(t) \in \mathbb{R}^m \mid u_\xi(t) + u_d(t) \in \mathcal{U}\}$.

A.3 Proofs of Main Results

Proof of Theorem 1

This follows directly from Propositions 1 and 2 as well as the invertibility of the mappings (2.2) and (2.3). □

Proof of Theorem 2

For a given prefix-based feedback law $\{(F^{(i)}, u_0^{(i)})\}_{i=1}^{|\mathcal{L}|}$, the transformed state trajectory $\xi^{(i)} = [\xi^{(i)}(t_0)^\intercal, \ldots, \xi^{(i)}(t_0+T)^\intercal]^\intercal$ under the i-th missing data pattern can be written as a nonlinear function of $\{(F^{(i)}, u_0^{(i)})\}_{i=1}^{|\mathcal{L}|}$ just by plugging in the transformed input term (3.7). After applying a change of variables via the mapping in Theorem 1, we can express $\xi^{(i)}$ as a linear function of $\{(Q^{(i)}, r^{(i)})\}_{i=1}^{|\mathcal{L}|} \in \mathcal{Q}(\mathcal{L})$ as in (3.15). Since the equalized recovery condition can also be written as linear constraints in $\xi^{(i)}$ that should hold for all initial states satisfying M_1 bound and for all possible noise values, problem (3.14) is a robust linear program, whose feasibility is equivalent to the existence of the desired estimator/controller. Finally, the gains of the prefix-based feedback law are obtained by applying the inverse of the mapping in Theorem 1. □

Proof of Theorem 3

In this proof, we will convert the semi-infinite constraints in Theorem 2 into linear constraints by leveraging the robust optimization approach in [20, 23]. Since we will repeat the same robustification process for each $i \in [1, |\mathcal{L}|]$ in (3.14), we will drop the dependence on i in the following.

We first rewrite the infinity norm expressions in (3.14) as linear inequalities and substitute the

expression for ξ in terms of w, v, $\xi(t_0)$ and r using the equations in Theorem 2, as follows:

$$\|\xi\| \leq M_2 \Rightarrow \begin{bmatrix} I \\ -I \end{bmatrix} \eta = \begin{bmatrix} I \\ -I \end{bmatrix} (G \begin{bmatrix} w \\ v \\ \xi(t_0) \end{bmatrix} + Sr + (I + SQ\bar{C})H\tilde{f}) \leq M_2 \mathbb{1}, \tag{A.1}$$

$$\|R_T\xi\| \leq M_1 \Rightarrow \begin{bmatrix} I \\ -I \end{bmatrix} R_T\eta = \begin{bmatrix} I \\ -I \end{bmatrix} R_T(G \begin{bmatrix} w \\ v \\ \xi(t_0) \end{bmatrix} + Sr + (I + SQ\bar{C})H\tilde{f}) \leq M_1 \mathbb{1}, \tag{A.2}$$

$$\|u_\xi + u_d\| \leq \eta_u \Rightarrow \begin{bmatrix} I \\ -I \end{bmatrix} (u_\xi + u_d) = \begin{bmatrix} I \\ -I \end{bmatrix} (\tilde{G} \begin{bmatrix} w \\ v \\ \xi(t_0) \end{bmatrix} + r + Q\bar{C}H\tilde{f} + u_d) \leq \eta_u \mathbb{1}, \tag{A.3}$$

$$\begin{matrix} \|w\| \leq \eta_w, \\ \|v\| \leq \eta_v, \\ \|\xi(t_0)\| \leq M_1, \end{matrix} \Rightarrow \begin{bmatrix} I & 0 & 0 \\ -I & 0 & 0 \\ 0 & I & 0 \\ 0 & -I & 0 \\ 0 & 0 & I \\ 0 & 0 & -I \end{bmatrix} \begin{bmatrix} w \\ v \\ \xi(t_0) \end{bmatrix} \leq \begin{bmatrix} \eta_w \mathbb{1} \\ \eta_v \mathbb{1} \\ M_1 \mathbb{1} \end{bmatrix}, \tag{A.4}$$

where $G \doteq \begin{bmatrix} (I + SQ\bar{C})H & SQN & (I + SQ\bar{C})J \end{bmatrix}$, $\tilde{G} \doteq \begin{bmatrix} Q\bar{C}H & QN & Q\bar{C}J \end{bmatrix}$ and $R_T \doteq \begin{bmatrix} 0_{n \times nT} & I_n \end{bmatrix}$. Then, leveraging the robust optimization approach in [20, 23], we can convert the constraints of (A.1) and (A.2) "for all disturbances w, v and $\xi(t_0)$", i.e., subject to (A.4), into linear constraints with dual matrix variables Π_1, Π_2 and Π_3 for each $i \in [1, |\mathcal{L}|]$. □

A.4 Detectability Related Proofs

Proof of Proposition 4

In this proof, we use an unobservable subspace argument to show that for initial conditions starting in two important subsets of the state space, equalized recovery parameters can be found and then by a direct sum argument we can conclude that such parameters can be found for any initial condition in the state space of a detectable system.

First, by assumption, any initial condition in the unobservable subspace $x_0 \in \mathcal{U}_0(q)$ asymptotically converges to zero. Convergence to zero implies that there exists a time $t^{(1)}$ such that the zero estimator (the estimator that always returns the zero vector) satisfies equalized recovery for time

horizon $t^{(1)}$, any recovery level M_1, and a finite intermediate level M_2.

Next, note that if there exists a time t' such that $y(t', x_0, q)$ (the output at time t' caused by initial condition x_0) is different from the output of any other initial condition, then equalized recovery is feasible. The estimator that achieves equalized recovery has time horizon t', any recovery level, and a finite intermediate level that depends on the dynamics. By definition, all initial conditions x_0 in the orthogonal complement of the unobservable subspace $x_0 \in \mathcal{U}_0(q)^\perp$ of (3.23) have such a time $t^{(2)}$ or else contradict the definition of $\mathcal{U}_0(q)^\perp$.

Finally, because any initial condition must be in the direct sum of $\mathcal{U}_0(q)$ and $\mathcal{U}_0(q)^\perp$, the proof is complete. For any initial condition $x_0 = x_0' + x_0''$ where $x_0' \in \mathcal{U}_0(q)$ and $x_0'' \in \mathcal{U}_0(q)^\perp$, equalized recovery is satisfied with time horizon $t^{(1)} + t^{(2)}$, a finite intermediate level, and a recovery level dependent on the choice of $t^{(1)}$ and $t^{(2)}$. □

Proof of Proposition 5

We show that equalized recovery is indeed a superset of detectability by providing a simple example. Consider the following scalar linear system:

$$\begin{aligned} x(t+1) &= x(t), \\ y(t) &= 0. \end{aligned}$$

It is trivial to see that this system does not satisfy detectability according to Definition 8 because $x(t, x_0, q)$ for any $x_0 \neq 0$ does not tend to zero. However, equalized recovery is trivially satisfied for any T, M_1 and $M_2 \geq M_1$. □

APPENDIX B

Defining the Resets and Resolution Function for the Highway Example

In order to define the reset map and the resolution function for the highway example, we need to introduce global states in a global coordinate system for the overall system. Both the reset map and the resolution function can be expressed as time-invariant functions of these global states. On the other hand, we model the states of individual vehicles in ITHA in some local coordinates in (7.3), Global-state dependent reset maps and resolution functions can be converted to time-varying variants on the individual states. Since the self-safety and responsibility conditions are defined for time-varying reset maps and resolution functions that depend on individual states, our safety results are still applicable with this conversion. This section clarifies what information each vehicle would need at a given time to compute $\hat{\rho}$ and to comply with condition (7.5).

Let agent i's global state be defined as $\bar{x}_i = \begin{bmatrix} v_i & p_i & \ell_i \end{bmatrix}^\top \in \bar{\mathcal{X}}_a = [0, v_{max}] \times [0, \infty) \times \{1, ..., n_\ell\}$ where $\ell_i \in \{1, ..., n_\ell\}$ is the current lane's number, p_i is the longitudinal position in the current lane, and v_i is the current longitudinal velocity in the direction of the current lane (i.e. the same value in (7.3)). Here we take the state spaces $\bar{\mathcal{X}}_a$ of the vehicles to be identical for simplicity. Also, by convention, we take the rightmost lane to have value 1. Let $\{\bar{x}_i\}_{i=1}^{|\mathcal{I}|}$ be the collection of all agents' global states, with $\bar{\mathcal{X}}_g \triangleq \bar{\mathcal{X}}_a^{|\mathcal{I}|}$ denoting this state space, and $\bar{\mathcal{X}}_S$ denoting the restriction of this space to agents $S \subset \mathcal{I}$.

The dynamics of the global state can be written as:

$$\begin{aligned} \bar{x}_i(t+1) &= \begin{bmatrix} 1 & 0 & 0 \\ \Delta t & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bar{x}_i(t) + \begin{bmatrix} \Delta t \\ 0 \\ 0 \end{bmatrix} u_i(t) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_{\tau_i}(t), \\ &\triangleq \bar{f}_i(\bar{x}_i(t), u_i(t), u_{\tau_i}(t)), \end{aligned} \tag{B.1}$$

where u_i is the acceleration, i.e., the same control input in the ITHA representation, and

$$u_{\tau_i}(t) = \begin{cases} 1 & \tau_i(t) = \text{left}, \\ 0 & \tau_i(t) = \text{stay}, \\ -1 & \tau_i(t) = \text{right}. \end{cases}$$

Note that it is more intuitive to define control invariant sets over the local state space defined in (7.3) rather than in the global state space of the global coordinates since safety depends only on inter-vehicle distances. Thus, local reasoning is sufficient for the proofs of this dissertation. However, to define the resets, local states are not sufficient.

Let us define the following nearest leader of agent i operator $L_i : \bar{\mathcal{X}}_a \times \bigcup_{S \subset \mathcal{I}} \bar{\mathcal{X}}_S \to \mathcal{I} \cup \emptyset$. It is defined by the following simple optimization:

$$\begin{aligned} L_i(\bar{x}_i, \{\bar{x}_j\}_{j \in S}) = \quad & \underset{j \in \mathcal{S} \setminus \{i\}}{\text{arg min}} \quad |p_i - p_j| \\ & \text{subject to} \quad p_i \leq p_j \\ & \qquad\qquad\ \ \ell_j = \ell_i \end{aligned} \tag{B.2}$$

In words, it finds the closest lead car within the set S to agent i on the same lane with it. Assuming that two cars cannot occupy the same point on the highway (in this case, the same longitudinal position and lane), the value of argmin will always be either a singleton or the empty set. However, even if it is not a singleton, picking any minimizer works for the purposes of the next lemma, which relates the global states to individual (local) ones.

Lemma 11. *There exists a unique mapping from the global state $\{\bar{x}_i\}_{i=1}^{|\mathcal{I}|}$ to the states $\{x_i\}_{i=1}^{|\mathcal{I}|}$ of the individual vehicles in the ITHA representation.*

Proof. To map the global state $\{\bar{x}_i\}_{i=1}^{|\mathcal{I}|}$ to the ITHA states $\{x_i\}_{i=1}^{|\mathcal{I}|}$, consider the mapping for one agent $\mathcal{H}_i$:

$$x_i = \begin{cases} \begin{bmatrix} v_i \\ \infty \\ v_{max} \end{bmatrix} & L_i = \emptyset \\ \begin{bmatrix} v_i \\ p_{L_i} - p_i \\ v_{L_i} \end{bmatrix} & \text{otherwise} \end{cases} \tag{B.3}$$

where L_i is the abbreviation of $L_i(\bar{x}_i, \{\bar{x}_j\}_{j \in \mathcal{I}})$. □

We define a localized version of the mapping (B.3), denoted $\mathcal{G}_S : \bar{\mathcal{X}}_S \to \prod_{i\in S} \mathcal{X}_i$, to be the mapping when L_i is taken to be $L_i(\bar{x}_i, \{\bar{x}_j\}_{j\in S})$. We denote by $\mathcal{G}_S^i : \bar{\mathcal{X}}_S \to \mathcal{X}_i$ the component of $\mathcal{G}_S$ corresponding to agent $\mathcal{H}_i$.

Now we will define the resolution functions $\rho_i : \bar{\mathcal{X}}_g \times \mathcal{T}_1 \times \cdots \times \mathcal{T}_{|\mathcal{I}|} \to 2^{\mathcal{I}}$ and reset maps $R_i : \bar{\mathcal{X}}_g \times \mathcal{X}_i \times \mathcal{U}_i \times 2^{\mathcal{I}} \to 2^{\mathcal{X}_i}$ that depend on the global quantities[1]. We have

$$\begin{aligned}
&\rho_i : (\{\bar{x}_i\}_{i=1}^{|\mathcal{I}|}, \tau_1, \cdots, \tau_{|\mathcal{I}|}) \mapsto \\
&\quad \{j \in \mathcal{I} \mid \tau_j \neq \text{stay}, \ell_j = \ell_i, p_j \geq p_i\} \cup \\
&\quad \{j \in \mathcal{I} \mid \tau_j \neq \text{stay}, \ell_j + u_{\tau_j} = \ell_i + u_{\tau_i}, p_j + v_j \Delta t \geq p_i + v_i \Delta t\}.
\end{aligned}$$

In words, the resolution function ρ_i maps the global state and the triggering actions to the set of all agents that use a non-null triggering action that are on the same lane with agent i and ahead of it and those whose triggering action will put them on the same lane with agent i ahead of it in the next step. Then, the trivial resolution over-approximation $\hat{\rho}_i(t)$ used in our experiments considers, in the worst-case, all agents ahead of agent i on the same lane at time t and all agents that can be on the same lane with agent i ahead of it at time $t+1$. However, many of these combinations S of reset-triggering agents appearing as an output of $\hat{\rho}_i(t)$ lead to the same reset value, as defined by:

$$R_i(\{\bar{x}_i\}_{i=1}^{|\mathcal{I}|}, x_i, u_i, S) = \mathcal{G}_{S'}^i(\{\bar{f}_j(\bar{x}_j, u_j, u_{\tau_j})\}_{j\in S'}),$$

where $S' = \{j \in \mathcal{I} \mid \ell_j + u_{\tau_j} = \ell_i + u_{\tau_i}, p_j + v_j \Delta t \geq p_i + v_i \Delta t\}$. By construction, $\tau_j = \text{stay}$ for $j \in S' \setminus S$. Therefore, the triggering actions of the agents in $S' \cap S$ is sufficient for estimating R_i at time t, in addition to the estimates of other arguments. Note that the first set in the ρ_i definition does not directly seem to contribute to the reset map but it captures the agents leaving in front of agent i, which in turn affect who the lead car will be in the next step. At run-time, the local controller γ_i does not need the knowledge of the entire global states but needs to know the lanes and relative positions $(p_i(t) - p_j(t))$ and relative velocities $(v_i(t) - v_j(t))$ of agents j for which $i \in \hat{\rho}_j(t)$.

[1]To be precise, the actual value the state is reset to, depends on the triggering actions, states, and inputs of agents in ρ_i rather than the agents' indices as mentioned in footnote 1.

APPENDIX C

The Difference between KLTL and LTL

In this appendix, we discuss the difference between KLTL and LTL. To do this, we introduce a Transition System with partial observability and analyze how the KLTL semantics are interpreted over this.

C.1 Transition Systems with Partially Visible States

Consider the following system:

Definition 23 (Partially Observed Transition System)**.** *A* transition system with outputs *TS is a tuple* $(S, Act, \rightarrow, I, AP, L, Y, H)$ *where*

- S *is a set of states,*
- Act *is a set of actions,*
- $\rightarrow \subseteq S \times Act \times S$ *is a transition relation,*
- $I \subseteq S$ *is a set of initial states,*
- AP *is a set of atomic propositions,*
- $L : S \rightarrow 2^{AP}$ *is a labelling function,*
- Y *is a set of observable outputs, and*
- $H : S \rightarrow 2^{Y}$ *is an observation function.*

TS is called *finite* if S, Act, AP, and Y are finite.

Definition 24 (Path Fragment)**.** *A finite path fragment* $\hat{\pi}$ *of* TS *is a finite state sequence* $s_0 s_1 ... s_n$ *such that* $s_i \in Post(s_{i-1})$ *for all* $0 < i \leq n$ *where* $n \geq 0$*. An infinite path fragment* π *is an infinite state sequence* $s_0 s_1 s_2 ...$ *such that* $s_i \in Post(s_{i-1})$ *for all* $i > 0$*.*

Definition 25. *An infinite path fragment $\pi = s_0 s_1 ... s_n$ that is initial (i.e. $s_0 \in I$) will be called a* path.

Let the set of all paths for a given transition system with outputs TS be the set Paths(TS).

Importantly, we will frequently discuss the output trajectory of a transition system with outputs. This is defined as follows:

Definition 26 (Output Trajectory)**.** *An alternating sequence of outputs and inputs $y_0 u_0 y_1 u_1 y_2 ...$ is an output trajectory of a transition system with outputs $TS = (S, Act, \rightarrow, I, AP, L, Y, H)$ if and only if there exists a path fragment $\pi = s_0 s_1 s_2 ...$ where*

- $y_k \in H(s_k)$ *for all* $k > 0$*, and*
- $(s_k, u_k, s_{k+1}) \in \rightarrow$ *for all* $k > 0$*.*

Let the set of all output trajectories for a given transition system with outputs TS be the set OPaths(TS).

We overload the observation function H to also operate on paths as follows. For any path $\pi = s_0 s_1 s_2 ...$, the observation function $H : \text{Paths}(TS) \rightarrow 2^{\text{OPaths}(TS)}$ is defined as

$$H(\pi) = \left\{ o = y_0 u_0 y_1 u_1 y_2 ... \in \text{OPaths}(TS) \;\middle|\; \begin{array}{ll} y_i \in H(s_0) & \forall i \\ (s_i, u_i, s_{i+1}) \in \rightarrow & \forall i \end{array} \right\}.$$

Definition 27 (Trace)**.** *Let $TS = (S, Act, \rightarrow, I, AP, L, Y, H)$ be a transition system with outputs and without terminal states. The* trace *of the infinite path fragment $\pi = s_0 s_1 ...$ is defined as $trace(\pi) = L(s_0)L(s_1)....$ The* trace *of the finite path fragment $\hat{\pi} = s_0 s_1 ... s_n$ is defined as $trace(\hat{\pi}) = L(s_0)L(s_1)...L(s_n)$.*

Let the set of all traces for a given transition system with outputs TS be the set Traces(TS).

C.2 KLTL Interpretation for Partially Observed Transition Systems

In this section, we develop a form of KLTL for the partially observed transition system in Definition 23. This form of KLTL is analagous to the definition from Chapter 5. It will use the same grammar, but its semantics will be slightly changed.

The grammar is exactly as is presented in Definition 12. For the sake of readability, the grammar is reproduced here:

Definition 28 (KLTL Grammar, Originally Definition 12)**.** *The grammar of KLTL is as follows:*

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc \varphi \mid \varphi \mathcal{U} \varphi \mid \mathcal{K}\varphi \tag{C.1}$$

in which $p \in AP$ is an atomic proposition, while $\bigcirc$ and $\mathcal{U}$ are the "next" and "until" operators. Formulas of the type $\mathcal{K}\varphi$ are read as "the system knows that the formula φ holds".

The semantics of KLTL will be similar to the one presented in Chapter 5. The things that will change will mostly be the naming conventions of objects.

Definition 29 (KLTL Semantics for Systems With Unknown Parameters)**.** *For any path $\pi = s_0 s_1 s_2 ...$ of a Transition System with Outputs TS and an associated output trajectory $o \in H(\pi)$, interpret the KLTL operators as follows:*

- $(\pi, o, i) \models p$ *if* $p \in L(s_i)$
- $(\pi, o, i) \models \neg p$ *if* $(\pi, o, i) \not\models p$
- $(\pi, o, i) \models \varphi_1 \wedge \varphi_2$ *if* $(\pi, o, i) \models \varphi_1$ *and* $(\pi, o, i) \models \varphi_2$
- $(\pi, o, i) \models \bigcirc\varphi$ *if* $(\pi, o, i+1) \models \varphi$
- $(\pi, o, i) \models \varphi_1 U \varphi_2$ *if* $\exists j \geq 0$ *such that* $(\pi, o, j) \models \varphi_2$ *and* $\forall 0 \leq k < j$, $(\pi, o, k) \models \varphi_1$,
- $(\pi, o, i) \models \mathcal{K}\varphi$ *if for all* $\pi' \in Paths(TS)$ *s.t.* $\pi' \sim_o \pi$, *we have* $(\pi', o, i) \models \varphi$.

The similarity relation $\sim$ is defined as follows: Two traces π, π' are similar $\pi \sim_o \pi'$ if and only if $o \in H(\pi) \cap H(\pi')$.

C.3 Example 1: Eventually Learning

In this section, we consider the difference in interpretation of the following formulae:

$$\varphi_1 = \bigcirc a \vee \bigcirc b$$

and

$$\varphi_2 = \bigcirc(\mathcal{K}a) \vee \bigcirc(\mathcal{K}b).$$

Consider the following transition system with outputs TS_1:

Example 9. $TS_1 = (S_1, Act_1, \rightarrow_1, I_1, AP_1, L_1, Y_1, H_1)$ *where:*

- $S_1 = \{s_0, s_1, s_2, s_1', s_2'\}$

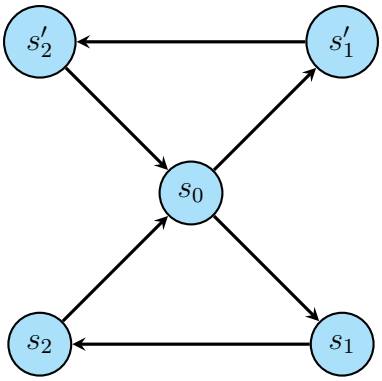


Figure C.1: An illustration of the transition system TS_1. Black arrows indicate the existence of one transition to another.

- $Act_1 = \{\alpha\}$
- $\rightarrow_1$ *is defined as is shown in Fig. C.1*
- $I_1 = \{s_0\}$
- $AP_1 = \{a, b\}$
- L_1 *is defined as:*
  - $L(s_0) = \emptyset$
  - $L(s_1) = L(s_2') = \{a\}$
  - $L(s_1') = L(s_2) = \{b\}$
- $Y_1 = \{y_0\}$ *and*
- $H_1(s) = \{y_0\}$ *for all* $s \in S_1$.

Proposition 23. *Example 10 satisfies φ_1 but does not satisfy φ_2.*

Proof. The proof of this proposition can be sketched as follows. Firstly, we can show that φ_1 is satisfied by writing down all of the possible trace prefixes of the system TS_1 with length three. In this case, there are only two possible prefixes: $\emptyset\{a\}\{b\}$ or $\emptyset\{b\}\{a\}$. By this brute force method, one can show that φ_1 is satisfied.

Secondly, Example 10 uses the same output for all states and so it is never possible to know which state of the transition system (in this case s_1 or s_1') we have reached. □

One can also consider the pair of formula templates:

$$\varphi_1' = \bigcirc^{3k} a \vee \bigcirc^{3k} b$$

and

$$\varphi_2' = \bigcirc^{3k} (\mathcal{K} a) \vee \bigcirc^{3k} (\mathcal{K} b).$$

For any value of k, this pair of formulas will have the same.

This example shows the semantic difference between two sets of formulas, but it also shows a fundamental limitation of LTL. LTL can not express this property with a label because the property is a function of infinitely many other paths in the transition system (i.e. formula φ_2' is a hyperproperty). If one wanted to create a "hack" to define such hyperproperties using LTL anyway, then they might be able to for this example but it would involve "expanding" the state space to include new states using memory.

In the next section, we will discuss an example for which no clever relabelling exists which LTL can use to satisfy a property.

C.4 Example 2: The Role of Inputs

In this section, we discuss an example that shows how KLTL can be used to express reachability tasks that LTL can not.

The formula we are interested in is as follows:

$$\varphi_3 = \Diamond \mathcal{K} a.$$

Consider the following transition system with outputs TS_2:

Example 10. $TS_2 = (S_2, Act_2, \rightarrow_2, I_2, AP_2, L_2, Y_2, H_2)$ *where:*

- $S_2 = \{s_0, s_1, s_2, s_3, s_4, s_1', s_2', s_3', s_4', s_1'', s_2'', s_3'', s_4''\}$
- $Act_2 = \{\alpha, \beta\}$
- $\rightarrow_2$ *is defined as is shown in Fig. C.2*
- $I_2 = \{s_0\}$
- $AP_2 = \{a\}$

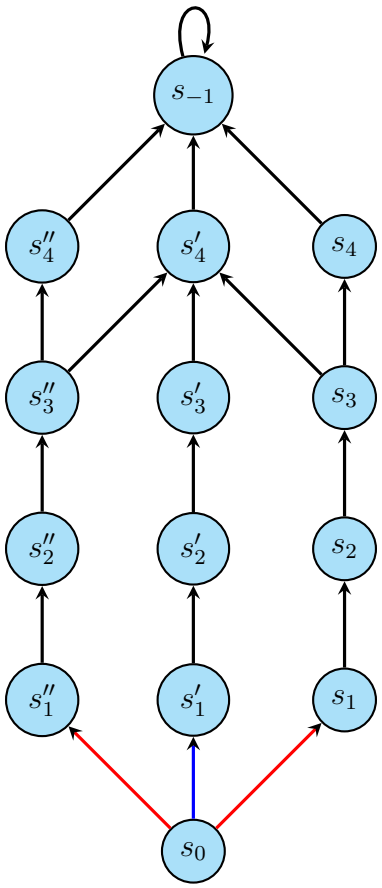


Figure C.2: An illustration of the transition system TS_2. The red arrows represent transitions under the control input α only, the blue arrows represent transitions under the action β only, and the black arrows represent possible transitions under any action in Act.

- L_2 *is defined as:*

$$L_2(s) = \begin{cases} \{a\} & s = s_4' \\ \emptyset & \textit{otherwise} \end{cases}$$

- $Y_2 = \{y_0\}$ *and*
- $H_2(s) = \{y_0\}$ *for all* $s \in S_2$.

For the above system, consider three paths:

- $\pi^{(1)} = s_0 s_1 s_2 s_3 s_4' (s_{-1})^\omega$,
- $\pi^{(2)} = s_0 s_1' s_2' s_3' s_4' (s_{-1})^\omega$, and
- $\pi^{(3)} = s_0 s_1'' s_2'' s_3'' s_4' (s_{-1})^\omega$.

Note that all three paths satisfy the temporal logic formula $\varphi_3' = \Diamond a$ (or $\varphi_3'' = \bigcirc^4 a$).

But, let us consider whether or not each path can satisfy the formula φ_3. To do this, we need to have knowledge of the output trajectories. Some example output trajectories for each path are:

- $H(\pi^{(1)})$ contains
  - $o^{(1)} = y_0 \alpha y_0 a' y_0 a'' y_0 (a''' y_0)^\omega$ where $a', a'', a''' \in$ Act
- $H(\pi^{(2)})$ contains
  - $o^{(2)} = y_0 \beta y_0 a' y_0 a'' y_0 (a''' y_0)^\omega$ where $a', a'', a''' \in$ Act
- $H(\pi^{(3)}) = H(\pi^{(1)})$.

Given these output trajectories, we can now evaluate whether or not each pair, $(\pi^{(i)}, o^{(i)})$, satisfies φ_3 for each values of i. Interestingly, only one (1) of the three pairs satisfies the formula. Only $(\pi^{(2)}, o^{(2)})$ satisfies φ_3.

BIBLIOGRAPHY

[1] Compositional safety rules for inter-triggering hybrid automata (codeocean). https://doi.org/10.24433/CO.3247007.v1.

[2] Correct-by-construction design of adaptive cruise control with control barrier functions under safety and regulatory constraints. June 2022.

[3] S. Aguilera, M. A. Murtaza, Y. Zhao, and S. Hutchinson. Mass estimation of a moving object through minimal manipulation interaction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6461–6467, 2021.

[4] S. Amin, A. A. Cárdenas, and S. S. Sastry. Safe and secure networked control systems under denial-of-service attacks. In R. Majumdar and P. Tabuada, editors, *Hybrid Systems: Computation and Control*, pages 31–45, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[5] M. Anand, V. Murali, A. Trivedi, and M. Zamani. Formal verification of control systems against hyperproperties via barrier certificates, 2021.

[6] S. M. Asghari and A. Nayyar. Dynamic teams and decentralized control problems with substitutable actions. *IEEE Transactions on Automatic Control*, 62(10):5302–5309, 2016.

[7] A. Aspeel, D. Dasnoy, R. M. Jungers, and B. Macq. Optimal intermittent measurements for tumor tracking in x-ray guided radiotherapy. In *Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling*, volume 10951, page 109510C. International Society for Optics and Photonics, 2019.

[8] A. Aspeel, K. Rutledge, R. M. Jungers, B. Macq, and N. Özay. Optimal control for linear networked control systems with information transmission constraints, 2021.

[9] K. Astrom and B. Wittenmark. Self-tuning controllers based on pole-zero placement. In *IEE Proceedings D-Control Theory and Applications*, volume 127, pages 120–130. IET, 1980.

[10] N. Athanasopoulos, K. Smpoukis, and R. M. Jungers. Invariant sets analysis for constrained switching systems. *IEEE Control Systems Letters*, 1(2):256–261, Oct 2017.

[11] M. Athans. Survey of decentralized control methods. *NASA. Ames Res. Center Large-Scale Dyn. Systems*, 1975.

[12] Y. Bai and K. Mallik. Accurate abstractions for controller synthesis with non-uniform disturbances. In *International Conference on Formal Engineering Methods*, pages 297–307. Springer, 2020.

[13] D. Baidya and R. G. Roy. Speed control of dc motor using fuzzy-based intelligent model reference adaptive control scheme. In R. Bera, S. K. Sarkar, and S. Chakraborty, editors, *Advances in Communication, Devices and Networking*, pages 729–735, Singapore, 2018. Springer Singapore.

[14] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press, 2008.

[15] L. Bakule. Decentralized control: Status and outlook. *Annual Reviews in Control*, 38(1):71–80, 2014.

[16] G. Battistelli, A. Benavoli, and L. Chisci. State estimation with remote sensors and intermittent transmissions. *Systems and Control Letters*, 61(1):155 – 164, 2012.

[17] C. Belcastro. Parametric uncertainty modeling: an overview. In *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No.98CH36207)*, volume 2, pages 992–996 vol.2, 1998.

[18] C. Belta and S. Sadraddini. Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):115–140, 2019.

[19] A. Bemporad, M. Heemels, M. Johansson, et al. *Networked control systems*, volume 406. Springer, 2010.

[20] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

[21] D. Bertsekas and I. Rhodes. Recursive state estimation for a set-membership description of uncertainty. *IEEE Transactions on Automatic Control*, 16(2):117–128, 1971.

[22] D. P. Bertsekas. Infinite time reachability of state-space regions by using feedback control. *TAC*, 17(5):604–613, 1972.

[23] D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[24] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997.

[25] S. Bharadwaj, S. Carr, N. Neogi, and U. Topcu. Decentralized control synthesis for air traffic management in urban air mobility. *IEEE Transactions on Control of Network Systems*, 8(2):598–608, 2021.

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

[27] F. Blanchini and M. Sznaier. A convex optimization approach to fixed-order controller design for disturbance rejection in siso systems. *IEEE Transactions on Automatic Control*, 45(4):784–789, April 2000.

[28] F. Blanchini and M. Sznaier. A convex optimization approach to synthesizing bounded complexity ℓ_∞ filters. *IEEE Trans. on Automatic Control*, 57(1):216–221, 2012.

[29] G. V. Bochmann and C. A. Sunshine. *A Survey of Formal Methods*, pages 561–578. Springer US, Boston, MA, 1982.

[30] U. Bodkhe, D. Mehta, S. Tanwar, P. Bhattacharya, P. K. Singh, and W.-C. Hong. A survey on decentralized consensus mechanisms for cyber physical systems. *IEEE Access*, 8:54371–54401, 2020.

[31] R. Bozianu, C. Dima, and E. Filiot. Safraless synthesis for epistemic temporal specifications. In *International Conference on Computer Aided Verification*, pages 441–456. Springer, 2014.

[32] Z. A. Caddick and B. M. Rottman. Motivated reasoning in an explore-exploit task. *Cognitive Science*, 45(8):e13018, 2021.

[33] K. Chatterjee, T. A. Henzinger, J. Otop, and A. Pavlogiannis. Distributed synthesis for ltl fragments. In *2013 Formal Methods in Computer-Aided Design*, pages 18–25. IEEE, 2013.

[34] Y. Chen, J. Anderson, K. Kalsi, S. H. Low, and A. D. Ames. Compositional set invariance in network systems with assume-guarantee contracts. In *2019 American Control Conference (ACC)*, pages 1027–1034. IEEE, 2019.

[35] S. Cheong and I. R. Manchester. Input design for discrimination between classes of lti models. *Automatica*, 53:103–110, 2015.

[36] L. Chisci, P. Falugi, and G. Zappa. Gain-scheduling mpc of nonlinear systems. *International Journal of Robust and Nonlinear Control*, 13(3-4):295–308, 2003.

[37] Z. Chu, Y. Ma, Y. Hou, and F. Wang. Inertial parameter identification using contact force information for an unknown object captured by a space manipulator. *Acta Astronautica*, 131:69–82, 2017.

[38] E. M. Clarke, O. Grumberg, and D. A. Peled. Model checking. 1999.

[39] A. Colombo, M. Bahraini, and P. Falcone. Measurement scheduling for control invariance in networked control systems. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 3361–3366. IEEE, 2018.

[40] M. Coon and D. Panagou. Control strategies for multiplayer target-attacker-defender differential games with double integrator dynamics. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1496–1502, 2017.

[41] E. Dallal and P. Tabuada. Decomposing controller synthesis for safety specifications. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 5720–5725. IEEE, 2016.

[42] R. De Iaco, S. L. Smith, and K. Czarnecki. Universally safe swerve manoeuvres for autonomous driving. *arXiv preprint arXiv:2001.11159*, 2020.

[43] E. De Santis, M. D. Di Benedetto, and L. Berardi. Computation of maximal safe sets for switching systems. *IEEE Trans. on Autom. Control*, 49(2):184–195, 2004.

[44] A. R. de Souza, D. Efimov, T. Raïssi, and X. Ping. Robust output feedback mpc: An interval-observer approach. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 2529–2534. IEEE, 2020.

[45] D. Del Vecchio. A partial order approach to discrete dynamic feedback in a class of hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 159–173. Springer, 2007.

[46] Y. Ding, F. Harirchi, S. Z. Yong, E. Jacobsen, and N. Ozay. Optimal input design for affine model discrimination with applications in intention-aware vehicles. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 297–307. IEEE, 2018.

[47] J. Farkas. Theorie der einfachen ungleichungen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1902(124):1–27, 1902.

[48] A. Feldbaum. Dual-control theory i-ii. *Automation and Remote Control 21*, 9(11):1961, 1960.

[49] F. Felicioni, N. Jia, Y.-Q. Song, and F. Simonot-Lion. Impact of a (m,k)-firm Data Dropouts Policy on the Quality of Control. In *6th IEEE International Workshop on Factory Communication Systems*, Factory Communication Systems, 2006 IEEE International Workshop on, pages 353–359, Torino, Italy, June 2006. IEEE.

[50] I. Fialho and G. Balas. Road adaptive active suspension design using linear parameter-varying gain-scheduling. *IEEE Transactions on Control Systems Technology*, 10(1):43–54, 2002.

[51] B. Finkbeiner, F. Klein, R. Piskac, and M. Santolucito. Temporal stream logic: Synthesis beyond the bools. In *International Conference on Computer Aided Verification*, pages 609–629. Springer, 2019.

[52] B. Finkbeiner and S. Schewe. Uniform distributed synthesis. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*, pages 321–330. IEEE, 2005.

[53] J. Fu, H. G. Tanner, J. Heinz, and J. Chandlee. Adaptive symbolic control for finite-state transition systems with grammatical inference. *IEEE Transactions on Automatic Control*, 59(2):505–511, 2014.

[54] B. Galloway and G. P. Hancke. Introduction to industrial control networks. *IEEE Communications Surveys Tutorials*, 15(2):860–880, 2013.

[55] K. Gatsis, M. Pajic, A. Ribeiro, and G. J. Pappas. Opportunistic control over shared wireless channels. *IEEE Transactions on Automatic Control*, 60(12):3140–3155, 2015.

[56] X. Ge, Q.-L. Han, and Z. Wang. A dynamic event-triggered transmission scheme for distributed set-membership estimation over wireless sensor networks. *IEEE transactions on cybernetics*, 49(1):171–183, 2017.

[57] K. Ghasemi, S. Sadraddini, and C. Belta. Compositional synthesis via a convex parameterization of assume-guarantee contracts. In *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 16:1–16:10. ACM, 2020.

[58] K. Ghasemi, S. Sadraddini, and C. Belta. Decentralized signal temporal logic control for perturbed interconnected systems via assume-guarantee contract optimization. *arXiv e-prints*, pages arXiv–2207, 2022.

[59] P. J. Goulart and E. C. Kerrigan. Output feedback receding horizon control of constrained systems. *International Journal of Control*, 80(1):8–20, 2007.

[60] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski. Optimization over state feedback policies for robust control with constraints. *Automatica*, 42(4):523–533, 2006.

[61] S. Gulwani, O. Polozov, R. Singh, et al. Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119, 2017.

[62] L. Guo. Convergence and logarithm laws of self-tuning regulators. *Automatica*, 31(3):435–450, 1995.

[63] L. "Gurobi Optimization, 2019.

[64] T. Gurriet, P. Nilsson, A. Singletary, and A. D. Ames. Realizable set invariance conditions for cyber-physical systems. In *2019 American Control Conference (ACC)*, pages 3642–3649. IEEE, 2019.

[65] F. Harirchi, S. Z. Yong, and N. Ozay. Passive diagnosis of hidden-mode switched affine models with detection guarantees via model invalidation. In *Diagnosability, Security and Safety of Hybrid Dynamic and Cyber-Physical Systems*, pages 227–251. Springer, 2018.

[66] S. M. Hassaan, M. Khajenejad, S. Jensen, Q. Shen, and S. Z. Yong. Incremental affine abstraction of nonlinear systems, 2020.

[67] S. M. Hassaan, Q. Shen, and S. Z. Yong. Bounded-error estimator design with missing data patterns via state augmentation. In *American Control Conference*, 2019. Accepted.

[68] M. Hekmatnejad, S. Yaghoubi, A. Dokhanchi, H. B. Amor, A. Shrivastava, L. Karam, and G. Fainekos. Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In *Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design*, MEMOCODE '19, pages 6:1–6:11, New York, NY, USA, 2019. ACM.

[69] L. Heng, S. Hui-Fen, and W. Hao. Self-tuning kalman filter for the city sewage treatment system. In *Proceedings of the 2017 2nd International Conference on Communication and Information Systems*, ICCIS 2017, page 414–418, New York, NY, USA, 2017. Association for Computing Machinery.

[70] M. Herceg, M. Kvasnica, C. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *ECC*, pages 502–510, Zürich, Switzerland, July 17–19 2013. http://control.ee.ethz.ch/~mpt.

[71] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

[72] O. Holub, M. Zamani, and A. Abate. Efficient hvac controls: A symbolic approach. In *2016 European Control Conference (ECC)*, pages 1159–1164, 2016.

[73] P. Jain and M. J. Nigam. Real time control of ball and beam system with model reference adaptive control strategy using mit rule. In *2013 IEEE International Conference on Computational Intelligence and Computing Research*, pages 1–4, 2013.

[74] M. Jankovic. Control barrier functions for constrained control of linear systems with input delay. In *2018 Annual American Control Conference (ACC)*, pages 3316–3321. IEEE, 2018.

[75] L. Jaulin and E. Walter. Guaranteed nonlinear parameter estimation from bounded-error data via interval analysis. *Mathematics and computers in simulation*, 35(2):123–137, 1993.

[76] J.-B. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadski, and A. Platzer. A Formally Verified Hybrid System for the Next-Generation Airborne Collision Avoidance System (CMU-CS-14-138). 4 2005.

[77] N. Jia, Y.-Q. Song, and R.-Z. Lin. Analysis of networked control system with packet drops governed by (m,k)-firm constraint. *IFAC Proceedings Volumes*, 38(2):63 – 70, 2005. 6th IFAC International Conference on Fieldbus Systems and their Applications.

[78] Z. Jin, C. Ko, and R. M. Murray. Estimation for nonlinear dynamical systems over packet-dropping networks. In *2007 American Control Conference*, pages 5037–5042, July 2007.

[79] R. M. Jungers, A. Kundu, and W. P. M. H. Heemels. Observability and controllability analysis of linear systems subject to data losses. *IEEE Transactions on Automatic Control*, 63(10):3361–3376, 2018.

[80] J. Kersten, A. Rauh, and H. Aschemann. Interval methods for robust gain scheduling controllers. *Granular Computing*, 5(2):203–216, 2020.

[81] S. Keshmiri and S. Payandeh. A centralized framework to multi-robots formation control: Theory and application. In *Collaborative Agents-Research and Development*, pages 85–98. Springer, 2009.

[82] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *ITSC*, 2018.

[83] P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier. Learning-based approach for online lane change intention prediction. In *IEEE Int. Veh. Sym.*, pages 797–802, June 2013.

[84] F. Laine, C.-Y. Chiu, and C. Tomlin. Eyes-closed safety kernels: Safety for autonomous systems under loss of observability. *arXiv preprint arXiv:2005.07144*, 2020.

[85] J. Lee and G. Dullerud. Optimal disturbance attenuation for discrete-time switched and markovian jump linear systems. *SIAM Journal on Control and Optimization*, 45(4):1329–1358, 2006.

[86] D. J. Leith and W. E. Leithead. Survey of gain-scheduling analysis and design. *International Journal of Control*, 73(11):1001–1025, 2000.

[87] S. J. Leon. *Linear algebra with applications*. Pearson, 2014.

[88] L. Liebenwein, W. Schwarting, C.-I. Vasile, J. DeCastro, J. Alonso-Mora, S. Karaman, and D. Rus. Compositional and contract-based verification for autonomous driving on road networks. In *Robotics Research*, pages 163–181. Springer, 2020.

[89] Z. Liu, L. Yang, and N. Ozay. Scalable computation of controlled invariant sets for discrete-time linear systems with input delays. In *2020 American Control Conference, ACC 2020, Denver, CO, USA, July 1-3, 2020*, pages 4722–4728. IEEE, 2020.

[90] J. Löfberg. Yalmip : A toolbox for modeling and optimization in matlab. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[91] B. T. Lopez. *Adaptive robust model predictive control for nonlinear systems*. PhD thesis, Massachusetts Institute of Technology, 2019.

[92] B. T. Lopez, J.-J. E. Slotine, and J. P. How. Robust adaptive control barrier functions: An adaptive and data-driven approach to safety. *IEEE Control Systems Letters*, 5(3):1031–1036, 2021.

[93] A. Loría, E. Panteley, and M. Maghenem. Strict lyapunov functions for model reference adaptive control: Application to lagrangian systems. *IEEE Transactions on Automatic Control*, 64(7):3040–3045, 2019.

[94] M. Maghenem, A. J. Taylor, A. D. Ames, and R. G. Sanfelice. Adaptive safety using control barrier functions and hybrid adaptation. In *2021 American Control Conference (ACC)*, pages 2418–2423, 2021.

[95] A. Mahajan, N. C. Martins, M. C. Rotkowitz, and S. Yüksel. Information structures in optimal decentralized control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1291–1306. IEEE, 2012.

[96] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[97] O. Mangasarian. Set containment characterization. *Journal of Global Optimization*, 24(4):473–480, 2002.

[98] I. M. Mareels, B. D. Anderson, R. R. Bitmead, M. Bodson, and S. S. Sastry. Revisiting the mit rule for adaptive control. In K. ÅSTRÖM and B. WITTENMARK, editors, *Adaptive Systems in Control and Signal Processing 1986*, IFAC Workshop Series, pages 161–166. Pergamon, Oxford, 1987.

[99] K. Mehlhorn, B. R. Newell, P. M. Todd, M. D. Lee, K. Morgan, V. A. Braithwaite, D. Hausmann, K. Fiedler, and C. Gonzalez. Unpacking the exploration–exploitation tradeoff: A synthesis of human and animal literatures. *Decision*, 2(3):191, 2015.

[100] A. Mesbah. Stochastic model predictive control with active uncertainty learning: A survey on dual control. *Annual Reviews in Control*, 45:107–117, 2018.

[101] P.-J. Meyer, A. Girard, and E. Witrant. Compositional abstraction and safety synthesis using overlapping symbolic models. *IEEE Transactions on Automatic Control*, 63(6):1835–1841, 2017.

[102] O. Mickelin, N. Ozay, and R. M. Murray. Synthesis of correct-by-construction control protocols for hybrid systems using partial state information. In *American Control Conference*, pages 2305–2311, June 2014.

[103] M. Milanese and A. Vicino. Optimal estimation theory for dynamic systems with set membership uncertainty: an overview. *Automatica*, 27(6):997–1009, 1991.

[104] J. Mirkovic and P. Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, Apr. 2004.

[105] S. Moarref and H. Kress-Gazit. Decentralized control of robotic swarms from high-level temporal logic specifications. In *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 17–23, 2017.

[106] S. Mohan and R. Vasudevan. Convex computation of the reachable set for hybrid systems with parametric uncertainty. In *2016 American Control Conference (ACC)*, pages 5141–5147, 2016.

[107] N. Nayyar, D. Kalathil, and R. Jain. Optimal decentralized control with asymmetric one-step delayed information sharing. *IEEE Transactions on Control of Network Systems*, 5(1):653–663, 2018.

[108] S. A. Nazin and B. T. Polyak*. Interval parameter estimation under model uncertainty. *Mathematical and Computer Modelling of Dynamical Systems*, 11(2):225–237, 2005.

[109] T.-H. D. Nguyen, D. Hsu, W. S. Lee, T.-Y. Leong, L. P. Kaelbling, T. Lozano-Perez, and A. H. Grant. CAPIR: Collaborative action planning with intention recognition. In *AIIDE*, 2011.

[110] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *IEEE Transactions on Control Systems Technology*, 24(4):1294–1307, 2015.

[111] P. Nilsson, O. Hussien, A. Balkan, Y. Chen, A. D. Ames, J. W. Grizzle, N. Ozay, H. Peng, and P. Tabuada. Correct-by-construction adaptive cruise control: Two approaches. *TCST*, 24(4):1294–1307, 2016.

[112] P. Nilsson and N. Ozay. Synthesis of separable controlled invariant sets for modular local control design. In *2016 American Control Conference (ACC)*, pages 5656–5663. IEEE, 2016.

[113] P. Osburn. New developments in the design of adaptive control systems. *February 1961 Instituteof Aeronautical Sciences*, pages Paper–No, 1961.

[114] D. Panagou, M. Turpin, and V. Kumar. Decentralized goal assignment and safe trajectory generation in multirobot networks via multiple lyapunov functions. *IEEE Transactions on Automatic Control*, 65(8):3365–3380, 2020.

[115] S. Pankaj, J. S. Kumar, and R. Nema. Comparative analysis of mit rule and lyapunov rule in model reference adaptive control scheme. *Innovative Systems Design and Engineering*, 2(4):154–162, 2011.

[116] P. Parks. Liapunov redesign of model reference adaptive control systems. *IEEE Transactions on Automatic Control*, 11(3):362–367, 1966.

[117] M. Parmar, M. Halm, and M. Posa. Fundamental challenges in deep learning for stiff contact dynamics. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5181–5188, 2021.

[118] A. Pneuli and R. Rosner. Distributed reactive systems are hard to synthesize. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*, pages 746–757. IEEE, 1990.

[119] G. Pola, P. Pepe, and M. D. D. Benedetto. Decentralized supervisory control of networks of nonlinear control systems. *IEEE Transactions on Automatic Control*, 63(9):2803–2817, 2018.

[120] D. M. Raimondo, G. R. Marseglia, R. D. Braatz, and J. K. Scott. Closed-loop input design for guaranteed fault diagnosis using set-valued observers. *Automatica*, 74:107–117, 2016.

[121] A. Rantzer. Minimax adaptive control for a finite set of linear systems. In A. Jadbabaie, J. Lygeros, G. J. Pappas, P. A. Parrilo, B. Recht, C. J. Tomlin, and M. N. Zeilinger, editors, *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 893–904. PMLR, 07 – 08 June 2021.

[122] W. Ren and R. Jungers. Reachability-based control synthesis under signal temporal logic specifications. June 2022.

[123] S. Ricker. A question of access: decentralized control and communication strategies for security policies. In *2006 8th International Workshop on Discrete Event Systems*, pages 58–63, 2006.

[124] J. Rothe, J. Zevering, M. Strohmeier, and S. Montenegro. A modified model reference adaptive controller (m-mrac) using an updated mit-rule for the altitude of a uav. *Electronics*, 9(7), 2020.

[125] M. Rotkowitz and S. Lall. Decentralized control information structures preserved under feedback. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 1, pages 569–575 vol.1, 2002.

[126] M. Rungger and P. Tabuada. Computing robust controlled invariant sets of linear systems. *IEEE Trans. Autom. Control*, 62(7):3665–3670, July 2017.

[127] K. Rutledge and N. Ozay. Belief-prefix control for autonomously dodging switching disturbances. In *2020 European Control Conference (ECC)*, pages 966–972, 2020.

[128] K. Rutledge and N. Ozay. Correct-by-construction exploration and exploitation for unknown linear systems using bilinear optimization. In *25th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '22, New York, NY, USA, 2022. Association for Computing Machinery.

[129] K. Rutledge, S. Z. Yong, and N. Ozay. Finite horizon constrained control and bounded-error estimation in the presence of missing data. *Nonlinear Analysis: Hybrid Systems*, 36:100854, 2020.

[130] K. J. Rutledge, G. Chou, and N. Ozay. Compositional safety rules for inter-triggering hybrid automata. In *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, HSCC '21, New York, NY, USA, 2021. Association for Computing Machinery.

[131] K. J. Rutledge, S. Z. Yong, and N. Ozay. Optimization-based design of bounded-error estimators robust to missing data. *IFAC-PapersOnLine*, ADHS Full Papers, July 2018. Analysis and Design of Hybrid Systems ADHS.

[132] K. J. Rutledge, S. Z. Yong, and N. Ozay. Prefix-based bounded-error estimation with intermittent observations. In *American Control Conference*, 2019. Accepted.

[133] S. Sadraddini and C. Belta. Formal methods for adaptive control of dynamical systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1782–1787, 2017.

[134] S. Sadraddini and C. Belta. Distributed robust set-invariance for interconnected linear systems. In *2018 Annual American Control Conference (ACC)*, pages 1274–1279. IEEE, 2018.

[135] S. Sadraddini and R. Tedrake. Linear encodings for polytope containment problems, 03 2019.

[136] Y. E. Sahin, Z. Liu, K. Rutledge, D. Panagou, S. Z. Yong, and N. Ozay. Intention-aware supervisory control with driving safety applications. In *2019 IEEE Conference on Control Technology and Applications (CCTA)*, pages 1–8, 2019.

[137] S. Sahu and S. Behera. Improved pole-placement for adaptive pitch control. *International Journal of Swarm Intelligence*, 4(2):111–126, 2019.

[138] A. Saoud, A. Girard, and L. Fribourg. Contract-based design of symbolic controllers for safety in distributed multiperiodic sampled-data systems. *IEEE Transactions on Automatic Control*, 2020.

[139] J. K. Scott, R. Findeisen, R. D. Braatz, and D. M. Raimondo. Input design for guaranteed fault diagnosis using zonotopes. *Automatica*, 50(6):1580–1589, 2014.

[140] P. Seiler and R. Sengupta. Analysis of communication losses in vehicle control problems. In *American Control Conference*, volume 2, pages 1491–1496 vol.2, June 2001.

[141] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a formal model of safe and scalable self-driving cars, 2017.

[142] J. S. Shamma and K.-Y. Tu. Set-valued observers and optimal disturbance rejection. *IEEE Trans. on Automatic Control*, 44(2):253–264, 1999.

[143] Y. Shi and B. Yu. Robust mixed h2/h control of networked control systems with random time delays in both forward and backward communication links. *Automatica*, 47(4):754–760, 2011.

[144] K. Singh, Y. Ding, N. Ozay, and S. Z. Yong. Input design for nonlinear model discrimination via affine abstraction. *IFAC-PapersOnLine*, 51(16):175–180, 2018.

[145] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry. Kalman filtering with intermittent observations. *IEEE Trans. on Automatic Control*, 49(9):1453–1464, 2004.

[146] V. Sinyakov and A. Girard. Formal Controller Synthesis from Specifications Given by Discrete-Time Hybrid Automata. *Automatica*, 131, 2021.

[147] J. Skaf and S. P. Boyd. Design of affine controllers via convex optimization. *IEEE Trans. on Automatic Control*, 55(11):2476–2487, Nov 2010.

[148] J.-J. Slotine and L. Weiping. Adaptive manipulator control: A case study. *IEEE Transactions on Automatic Control*, 33(11):995–1003, 1988.

[149] S. W. Smith, P. Nilsson, and N. Ozay. Interdependence quantification for compositional control synthesis with an application in vehicle safety systems. In *CDC*, pages 5700–5707. IEEE, 2016.

[150] A. J. Taylor and A. D. Ames. Adaptive safety with control barrier functions. In *2020 American Control Conference (ACC)*, pages 1399–1405. IEEE, 2020.

[151] S. Thangavel, S. Lucia, R. Paulen, and S. Engell. Towards dual robust nonlinear model predictive control: A multi-stage approach. In *2015 American Control Conference (ACC)*, pages 428–433, 2015.

[152] S. Thangavel, S. Lucia, R. Paulen, and S. Engell. Robust nonlinear model predictive control with reduction of uncertainty via dual control. In *2017 21st International Conference on Process Control (PC)*, pages 48–53, 2017.

[153] M. Tillerson, L. Breger, and J. P. How. Distributed coordination and control of formation flying spacecraft. In *Proc. Amer. Control Conf*, volume 2, pages 1740–1745, 2003.

[154] M. Treiber, A. Hennecke, and D. Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E*, 62(2):1805–1824, Aug 2000.

[155] J. K. Tugnait. Approaches of fir system identification with noisy data using higher order statistics. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(7):1307–1317, 1990.

[156] K. Uçak and G. Ö. Günel. Model-free mimo self-tuning controller based on support vector regression for nonlinear systems. *Neural Computing and Applications*, pages 1–20, 2021.

[157] P. Varaiya. Trends in the theory of decision-making in large systems. In *Annals of Economic and Social Measurement, Volume 1, number 4*, pages 493–500. NBER, 1972.

[158] S. Vaskov, S. Kousik, H. Larson, F. Bu, J. R. Ward, S. Worrall, M. Johnson-Roberson, and R. Vasudevan. Towards provably not-at-fault control of autonomous robots in arbitrary dynamic environments. In *Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019*, 2019.

[159] R. Verma and D. Del Vecchio. Safety control of hidden mode hybrid systems. *IEEE Transactions on Automatic Control*, 57(1):62–77, 2011.

[160] R. Verma and D. Del Vecchio. Safety control of hidden mode hybrid systems. *IEEE Transactions on Automatic Control*, 57(1):62–77, Jan 2012.

[161] R. G. Walters and M. M. Bayoumi. Application of a self-tuning pole-placement regulator to an industrial manipulator. In *1982 21st IEEE Conference on Decision and Control*, pages 323–329, 1982.

[162] D. Wang, M. Ha, and J. Qiao. Self-learning optimal regulation for discrete-time nonlinear systems under event-driven formulation. *IEEE Transactions on Automatic Control*, 65(3):1272–1279, 2020.

[163] L. Wang, A. D. Ames, and M. Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33(3):661–674, June 2017.

[164] Y. Wang, J. Huang, D. Wu, Z.-H. Guan, and Y.-W. Wang. Set-membership filtering with incomplete observations. *Information Sciences*, 517:37–51, 2020.

[165] R. C. Wilson, A. Geana, J. M. White, E. A. Ludvig, and J. D. Cohen. Humans use directed and random exploration to solve the explore–exploit dilemma. *Journal of Experimental Psychology: General*, 143(6):2074, 2014.

[166] A. Wintenberg and N. Ozay. Implicit invariant sets for high-dimensional switched affine systems. In *2020 IEEE 59th Conference on Decision and Control (CDC)*, pages 3291–3297. IEEE, 2020.

[167] T. Wongpiromsarn, U. Topcu, and R. M. Murray. Synthesis of control protocols for autonomous systems. *Unmanned Systems*, 01(01):21–39, 2013.

[168] J. Yang, Z. Cai, Q. Lin, and Y. Wang. Self-tuning pid control design for quadrotor uav based on adaptive pole placement control. In *2013 Chinese Automation Congress*, pages 233–237, 2013.

[169] L. Yang and N. Ozay. Fault-tolerant output-feedback path planning with temporal logic constraints. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4032–4039. IEEE, 2018.

[170] L. Yang and N. Ozay. Efficient safety control synthesis with imperfect state information. In *Conference on Decision and Control (CDC) 2020*. IEEE, 2020.

[171] L. Yang and N. Ozay. Safety control synthesis for systems with missing measurements. *IFAC-PapersOnLine*, 54(5):97–102, 2021.

[172] L. Yang and N. Ozay. Scalable zonotopic under-approximation of backward reachable sets for uncertain linear systems. *IEEE Control Systems Letters*, 6:1555–1560, 2022.

[173] T. C. Yang. Networked control system: a brief survey. *IEE Proceedings-Control Theory and Applications*, 153(4):403–412, 2006.

[174] L. Ye, H. Zhu, and V. Gupta. On the sample complexity of decentralized linear quadratic regulator with partially nested information structure. *arXiv preprint arXiv:2110.07112*, 2021.

[175] W. Zhang, M. S. Branicky, and S. M. Phillips. Stability of networked control systems. *IEEE Control Systems*, 21(1):84–99, 2001.

[176] X.-M. Zhang and Q.-L. Han. Network-based h_∞ filtering for discrete-time systems. *IEEE Transactions on Signal Processing*, 60(2):956–961, 2012.

[177] X.-M. Zhang, Q.-L. Han, X. Ge, D. Ding, L. Ding, D. Yue, and C. Peng. Networked control systems: a survey of trends and techniques. *IEEE/CAA Journal of Automatica Sinica*, 7(1):1–17, 2020.

[178] K. Åström, U. Borisson, L. Ljung, and B. Wittenmark. Theory and applications of self-tuning regulators. *Automatica*, 13(5):457–476, 1977.

[179] K. J. Åström and B. Wittenmark. Adaptive control. 1995.