Advancing Optimization-Based Planning and Control of Mobile Robots under Temporal Logic Specifications

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Ali Tevfik Büyükkoçak

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Derya Aksaray, Advisor

December, 2024

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Derya Aksaray, whose guidance, vision, and unwavering support shaped me into the researcher I am today. Her continuous encouragement and timely feedback were instrumental to my growth. I also extend my sincere thanks to Prof. Yasin Yazıcıoğlu for his support, constructive input, and the invaluable perspective he brought to our joint projects. I am similarly grateful to Prof. Demoz Gebre-Egziabher for his expertise, thoughtful insights, and service as the committee chair and to Prof. Ryan Caverly, whose insightful feedback and excellent coursework deeply enriched my learning experience. In addition, I am thankful for the collaboration and profound discussions I shared with Prof. Peter Seiler and Prof. Vijay Gupta.

My time in graduate school would not have been the same without the friendship and support of my colleagues. Thank you, Sze Kwan Cheah, Yingjie Hu, Ryan Peterson, Liam Elke, Ahmet Şemi Asarkaya, Xiaoshan Lin, Azizollah Taheri, and many others whose names I may have inadvertently missed. Your camaraderie, diverse perspectives, and willingness to share experiences made the journey rewarding and memorable.

I am especially grateful to my mentors at Mitsubishi Electric Research Laboratories, Abraham Vinod and Stefano Di Cairano, for providing an environment that sparked creativity, including the unforgettable experience of flying autonomous drones on the top floor of a tall Boston building. To my co-interns, Andres Chavez Armijos and Andrea Ghezzi, thank you for your friendship, which made our time together both productive and enjoyable.

Over the course of my doctoral studies, I faced not only the intellectual challenges that accompany a PhD but also personal hardships and loss. During this period, I mourned the passing of my father-in-law, my friend Sri, and both of my grandmothers.

Being far away and unable to help weighed heavily on me, yet their memory continued to inspire me to persevere. I will forever cherish their influence on my life and honor them through my work and accomplishments.

To my family—my mother and father, who shaped my character and supported me throughout my entire upbringing, and my brother and sister, who ensured that my early years were filled with warmth, laughter, and comfort—I am profoundly grateful. Your unwavering belief in me has always been my foundation.

My heartfelt thanks also go to Sokka, the "goodest" boy of all time, whose presence lifted my spirits and eased my anxieties throughout this journey. You have been an unfailing source of comfort. I know we both eagerly await the arrival of our newest "gang member," Ediz, to continue building our joyful household.

I would also like to acknowledge the Galatasaray team, whose victories and spirit provided a much-needed respite, entertainment, and source of joy amid the rigors of academic life.

Finally—and certainly not least—I express my deepest love and appreciation to my high school sweetheart, best friend, wife, and soon-to-be mom, Zeynep. Your patience, encouragement, continuous support, and genuine companionship have carried me through every milestone. I will always remain grateful for the role you have played in my life and the happiness you continue to bring.

Dedication

To Zeynep, Ediz, and Sokka.

Abstract

The growing adoption of autonomous mobile robots underscores the need for reliable and resilient motion planning and control across diverse applications, ranging from warehouse automation to aerial surveying. Meeting complex, often time-sensitive objectives demands advanced motion planning techniques. In this regard, temporal logics, such as Signal Temporal Logic (STL), serve as rigorous and compact frameworks for encoding mission specifications, encompassing logical, spatial, and temporal constraints. However, designing control algorithms that fully satisfy these specifications—particularly in scenarios requiring multi-agent collaboration, resilience to unforeseen events, and real-time decision-making—presents significant challenges.

This dissertation tackles these challenges through three primary objectives. First, it investigates scalable frameworks for multi-agent systems to fulfill collective temporal logic specifications, including preemptable tasks agents can complete asynchronously. It combines sampling-based trajectory generation (e.g., RRT*) with mixed-integer programming (MIP), to address missions involving heterogeneous agents. Second, it explores resilient motion planning strategies, introducing a quantitative metric to minimize specification violations when constraints are breached. This metric captures cumulative task relaxation, allowing structural adjustments in STL specifications, such as modifying task time intervals or removing tasks when necessary. Lastly, the dissertation addresses real-time STL-based planning for missions requiring fast decision-making, such as completing tasks defined on noncooperative targets, by constructing control barrier functions (CBFs) based on the robots' actuation limits and an optimized sequence of STL tasks. The proposed frameworks and algorithms are validated through theoretical analyses, simulations, and experiments, demonstrating their scalability and effectiveness in complex robotic applications.

Through these contributions, this work advances temporal logic-based control, enhancing STL's applicability to resilient and real-time motion planning. It provides essential insights and tools for mobile robots capable of autonomous operation in evolving environments with intricate mission requirements.

Contents

Acknowledgements **i**

Dedication **iii**

Abstract **iv**

Contents **v**

List of Tables **ix**

List of Figures **x**

1 Introduction **1**

1.1 Motivation 3

1.2 Primary Research Questions 6

1.3 Dissertation Overview and Contributions 8

2 Background **13**

2.1 Notation 13

2.2 Signal Temporal Logic 14

2.3 Graph Theory 19

2.4 Time-Varying (Zeroing) Control Barrier Functions 19

3 Enhancing the Expressiveness of Signal Temporal Logic **21**

3.1 Introduction and Literature Review 22

3.2 Integral and Derivative Predicates 25

3.2.1 Comparison with the Existing Metrics 28
3.2.2 Optimal STL Control Synthesis Problem 29
3.3 Mixed-Integer Encoding of the STL Control Synthesis Problem 30
3.4 Case Study: STL Control Synthesis for Ground Robots 32
3.4.1 Simulation Results 35
3.5 Discussion 37

4 Scalable Operation of Heterogeneous Multi-Agent Systems under STL Specifications 38
4.1 Introduction and Literature Review 39
4.2 Multi-Agent Coordination Problem under STL Specifications with Integral Predicates 41
4.2.1 Formulation of the Coordination Problem 42
4.2.2 Environment Model 43
4.2.3 Swarm Flow Dynamics 43
4.2.4 Service Propositions 45
4.2.5 MILP Encoding of the Coordination Problem 49
4.2.6 Case Study: A Cooperative Multi-Agent System 50
4.3 Energy-Aware Planning of Heterogeneous Multi-Agent Systems for Serving Cooperative Tasks 54
4.3.1 Coordination Problem with Energy Constraints 55
4.3.2 Team-Level Energy-Aware Planning 57
4.3.3 Agent-Level Implementation under Charging Constraints 60
4.3.4 Case Study: A Cooperative Energy-Aware Multi-Agent System . 63
4.4 Discussion 66

5 Resilient Motion Planning under STL Specifications 68
5.1 Introduction and Literature Review 69
5.2 A Family of STL Specifications for Temporal Relaxation 72
5.3 Temporal Relaxation Metric 74
5.4 Temporally Relaxed STL Control Synthesis 79
5.5 Temporal Relaxation vs. Time Robustness 83
5.5.1 Benchmark Analysis 84

5.6 Resilience via Reactive Motion Planning 87
5.6.1 Planning Problem in Dynamic Environments 87
5.6.2 Reactive Motion Planning with Temporal Relaxation 90
5.7 Discussion 98

6 Real-Time Resilient Control Synthesis under STL Specifications 100
6.1 Introduction and Literature Review 101
6.2 A Family of STL Specifications for Real-Time Control 103
6.3 Problem Formulation: STL Control Synthesis with Time-Varying Specifications 104
6.4 Solution Approach: STL Control Synthesis via Control Barrier Functions 106
6.4.1 Scheduling STL Tasks 109
6.4.2 Offline Task Sequence Generation 117
6.4.3 Control Barrier Functions with Sequential Tasks 122
6.4.4 STL Control Synthesis under CBFs with Sequential Tasks 124
6.4.5 Complexity Analysis 128
6.4.6 Extension of the Approach 129
6.4.7 Conservativeness of the Approach 130
6.5 Case Studies: Dynamic Targets 130
6.5.1 Scenarios 1 and 2: Main Scenario with Different Target Trajectories 132
6.5.2 Scenario 3: All Periodic Tasks 135
6.5.3 Scenarios 4 and 5: Pursuit-Evasion under Different Target Velocities 135
6.5.4 Scalability Analysis 137
6.6 Application to Stationary Targets 139
6.6.1 Benchmark Analysis 140
6.6.2 Case Study 142
6.7 Discussion 144

7 Conclusion and Future Directions 146
7.1 Conclusion 146
7.2 Future Work 147

References 151

Appendix A. Abbreviations and Symbols **165**
A.1 List of Abbreviations 165
A.2 List of Symbols 166

List of Tables

3.1 Results of the simulation scenarios in Fig. 3.2. 36
4.1 Properties of the three agent types. 51
4.2 Computation times for the different number of agents with three agent types designated as $n^{agent}_{diamond} + n^{agent}_{triangle} + n^{agent}_{circle}$. 53
4.3 Properties of the three agent teams with energy constraints. 64
5.1 Simulation results and optimization parameters for temporal relaxation minimization and standard time robustness maximization. 85
5.2 Mean values with 95% confidence intervals of the total of 300 randomized trials. 97
6.1 How to implement the CBF constraints under STL. 127
6.2 Mission parameters for different scenarios. 132
6.3 Initial and realized task sequences. 133
6.4 Simulation execution times. 135
6.5 Comparison of the results with popular control synthesis approaches. . . 144
A.1 Abbreviations and Acronyms. 165

List of Figures

1.1 Organization of the dissertation. 8
2.1 An example of signal yielding different space robustness for different STL specifications. 17
3.1 Example of signal quantities defined via integration over particular signal segments. 23
3.2 Trajectories for four different cases of natural habitat monitoring mission. 34
3.3 Velocity results for four different cases of natural habitat monitoring mission. 36
3.4 Acceleration results for four different cases of natural habitat monitoring mission. 37
4.1 Pairwise obstacle-free RRT* paths between regions of interest. 50
4.2 Team trajectories satisfying team-level tasks collaboratively. 52
4.3 Snapshot from the experiment with six drones (two in each team). . . . 53
4.4 An illustration of heterogeneous agent paths with common objectives. . 54
4.5 Proposed planning and execution scheme for energy-aware heterogeneous agents. 56
4.6 Energy-aware team plans at selected time instants. 65
4.7 How threshold γ on the probability of available charging affects the actual and expected temporal relaxation. 66
4.8 Snapshot from the Robotarium simulation with six ground robots. . . . 67
5.1 Two signals that violate the given STL specifications by different amounts. 73
5.2 Trajectories of the robot under the proposed minimal temporal relaxation control approach together with right and left time robustness maximizing approaches. 86

5.3 Time history of state x. 86
5.4 Two sample environment modes with fixed goal regions (gray) and dynamic obstacle settings (red). 88
5.5 Potential recovery strategies in the face of a major event defined in Def. 5.6.2. 90
5.6 Proposed planning and execution scheme. 91
5.7 Robot progression at various times in a dynamic environment. 97
5.8 Eight potential obstacles that emerge under the specification $\Phi_{\text{goal}'}$. . . 98
6.1 Sample conflicting tasks. 108
6.2 Outline of the proposed framework. 109
6.3 Derivation of new sequences. 114
6.4 A sample sequence with laxity values. 122
6.5 Physical environment with Crazyflies. 131
6.6 Time history of the ego and target drones yielding the satisfaction of the specification $\Phi_{case\,1}$. 134
6.7 Satisfactory robot trajectory for the specification $\Phi_{case\,2}$. 136
6.8 Satisfactory robot trajectories for the specification $\Phi_{case\,3}$. 137
6.9 Robot trajectories satisfying the specification of $\Phi_{case\,4'}$. 138
6.10 How execution time scales with the length of the STL specification. . . . 139
6.11 Comparison of two CBF approaches 1. 141
6.12 Comparison of two CBF approaches 2. 141
6.13 Comparison of three CBF approaches. 142
6.14 Trajectory of the system under STL CBF. 143
6.15 History of CBF values. 144
7.1 A block diagram of the planner and tracker. 148
7.2 A block diagram illustrating the layers of estimation-aware planning. . . 150

Chapter 1

Introduction

The availability of inexpensive mobile robots with advanced sensing, communication, and computation capabilities has significantly expanded the application areas of autonomous robotic systems. Consequently, it is now possible to deploy such systems to dull, dirty, and dangerous duties instead of human beings in areas such as warehouse servicing, automated manufacturing, and transportation, to name a few. Moreover, aerial robotics, including unmanned aerial vehicles (UAVs) or drones, is also a rapidly growing field with many applications, from search and rescue to mapping and delivery.

For an autonomous robot to operate independently, designers must ensure some guarantees on the motion planning and control of the vehicle. First, it must operate safely both for itself and for the environment. Moreover, it must meet predefined performance criteria regarding its assigned mission. While safety and performance requirements may conflict with each other, a common approach is to ensure safety with a compromise on performance when necessary. These considerations can be formulated as an optimization problem, i.e., maximizing the performance, subject to safety constraints. For example, a drone may need to deliver a package, consuming as little fuel as possible. However, it must also avoid collisions; hence, it may be required to stay away from tall buildings. Therefore, avoiding cluttered areas might be a better trajectory with the cost of additional fuel consumption.

While algebraic functions can be effectively used to express tasks such as reach-and-avoid missions, it becomes harder to satisfy mission specifications as they get more

complicated with explicit space and time constraints. For instance, a drone may be assigned to periodically visit regions of interest in a specific order (persistence and precedence), taking high-resolution images if an intruder is detected (response) while not running out of energy and avoiding no-fly zones (safety). Moreover, the drone may be asked to pick up the package from any one of multiple warehouse locations redundantly and deliver it before a specified deadline. It may also be assigned secondary tasks such as frequently communicating its observations (e.g., monitoring traffic on its route). Some secondary tasks might be crucial. For example, the drone may have to keep visiting one of the charging stations every two hours to ensure continuity of operation.

Such intricate robotic tasks, encompassing logical, spatial, and temporal requirements with complex and time-dependent properties and behaviors, can be expressed by *temporal logic* (TL) specifications [1]. Temporal logic is a helpful mathematical framework for robotic systems that allows for the precise specification of the desired robot trajectories and the fulfillment of complex high-level specifications. This capability is often essential in motion planning and control of robotic systems deployed in a wide range of applications such as surveillance/reconnaissance, warehousing, autonomous driving, and precision agriculture. For example, Linear Temporal Logic (LTL) [2] has been extensively used in planning and control of autonomous robots (e.g., [3–11]). Moreover, several other temporal-logic-based studies investigate the generation of motion plans ensuring satisfaction of various temporal logic specifications (e.g., [12–22]).

Temporal logics such as Metric Temporal Logic (MTL) [23], Metric Interval Temporal Logic (MTL) [24], and Signal Temporal Logic (STL) [25], [26] are expressive specification languages that can define signal properties with explicit space and time parameters. In particular, STL can define properties of dense-time real-valued signals. Different than the existing temporal logics, STL also allows for *predicates* in the form of inequalities and is endowed with a robustness metric that can quantify how well a signal satisfies a specification [26]. The STL robustness metric not only gives a yes/no answer but also provides a real value that quantifies the degree of satisfaction. For example, a large positive robustness value implies a robust satisfaction of the specification whereas a large negative value means an utter failure. Such a metric also enables formulating an optimization problem that solves for a trajectory satisfying the temporal logic specifications (e.g., [4, 14, 17, 27]).

Control synthesis under the STL constraints has been broadly studied, and various methods have been proposed such as mixed-integer encoding of specifications (e.g., [14, 16, 28, 29]), nonlinear program solutions using smooth approximations of STL robustness metrics (e.g., [27, 30–33]), or hierarchically employing both approaches (e.g., [34]). Alternatively, more time-efficient methods such as control barrier functions are also proposed to achieve such specifications [22, 35]. Although called "STL control synthesis," the aforementioned methods tackle the motion planning and control problems simultaneously, aiming to find state trajectories that satisfy an STL specification and the control inputs that realize these trajectories.

This dissertation addresses the satisfaction of mission requirements, expressed as Signal Temporal Logic specifications, by developing motion planning and control methods for autonomous mobile robots, while ensuring safety and performance guarantees. In particular, this work extends the expressiveness and capabilities of STL by addressing its application to motion planning and control in complex scenarios, ranging from cooperative multi-agent settings and real-time applicability of STL specifications to resilient operation in dynamic environments with changing conditions.

1.1 Motivation

Among the many challenges faced by robotics applications, three major topics that this dissertation research is related to and have received significant attention in recent years are *multi-agent coordination*, *resilient motion planning*, and *real-time decision-making*. First, in the scope of this dissertation, multi-agent coordination refers to the coordination and control of multiple robots collaborating to achieve a common goal. Second, resilient motion planning refers to the ability of the system to continue functioning in the face of unexpected or adverse conditions, such as equipment failures or unforeseen environmental changes. Finally, real-time decision-making is the ability of the system to respond quickly and accurately to changes in its environment, requiring computationally efficient online motion planning and control methods. These three topics are essential to ensuring robotic systems' safety, reliability, and high performance in various real-world scenarios.

Multi-agent systems are useful in many ways, such as task allocation yielding shorter mission completion times, coordinated task execution, and resilience to agent loss. Several multi-agent applications utilize temporal logic to incorporate safety and collaboration constraints. However, the existing research is somehow limited when it comes to heterogeneous robots with different dynamic properties and capabilities. For example, one ground robot, which can measure temperature, may be needed in a particular area, while at least three drones equipped with cameras might be required to complete a mapping task on time. An approach addressing this issue could allow for the deployment of certain numbers of heterogeneous agents from different teams with specific capabilities via temporal logic specifications. Moreover, such a solution would enable robots to give preemptable services that different agents can do at separate times. In other words, after one agent starts it, another agent can continue the task without an additional cost. For instance, one may require a specific number of temperature readings from a region. This can be met cumulatively by the agents equipped with a thermometer. However, while drones can do that faster, ground robots consume less energy. An optimal solution, both in time and energy, may include using a combination of drones and ground robots. Finding such a solution is an NP-hard problem as there are arbitrarily many ways of achieving such tasks with numerous combinations of particular agents, especially when there are too many robots. Therefore, the solution should scale well with the increasing number of agents. We address these concerns in this dissertation

Robotic systems, whether multi- or single-agent, are often deployed in challenging environments, such as industrial settings, disaster response, or space exploration. In such real-world applications, autonomous robots operate under various disturbances (e.g., internal, external, human-triggered), and there might be cases where the desired mission specification becomes unsatisfiable. In such cases, resilient motion planning and control are required to search for trajectories, resulting in minimal violations of the original specification. In general, resilience in robotic systems refers to the ability of the system to continue functioning in the face of unexpected or adverse conditions.

A resilient system can detect and recover from failures or disturbances and continue to perform its intended tasks, thus providing continuity of operation and minimizing the impact of failures. One common way of achieving resilience in motion planning with temporal logic specifications is by relaxing the spatial requirements of the specification,

such as dropping a package half a meter away from the desired location. On the other hand, relaxing deadlines by strictly enforcing task locations has not been investigated much. However, applications such as robot arms picking up objects precisely from fixed locations would benefit from it. A reactive method can be proposed for satisfying temporal logic specifications by robot trajectories with minimal relaxation of deadlines or other time constraints in the face of inevitable failures due to actuator limits or dynamic obstacles. User preferences can also be considered when prioritizing tasks over others when deciding what to compromise in the face of such events.

Besides the high number of robots and dynamic environments, many other aspects also make the generation of satisfactory robot trajectories harder, e.g., more obstacles to avoid and targets to visit or longer mission durations. The harder the problem gets, the longer it takes to obtain the solution. This may dictate the generation of trajectories beforehand in an offline fashion without being able to react to unforeseen events during the actual mission. However, robotic systems are inherently real-time systems. As they interact with the physical world, they must react to their environment, preferably fast.

In many cases, robotic systems are used in environments where a delay in response could have serious consequences, such as in industrial automation and search-and-rescue operations. In such scenarios, real-time and reactive performance is crucial for the safety and effectiveness of the system. Real-time decision-making is particularly important for autonomous systems because it ensures quick and accurate responses to environmental changes without compromising the mission's overall success.

For generating satisfactory robot trajectories in real time, existing work is somehow restricted to limited families of mission specifications, such as not allowing for repetitive tasks or the inability to satisfy tasks with overlapping time intervals. One approach to enhance the spectrum of specifications achieved in real time could be the generation of a feasible sequence of reach-and-avoid tasks and executing that sequence in real time with potential modifications or corrections. This involves the sequential solution of simple optimization problems over time after the feasible scheduling of tasks, which can result in achieving a broader family of temporal logic specifications in real time and in a computationally efficient manner. With this capability, it would be possible to define tasks over dynamic and noncooperative targets whose future trajectories are not known to the controlled robot.

1.2 Primary Research Questions

Temporal Logic, particularly Signal Temporal Logic (STL), provides a versatile framework for expressing a wide array of specifications over real-valued signals. This includes the ability to define thresholds on signal values within specific time intervals. However, there might be cases when cumulative or transitional values over a signal hold more significance than the satisfaction of instantaneous thresholds. For instance, consider the specification "robot A needs to visit region X within a minute." This can be readily captured by an existing temporal logic specification, which mandates the robot to enter the region at any given instant within the allowed time frame. Now suppose it is desired to satisfy "the region X has to be visited a total of 10 times within an hour." This specification introduces a "preemptable task" that can be interrupted and resumed later—a nontrivial notion to express and synthesize effectively. Furthermore, in a multi-agent setting, different agents can contribute to the same preemptable task. Although algebraic functions can somehow define such cumulative tasks, they become much more complicated when they are combined with temporal logic constraints that reflect additional temporal, logical, and spatial requirements. That said, the existing temporal logic syntax might not suffice to express such tasks either. Therefore, a viable control synthesis approach for managing these redundant mission tasks and the ability to define them in combination with temporal logic specifications is not trivial. Consequently, the first research question driving this dissertation emerges:

Research Question 1. *How can we express preemptable tasks by temporal logic specifications with explicit deadlines? How can we design a scalable planning and control framework for a heterogeneous multi-agent system that can ensure the satisfaction of temporal logic specifications, including such preemptable tasks?*

When designing robot trajectories governed by temporal logic specifications, achieving complete satisfaction in all aspects, including logical, temporal, and spatial requirements, may not always be feasible due to various factors, such as disturbances, dynamic environments (e.g., moving targets or obstacles), or impractical specifications. In such cases, an alternative objective could be to attain the specification with minimal violations. Deciding what to compromise reactively in the face of inevitable violations is a critical consideration. It is essential to explicitly state which aspects of the resulting

specification are realized, as opposed to the original one. While relaxing spatial requirements is a common approach in the face of failure, the nature of the mission may lead users to prefer relaxing deadlines, allowing temporal deviations from the specification. Moreover, for a more favorable cumulative outcome, specific tasks within the mission can be entirely compromised, on top of relaxing certain temporal requirements. These alternatives lead us to formulate the second research question guiding this study:

Research Question 2. *How can we quantify the difference between the original and relaxed specifications? How can we design a resilient control synthesis method that minimally relaxes the temporal requirements of the original STL specification in the face of infeasibilities?*

Optimization-based control synthesis methods are powerful in the sense of providing near-optimal solutions for trajectory planning. However, temporal logic specifications can be intricate, encompassing temporal, spatial, and logical conditions. This complexity engenders a large number of variables and parameters, particularly for extended missions and complicated specifications. Consequently, real-time applicability becomes compromised. Despite the complexity of the specifications, the resulting trajectories that fulfill them can be depicted as several simple reach-and-avoid segments. This insight has prompted significant research into iteratively applying basic reach-and-avoid planning techniques to meet complex temporal logic specifications. While this approach has shown promise, it falls short in comprehending the full spectrum of logical and temporal connections that temporal logics can encapsulate, especially with dynamic noncooperative targets. As a result, the last primary research question takes shape:

Research Question 3. *How can we design real-time control synthesis methods that ensure the satisfaction of time-varying STL specifications?*

Answering these research questions will advance the theoretical understanding and practical application of temporal logic in control synthesis and planning for autonomous mobile robots.

1.3 Dissertation Overview and Contributions

After the introduction, the dissertation continues with background information on Signal Temporal Logic and some additional preliminaries (Chapter 2). The following four chapters include the primary contributions of this dissertation (Chapters 3,4,5, and 6), answering the research questions in Sec. 1.2. Each of these chapters also includes a comprehensive literature review. Figure 1.1 shows the organization of the dissertation.

The research effort highlighted in this dissertation has yielded peer-reviewed publications or submitted for publication by the time of completion. In what follows, a brief description of each chapter and its contributions are discussed.

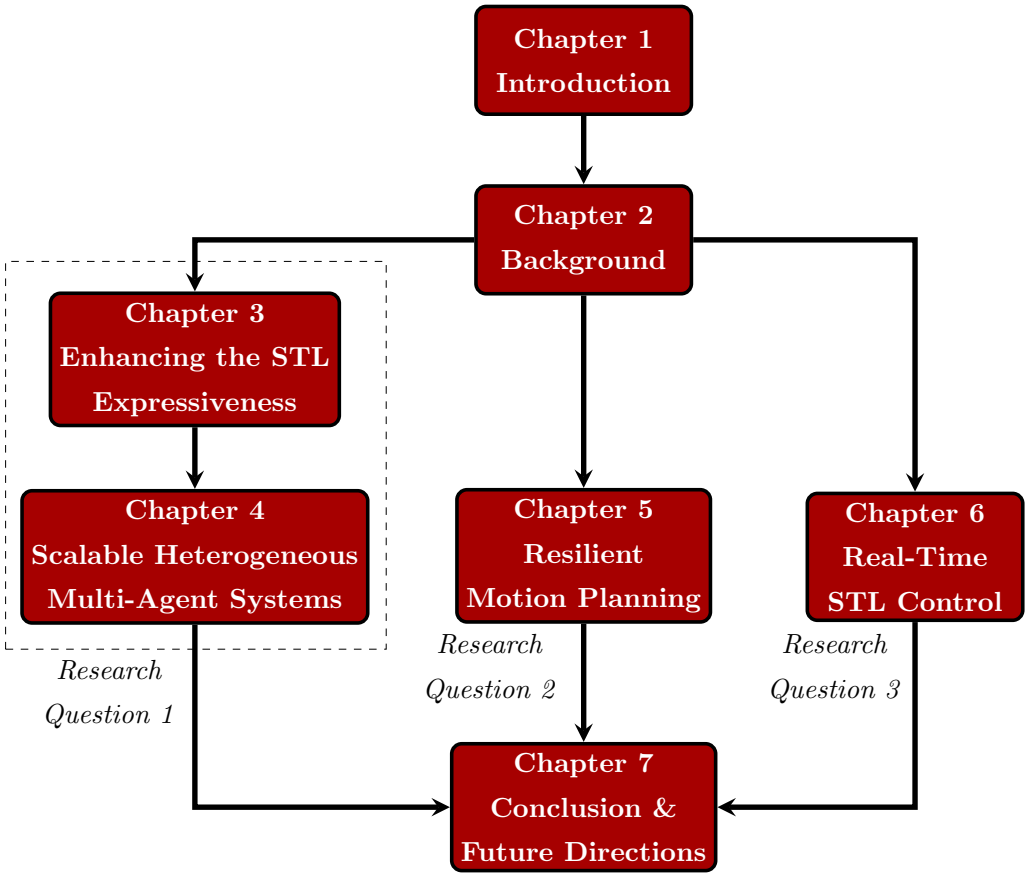


Figure 1.1: Organization of the dissertation.

Chapter 2: Background on Signal Temporal Logic (STL) is provided along with quantification metrics it is endowed with and some other properties. Moreover, some preliminaries on Graph Theory and Time-Varying Control Barrier Functions (TV-CBFs) are also discussed, as they are significantly used in Chapters 4, 5, and 6.

Chapter 3: This chapter extends the expressiveness of STL by introducing predicates that can define rich properties related to the integral and derivative of a signal. The new predicates involve the signal values at different time steps to define cumulative and relative specifications within desired intervals. The proposed predicates allow the local and global signal/trajectory characteristics to be controlled extensively without losing the existing STL capabilities. We also use these predicates for multi-agent settings in Chapter 4 to address Research Question 1.

Neither the conventional STL nor other temporal logics can specify queries about a cumulative success over the signal values. This is due to the lack of operators or predicates with references to explicit time bounds encompassing cumulative progress. We propose an enriched syntax to request a specific amount of progress in preemptable and cumulative properties via temporal logic specifications, in addition to utilizing the conventional STL. We also present a case study on the control of an autonomous robot in two-dimensional continuous space showing how the new predicates can be used in the design of the mission specifications more richly and expressively compared to the conventional STL.

The chapter also offers mixed-integer encoding of the standard STL semantics and that of new predicates. This mixed-integer encoding is also (partially) used for the STL control synthesis in the form of a mixed-integer linear program (MILP) in Chapters 4 and 5 to find trajectories satisfying STL specifications.

The covered material is based on the following publication:

- A.T. Büyükkoçak, D. Aksaray, and Y. Yazıcıoğlu, "Control Synthesis using STL Specifications with Integral and Derivative Predicates," American Control Conference (ACC) 2021, New Orleans, LA, 26-28 May 2021.

Chapter 4: We address the problem of coordinating the trajectories of heterogeneous multi-agent systems under spatio-temporal specifications. We use the new predicates defined in Chapter 3 to express the number and type of agents that should be present at specific locations within the desired time windows. Moreover, integral predicates are used to specify cumulative progress that can be achieved asynchronously by multiple agents, addressing Research Question 1.

To generate optimal trajectories, a mixed-integer linear program (MILP), the objective of which is to minimize the agent movement, is formulated. We also introduce energy constraints and a hierarchical solution to the energy-aware coordination problem. The agents are steered under stochastic energy consumption models, ensuring recharge availability. Such stochastic energy dynamics cause deviations from the nominal plan and delay the completion of tasks. Accordingly, a preliminary metric is defined to evaluate the expected delay (temporal relaxation), which will be investigated rigorously later in Chapter 5.

The approach we propose in this chapter allows for the definition of common objectives at the team level via integral predicates and the deployment of heterogeneous agents with specific capabilities and unique dynamics via STL specifications, by combining sampling-based trajectory generation (e.g., RRT*) with MILP-based optimal planning. We show how the proposed framework can be used in the planning of the multi-agent trajectories more richly and expressively compared to the conventional STL via two case studies with multiple agents that are heterogeneous in both dynamics and capabilities (e.g., sensors and payload).

The covered material is based on the following publications:

- A.T. Büyükkoçak, D. Aksaray, and Y. Yazıcıoğlu, "Planning of Heterogeneous Multi-Agent Systems under Signal Temporal Logic Specifications with Integral Predicates," IEEE Robotics and Automation Letters (RA-L), vol 6, no 2, 1375-1382, 2021.

- A.T. Büyükkoçak, D. Aksaray, and Y. Yazıcıoğlu, "Energy-aware Planning of Heterogeneous Multi-Agent Systems for Serving Cooperative Tasks with Temporal Logic Specifications," International Conference on Intelligent Robots and Systems (IROS) 2023, Detroit, MI, 1-5 October 2023.

Chapter 5: In this chapter, Research Question 2 is addressed. We introduce a metric that can quantify the temporal relaxation of STL specifications and facilitate resilient control synthesis in the face of infeasibilities. The proposed metric quantifies a cumulative notion of relaxation among the tasks, and minimizing it yields structural changes in the original STL specification by i) modifying time intervals and ii) removing tasks entirely if needed. We also propose a reactive planning algorithm that generates robot trajectories resilient to temporal violations. By monitoring the robot's behavior in real time, the system makes on-the-fly decisions when confronted with unforeseen events—such as potential collisions or environmental changes necessitating *task removal* or *partial satisfaction*—by initiating local corrections using control barrier functions or replanning trajectories to minimize temporal violations.

This strategy is designed to prevent arbitrarily long delays in mission completion and facilitate the satisfaction of the mission with minimal temporal relaxation. To ensure this while maintaining spatial robustness, we introduce an efficient hierarchical two-fold MILP-based optimization scheme that simultaneously minimizes temporal relaxation and maximizes spatial robustness. Further computational efficiency is enabled by a reduced number of optimization parameters, focusing on local and future variables, and using fewer mixed-integer constraints.

Unlike existing literature, which focus on offline solutions and/or guaranteed satisfaction without room for violation, our approach provides online, reactive planning that continuously monitors and adjusts the robot's trajectory as needed to minimize violations as per user preferences. We present theoretical results supporting our framework, demonstrate its effectiveness through simulations with a mobile robot, and compare it with existing metrics.

The covered material is based on the following publications:

- A.T. Büyükkoçak and D. Aksaray, "Temporal Relaxation of Signal Temporal Logic Specifications for Resilient Control Synthesis," IEEE Conference on Decision and Control (CDC) 2022, Cancun, Mexico, 6-9 December 2022.

- A.T. Büyükkoçak and D. Aksaray, "Resilient Online Planning for Mobile Robots with Minimal Relaxation of Signal Temporal Logic Specifications," under review.

Chapter 6: This chapter addresses the problem of planning trajectories of robotic systems in real time which is highlighted in Research Question 3, satisfying STL specifications that contain dynamic targets (i.e., time-varying specifications). The main challenge is making decisions in a way that the STL specification over the dynamic non-cooperative targets is satisfied while the robot does not know how targets are moving. We propose control barrier functions (CBFs) that take into account the actuation limits of the robots and a feasible sequence of STL tasks. We show the feasible sequence generation process that even includes the decomposition of periodic tasks and alternative/redundant scenarios.

Over the obtained sequence, we construct sequential CBFs to account for the worst-case bounds for target motions and achieve logical, temporal, and spatial requirements associated with these targets. We guarantee the satisfaction of STL specifications defined over the compact target sets if there is a feasible sequence over the convex under-approximations of these sets. Under these assumptions, the target motion—whether random, predefined, or adversarial—has no major effect on the success. Moreover, such changes in the mission setting, along with the high number of targets, do not undermine the real-time applicability.

We show theoretical results on the correctness of the proposed method. In particular, the results highlight the i) relationship between the STL deadlines and feasibility of task sequence, ii) effect of periodic task decomposition on sequence properties, iii) relationship between (sub)sequence feasibility and CBF constraints, and iv) overall satisfaction of STL specifications with the proposed framework. We also illustrate the benefits of our method and analyze its performance via simulations and drone experiments.

The covered material is based on the following publications:

- A.T. Büyükkoçak, D. Aksaray, and Y. Yazıcıoğlu, "Control Barrier Functions with Actuation Constraints under Signal Temporal Logic Specifications," European Control Conference (ECC) 2022, London, UK, 12-15 July 2022.
- A.T. Büyükkoçak, D. Aksaray, and Y. Yazıcıoğlu, "Sequential Control Barrier Functions for Mobile Robots with Dynamic Temporal Logic Specifications," Robotics and Autonomous Systems, 176 (2024): 104681.

Chapter 7: The last chapter gives the conclusion of this dissertation and makes recommendations for future research, discussing some preliminary results.

Chapter 2

Background

This chapter provides preliminary concepts on Signal Temporal Logic (STL) in Sec. 2.2, the basics of graph theory in Sec. 2.3, and the definition of control barrier functions, particularly the time-varying ones in Sec. 2.4. The chapter starts with the notation used throughout the dissertation.

2.1 Notation

In the following chapters, $\mathbb{R}$ and $\mathbb{Z}$ represent the sets of real and integer numbers, respectively. More specifically, $\mathbb{R}_{\geq 0}$ ($\mathbb{Z}_{\geq 0}$) is the set of nonnegative real (integer) numbers, and $\mathbb{R}^+$ ($\mathbb{Z}^+$) is the set of positive real (integer) numbers. While $\mathbb{R}^n$ is the set of all n-dimensional vectors, $\mathbb{R}^{m\times n}$ ($\mathbb{Z}^{m\times n}$) denotes the set of all real (integer) valued $m \times n$ matrices. The expression of $\mathbf{1}_{m\times 1}$ is an m element column-vector of ones. For any $X \in \mathbb{R}^{m\times n}$, $\|X\|_1$ denotes the element-wise absolute summation:

$$\|X\|_1 := \sum_{i=1}^{m}\sum_{j=1}^{n} |X_{i,j}|,$$

where $|X_{i,j}|$ is the absolute value of $X_{i,j}$. Operator $|\cdot|$ is also used to designate the cardinality of a set. Moreover, the Euclidean norm is interchangeably represented by the operators of $\|\cdot\|$ and $\|\cdot\|_2$. A function's right and left time derivatives are denoted as $d(\cdot)/dt^+$ and $d(\cdot)/dt^-$, respectively.

For any $Y, Y' \in \mathbb{R}^n$, $Y \leq Y'$ (or $Y < Y'$) denotes the element-wise inequalities, i.e., $Y_i \leq Y_i'$ (or $Y_i < Y_i'$) for all $i = 1, 2, \ldots, n$.

Let $\mathrm{v}_{a,b}$ be a vector defined via indices $a \in A$ and $b \in B$ with $A, B \subset \mathbb{Z}^+$ being finite index sets, the set of all such vectors is designated as $\bigcup_a \bigcup_b \mathrm{v}_{a,b} = \{\mathrm{v}_{a,b} \mid a \in A,\ b \in B\}$. For $x \in \mathbb{R}$, the operator $(\cdot)_+ : \mathbb{R} \to \mathbb{R}_{\geq 0}$ projects the expression as $(x)_+ = \begin{cases} x, & \text{if } x \geq 0, \\ 0, & \text{otherwise.} \end{cases}$

Floor and ceiling functions are designated by $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$, respectively. Minkowski sum of two sets $A, B \subset \mathbb{Z}$ is denoted by $\oplus$ where $A \oplus B = \{a + b \mid a \in A, b \in B\}$. With a slight abuse of notation, we also allow $a \oplus B = \{a + b \mid b \in B\}$.

2.2 Signal Temporal Logic

Signal Temporal Logic (STL) [25] can compactly express rich time series over various signals. The signal can be comprised of spatial data or another mission-related quantity. For example, STL can compactly and rigorously express mission requirements such as: "A drone should be visiting region A within the first 30 seconds for at least 5 consecutive seconds" and "another drone should be flying at most 5 meters away from the first one while keeping visiting region B every 15 seconds." STL is highly suitable to represent single- or multi-agent mission specifications, including but not limited to coordination constraints among the agents and other individual tasks with explicit deadlines. Syntax of STL that is used to generate temporal logic specifications is given as follows:

$$\phi ::= \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 U_{[t_1,t_2]} \phi_2, \tag{2.1}$$

where $t_1, t_2 \in \mathbb{Z}_{\geq 0}$ are discrete time bounds with $t_1 \leq t_2 < \infty$; $U_{[t_1,t_2]}$, $\neg$, $\wedge$ are until, negation, and conjunction operators, respectively; ϕ is an STL specification, and μ is a predicate in the inequality form such as $\mu := p(s) \sim \delta$ with a constant $\delta \in \mathbb{R}$, a signal $s : \mathbb{Z}_{\geq 0} \to \mathbb{R}^n$ mapping each time instant t to a signal value, and a function $p : \mathbb{R}^n \to \mathbb{R}$ where $\sim \in \{\leq, \geq\}$. More operators are generated from the others as follows: $F_{[t_1,t_2]}\phi := \bot U_{[t_1,t_2]}\phi$ is finally (i.e., eventually), $G_{[t_1,t_2]}\phi := \neg F_{[t_1,t_2]} \neg\phi$ is globally (i.e., always) operator, and $\phi_1 \vee \phi_2 := \neg(\neg\phi_1 \wedge \neg\phi_2)$ is disjunction operator. An implication operator can also be defined as $\phi_1 \Rightarrow \phi_2 := \neg\phi_1 \vee \phi_2$.

For the signal s representing the run of the system, we denote s_t as the value of s at time t and (s,t) as the part of the signal that is a sequence of $s_{t'}$ for $t' \in [t,\infty)$. Satisfaction by the signal (s,t) is then determined as follows:

$$
\begin{aligned}
&(s,t) \models \mu \iff p(s_t) \sim \delta,\\
&(s,t) \models \neg\mu \iff \neg\big((s,t) \models \mu\big),\\
&(s,t) \models \phi_1 \wedge \phi_2 \iff (s,t) \models \phi_1 \text{ and } (s,t) \models \phi_2,\\
&(s,t) \models \phi_1 \vee \phi_2 \iff (s,t) \models \phi_1 \text{ or } (s,t) \models \phi_2,\\
&(s,t) \models \phi_1 U_{[t_1,t_2]} \phi_2 \iff \exists t' \in [t+t_1, t+t_2],\ (s,t') \models \phi_2 \text{ and } \forall t'' \in [t,t'],\ (s,t'') \models \phi_1,\\
&(s,t) \models G_{[t_1,t_2]} \phi \iff \forall\ t' \in [t+t_1, t+t_2],\ (s,t') \models \phi,\\
&(s,t) \models F_{[t_1,t_2]} \phi \iff \exists\, t' \in [t+t_1, t+t_2],\ (s,t') \models \phi.
\end{aligned}
$$

In English, while $(s,t) \models F_{[t_1,t_2]}\phi$ implies that ϕ must hold at least in one time instant between $[t+t_1, t+t_2]$, $(s,t) \models G_{[t_1,t_2]}\phi$ requires the satisfaction of ϕ at all time instants within the same interval. On the other hand, $(s,t) \models \phi_1 U_{[t_1,t_2]}\phi_2$ means that ϕ_1 needs to hold until ϕ_2 is becomes true which occurs eventually within $[t+t_1, t+t_2]$.

Note that, in this dissertation, in case of a negation operator inside the predicates, the associated predicate is converted to its negated version, e.g., $\neg(p(s) \geq \delta) = p(s) < \delta$. Any STL specification can be represented in negation-free form (Positive Normal Form (PNF) [36]).

Definition 2.2.1 (STL Horizon)**.** *The horizon of an STL specification ϕ, i.e., $hrz(\phi)$, can be defined as the minimum time required to decide whether the specification is satisfied or not [37]. Formally, $hrz(\phi)$ is inductively computed as below:*

$$
\begin{aligned}
&\mu = p(s) \sim \delta \Longrightarrow hrz(\mu) := 0,\\
&\phi = \neg\varphi \Longrightarrow hrz(\phi) := hrz(\varphi),\\
&\phi = \bigwedge_{i=1}^{m} \varphi_i \ \ or \ \ \bigvee_{i=1}^{m} \varphi_i \Longrightarrow hrz(\phi) := \max_{i\in\{1,\dots,m\}} hrz(\varphi_i),\\
&\phi = \varphi_1 U_{[t_1,t_2]} \varphi_2 \Longrightarrow hrz(\phi) := \max\big(hrz(\varphi_1)+t_2, hrz(\varphi_2)+t_2\big),\\
&\phi = G_{[t_1,t_2]}\varphi \ \ or \ \ \phi = F_{[t_1,t_2]}\varphi \Longrightarrow hrz(\phi) := hrz(\varphi) + t_2.
\end{aligned}
\tag{2.2}
$$

For instance, the specification $G_{[0,7]}F_{[0,3]}x \geq 0$ has a horizon of $7 + 3 = 10$, as the last interval for the periodic satisfaction of $x \geq 0$ becomes $[7, 10]$. Furthermore, $F_{[0,13]}x \geq 0 \wedge G_{[0,7]}x \geq 10$ has a horizon of $\max(13, 7) = 13$, as the STL task with the longer horizon determines the horizon of the entire specification. In this dissertation, the horizon of the mission is assumed to be equal to the horizon of the STL specification as $H = hrz(\Phi)$. However, the actual mission horizon can be extended due to actual requirements during the mission, such as the relaxation of STL specification deadlines.

Apart from other temporal logics most of which denote the satisfaction of a specification as either *True* or *False*, STL is endowed with a metric called "space robustness" that quantifies how well (or how bad) an STL specification is satisfied (or violated) by measuring the distance to violation (or satisfaction).

Definition 2.2.2 (STL Space Robustness)**.** *Space robustness, $\rho(s, \phi, t) \in \mathbb{R}$, is a real-valued function that is used to quantify the satisfaction of an STL specification ϕ with respect to a signal (s, t). The space robustness metric can be formally and recursively defined as follows [26]:*

$$
\begin{aligned}
\rho(s, p(s) \geq \delta, t) &:= p(s_t) - \delta, \\
\rho(s, \neg(p(s) \geq \delta), t) &:= -\rho(s, p(s) \geq \delta, t), \\
\rho(s, \phi_1 \wedge \phi_2, t) &:= \min\big(\rho(s, \phi_1, t), \rho(s, \phi_2, t)\big), \\
\rho(s, \phi_1 \vee \phi_2, t) &:= \max\big(\rho(s, \phi_1, t), \rho(s, \phi_2, t)\big), \\
\rho(s, \phi_1 U_{[t_1,t_2]} \phi_2, t) &:= \max_{t' \in [t+t_1, t+t_2]} \Big(\min\big(\rho(s, \phi_2, t'), \min_{t'' \in [t, t']} (\rho(s, \phi_1, t''))\big)\Big), \\
\rho(s, F_{[t_1,t_2]} \phi, t) &:= \max_{t' \in [t+t_1, t+t_2]} \rho(s, \phi, t'), \\
\rho(s, G_{[t_1,t_2]} \phi, t) &:= \min_{t' \in [t+t_1, t+t_2]} \rho(s, \phi, t').
\end{aligned}
\tag{2.3}
$$

While positive space robustness indicates the satisfaction of the specification ϕ (i.e., $\rho(s, \phi, t) > 0 \Rightarrow (s, t) \models \phi$), negative one represents violation (i.e., $\rho(\mathbf{x}, \phi, t) < 0 \Rightarrow (s, t) \not\models \phi$). Although zero space robustness is generally considered inconclusive, we consider this case to be satisfactory throughout this dissertation, i.e., $\rho(s, \phi, t) \geq 0 \Rightarrow (s, t) \models \phi$. Note that the space robustness of an STL specification is calculated via recursive min and max functions as shown in (2.3), which are piecewise linear, nonsmooth, and nonconvex. Due to several potential local minimums and nonsmoothness,

these functions can be difficult to handle in an optimization process [38], [14].

Example 2.2.1. *Consider a signal s. Assume that we have some constraints on this signal such that it needs to be* always *between $0.8 \leq s \leq 1.2$ along a mission with the horizon of 10 s. This specification is represented as an STL specification, φ_1, in Fig. 2.1. Note that this STL specification is violated since the signal is out of the desired region for $t \in (2, 5)$. Moreover, the space robustness is $\rho(s, \varphi_1, 0) = \min_{t\in[0,10]} \big[\min\big((1.2 - s_t), (s_t - 0.8)\big)\big] \approx -0.4 < 0$ which is determined by the furthest distance to the boundary (happening at $t = 3$). Now let another specification, φ_2, be "eventually the signal must satisfy $s \geq 1.2$ for any two consecutive time units along a mission horizon of 10 s." This requirement can be expressed with a combination of* finally *and* globally *operators with the given predicate. As shown in Fig 2.1, the signal satisfies $s \geq 1.2$ for $t \in [2, 5]$, which implies the satisfaction of the specification with a space robustness of $\rho(s, \varphi_2, 0) = \max_{t\in[0,8]} \big(\min_{t'\in[t+0,t+2]}(s_{t'} - 1.2)\big) \approx 0.13 \geq 0$.*

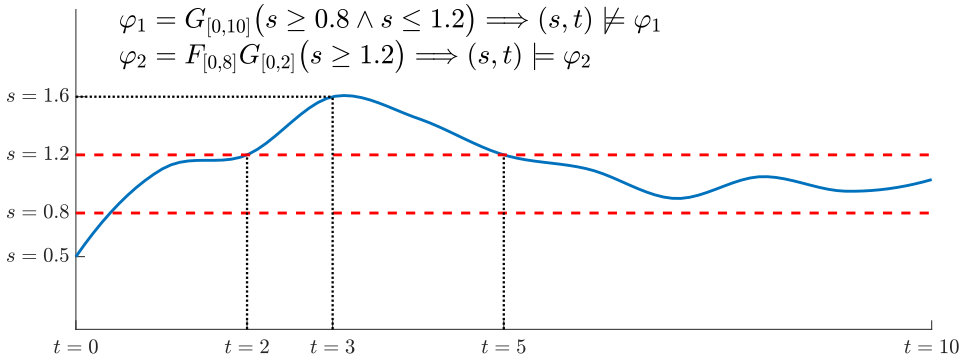


Figure 2.1: Evolution of signal s (in blue) in time along with the linear constraints in red that constitute the predicates of the STL specifications. While φ_1 constrains s to stay between the bounds all the time (violated), φ_2 requires $s \geq 1.2$ for at least two consecutive time units, which is achieved successfully.

A combination of multiple hyperplanes, e.g., the intersection of lines and the respective linear inequalities (predicates), can be used in STL specifications to define a requirement of being inside regions of interest. Let a region be bounded by the lines of $x = 1$, $x = 2$, $y = 3$, and $y = 4$ forming a square. Then we can use *conjunction* to define the region as

$$\mathit{Region} := \big(x \geq 1 \wedge x \leq 2 \wedge y \geq 3 \wedge y \leq 4\big). \tag{2.4}$$

Suppose that an agent has a specification of $\phi_1 := G_{[0,\tau]}\big(Region\big)$; this would require the agent to stay inside the region, i.e., satisfy the inequalities in (2.4), for all the time instants between $t = 0$ and $t = \tau$. Now going further, let another specification be defined as $\phi_2 := F_{[0,T]}\phi_1$, i.e., $\phi_2 = F_{[0,T]}G_{[0,\tau]}\big(Region\big)$. This requires the agent to visit region A for τ consecutive seconds starting at any time between $t = 0$ and $t = T$. The horizon of such a specification would be $hrz(\phi_2) = T + \tau$ which is mainly the time required to decide whether the specification is satisfied or violated [37]. In many applications of cyber-physical systems, more than one region of interest is tried to be serviced. Assuming we have three regions of interest, namely A, B, and C with the respective servicing requirements, one compact STL specification to represent all spatio-temporal goals can be defined by the conjunction operators as

$$\phi := F_{[0,T]}G_{[0,\tau_1]}\big(Region\ A\big) \wedge F_{[0,T]}G_{[0,\tau_2]}\big(Region\ B\big) \wedge F_{[0,T]}G_{[0,\tau_3]}\big(Region\ C\big).$$

Definition 2.2.3 (STL Time Robustness)**.** *Right (+) and left (-) time robustness of an STL specification quantifies how much the signal s can be shifted either right or left in time so that the specification is still satisfied. The time robustness of s with respect to a predicate μ can be computed as [26]:*

$$\theta^{+}(s,\mu,t) := \mathcal{X}(\mu,t)\cdot\max\Big\{d \geq 0\ s.t.\ \forall t' \in [t,t+d], \mathcal{X}(\mu,t') = \mathcal{X}(\mu,t)\Big\},$$
$$\theta^{-}(s,\mu,t) := \mathcal{X}(\mu,t)\cdot\max\Big\{d \geq 0\ s.t.\ \forall t' \in [t-d,t], \mathcal{X}(\mu,t') = \mathcal{X}(\mu,t)\Big\},$$

where the characteristic function $\mathcal{X}(\mu,t)$ is

$$\mathcal{X}(\mu,t) := \begin{cases} -1, & p(s_t) < 0, \\ +1, & p(s_t) \geq 0. \end{cases}$$

After computing the right/left time robustness of a predicate, the rules in (2.3) can be used to quantify the overall time robustness of a signal with respect to an STL specification. There also exists a combined notion called space-time robustness, where the inequalities in the characteristic equation are defined based on a desired space robustness threshold [26].

In the remainder of this dissertation, the time of the signals starting from $t = 0$ is omitted when it is clear from the context, i.e., $(s, 0)$ may be denoted by s. While s is used to denote general signals, $\mathbf{x}$ is also used when the signal is comprised of the state of a dynamical system.

2.3 Graph Theory

A weighted graph is a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ where $\mathcal{V}$ is a set of nodes, and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of edges between the nodes with weight $\mathcal{W} : \mathcal{E} \to \mathbb{R}^+$. A node $v_i \in \mathcal{V}$ is said to be adjacent to another node $v_j \in \mathcal{V}$ if $(v_i, v_j) \in \mathcal{E}$. The *adjacency matrix* between v_i and v_j is defined as follows:

$$\mathcal{A}_{i,j} := \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathcal{E}, \\ 0 & \text{otherwise.} \end{cases}$$

A node v_i is said to be a neighbor of another node v_j if they are adjacent, i.e., $A_{i,j} = 1$. With a slight abuse of notation, nodes that can be reached from v_i at most h-hops (i.e., transitions) can be denoted by $\mathcal{N}^h(v_i) \subseteq \mathcal{V}$.

Let a sequence of nodes be represented by $\mathbf{S} = \{v_1, v_2, \ldots\}$, then the length of $\mathbf{S}$ is denoted by $|\mathbf{S}|$, i.e., the total number of nodes visited on the path of $|\mathbf{S}|$. The weighted graph distance between the nodes, v_i and v_j is equal to the cumulative weight of edges traversed along the shortest (minimum cumulative weight) path from v_i to v_j.

2.4 Time-Varying (Zeroing) Control Barrier Functions

A function $h : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz continuous at $\mathcal{X}$ if every point on $\mathcal{X}$ has a neighborhood $\mathcal{N}$ and there exists a constant $L > 0$ such that $\|h(\mathbf{x}) - h(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \mathcal{N}$. Moreover, zero superlevel set of $h(\cdot)$ is defined as $\{\mathbf{x} \in \mathcal{X} \,|\, h(\mathbf{x}) \geq 0\}$.

Let the state and input of a dynamical system be denoted by $\mathbf{x} \in \mathcal{X}$ and $\mathbf{u} \in \mathcal{U}$, respectively, where $\mathcal{X} \subseteq \mathbb{R}^n$ and $\mathcal{U} \subseteq \mathbb{R}^m$ are the state space and the admissible set of control, respectively. Moreover, a control affine system is given as,

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{2.5}$$

with locally Lipschitz $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n \times m}$.

Definition 2.4.1 (Forward Invariance)**.** *A set $\mathcal{C} \in \mathbb{R}^n$ is forward invariant with respect to* (2.5) *if for every $\mathbf{x}_0 \in \mathcal{C},\ \mathbf{x}_t \in \mathcal{C},\ \forall t \geq 0$.*

Control barrier functions (CBFs) are used in [39] to render the desired set of states forward invariant. That is, let $h_{CBF} : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}$ be a differentiable function. If the

system starts in the zero superlevel set $\mathcal{C} = \{\mathbf{x} \in \mathcal{X} \mid h_{CBF}(\mathbf{x}) \geq 0\}$, i.e., $\mathbf{x}_0 \in \mathcal{C}$, then the CBF constraints force the system to stay always inside the set $\mathcal{C}$.

The author of [40] uses time-varying CBFs to ensure forward invariance in sets changing with time, i.e., $\mathcal{C}_t$. In this case, a differentiable function $\mathfrak{h}(\mathbf{x}, t) : \mathcal{X} \times I \to \mathbb{R}^n$ is defined as a time-varying control barrier function in the time domain $I \subseteq \mathbb{R}_{\geq 0}$. The time-varying set $\mathcal{C}_t = \{\mathbf{x} \in \mathcal{X} \mid \mathfrak{h}(\mathbf{x}, t) \geq 0\}$ is rendered forward invariant by the input $\mathbf{u} \in \mathcal{U}$, if $\mathbf{x}_0 \in \mathcal{C}_0$ and there exists a locally Lipschitz continuous extended class $\mathcal{K}$ function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that

$$\sup_{\mathbf{u} \in \mathcal{U}} \frac{\partial \mathfrak{h}(\mathbf{x}, t)}{\partial \mathbf{x}}^T (f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}) + \frac{\partial \mathfrak{h}(\mathbf{x}, t)}{\partial t} + \alpha(\mathfrak{h}(\mathbf{x}, t)) \geq 0, \tag{2.6}$$

then $\mathbf{x}_t \in \mathcal{C}_t, \ \forall t \geq 0$, where extended class $\mathcal{K}$ functions, $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, are strictly increasing, i.e., $\alpha(x) > \alpha(y)$ if $x > y$, with $\alpha(0) = 0$. In the remainder of this dissertation, we use $\mathcal{C}$ to denote $\mathcal{C}_t$ for the sake of simplicity. With a slight abuse of notation, $\mathcal{C}_i$ will denote the feasible set associated with the i^{th} task in the mission specification. We encode desired specifications by using Signal Temporal Logic (STL), and time-varying CBFs will be utilized to enforce the satisfaction of the STL tasks.

Chapter 3

Enhancing the Expressiveness of Signal Temporal Logic

In several applications, the integrals and derivatives of signals carry valuable information (e.g., cumulative success over a time window, the rate of change) regarding the behavior of the underlying system. In this chapter, the expressiveness of Signal Temporal Logic (STL) is extended by introducing predicates that can define rich properties related to the integral and derivative of a signal. For control synthesis, the new predicates are encoded into mixed-integer linear inequalities and are used in the formulation of a mixed-integer linear program (MILP) to find a trajectory that satisfies an STL specification. The benefits of using the new predicates are illustrated and discussed in a case study, which shows the influence of the new predicates on the trajectories of an autonomous robot.

This chapter is organized as follows: In Sec. 3.1, a brief literature survey is presented regarding the use of different signal properties inside STL specifications, and the gap in the literature is motivated. Next, new predicates are introduced in Sec. 3.2, and the optimization problem that solves for the trajectories satisfying the specifications with the proposed syntax is formulated. Then, mixed-integer linear program (MILP) encodings of the new predicates are presented in Sec. 3.3. Simulation results of a case study with an autonomous robot are presented in Sec. 3.4 along with a brief discussion in Sec. 3.5.

3.1 Introduction and Literature Review

Standard STL capabilities are enhanced by several studies in the literature (e.g., [41–45]). For example, since conventional robustness metric in (2.3) focuses on critical time instants, authors of [41] and [42] define new measures to differentiate the satisfaction of predicates achieved at multiple time instants from the instantaneous satisfaction of them. These methods can be said to depend on the counting of the time instants at which a predicate is satisfied. With a similar motivation, a temporal operator is defined in [43] that explicitly specifies how long a predicate must be satisfied. However, existing approaches do not accommodate predicates expressing cumulative success or local transitional behavior over a time interval. To this end, this chapter introduces integral and derivative predicates for STL to specify more detailed queries about the signals. While the derivative predicate enables the definition of properties for the rate of change of the signal, the integral predicate allows the definition of properties such as average or cumulative progress at desired time intervals.

There is similar research in the literature that can be closely relevant to the underlying motivation in this chapter. For instance, the work in [46] defines predicates to compare signal values at different time instants to find the local extrema of the signal. Also, the authors of [47–49] propose the notions of cumulative and average robustness metrics calculated via the signal values at different time steps. With these methods, while the system can satisfy a conventional predicate as long and robustly as possible by maximizing the new metrics, it does not result in the satisfaction of cumulative properties, such as getting a specific amount of reward/value within a given time window. To this end, this chapter proposes to define new integral and derivative predicates that can take into account the signal values at different time steps to define cumulative and relative specifications within desired time windows. By the proposed predicates, the local and global signal characteristics can be controlled extensively without losing the existing STL capabilities.

Conventional robustness metric defined in (2.3) has limitations due to its main focus on the critical time instances and neglecting the remaining parts of the signal. For instance, suppose that eventually the value of $\mathbf{x}$ needs to be at least 1 within $[0, 100]$, i.e., $\phi = F_{[0,100]}\, \mathbf{x} \geq 1$. Consider two trajectories $\mathbf{x}$ and $\mathbf{x}'$ shown in Fig. 3.1. Since the

degree of satisfaction is evaluated over any critical instant at which the predicate $\mathbf{x}_t \geq 1$ is satisfied for $t \in [0, 100]$, although the signal $\mathbf{x}$ satisfies the predicate for a longer time, the robustness metrics of both signals would be equal, i.e., $\rho(\mathbf{x}, \phi, 0) = \rho(\mathbf{x}', \phi, 0)$. Therefore, the conventional robustness metric cannot differentiate between such cases and cannot provide comprehensive information about the signals $\mathbf{x}$ and $\mathbf{x}'$. To address this issue, [41–43] define new measures that can capture the duration of predicate satisfaction. While existing metrics can track how long a predicate $\mathbf{x}_t \geq \delta$ is satisfied within an interval $[0, H]$, they are not able to address a notion of cumulative success such as $\int_0^t \mathbf{x}_\tau d\tau \geq \delta$ within an interval $[0, H]$. While the duration of satisfaction of a

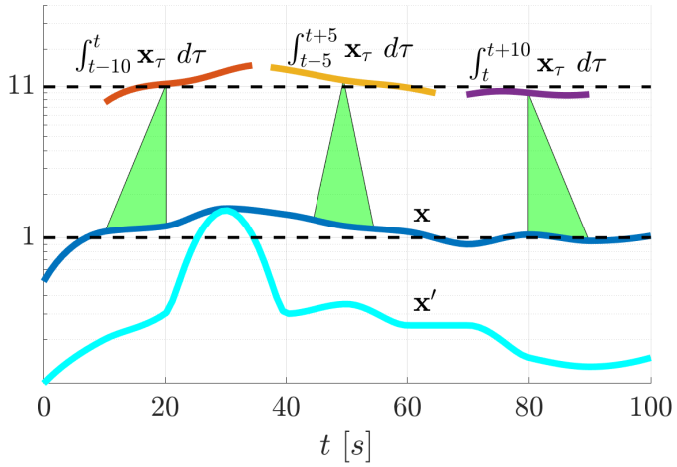


Figure 3.1: Signal functions $\mathbf{x}$ and $\mathbf{x}'$ in blue and cyan, respectively, in logarithmic scale, and portions of the curves generated by integrating $\mathbf{x}$ over three different relative time intervals. The values in the red, orange, and magenta curves are generated from the previous, closest, and next 10 s of the blue $\mathbf{x}$ curve, respectively.

conventional predicate may be important for certain applications; it may also be desired to meet some cumulative success criteria. For instance, in [44], some cumulative properties are defined over a swarm. Authors in [47–49] propose discrete-time cumulative and average robustness metrics calculated by the summation of the robustness metrics of the same predicate at different time steps. This enables a predicate to be satisfied as long and robust as possible. In [46], a freezing operator is used to store the signal values at different time instants in memory and to compare them inside the predicates, which enables the finding of local extrema and examining the oscillatory behavior. In

other words, the satisfaction of a predicate now depends on more than one signal value at different time instants. Similarly, in the case of progressive events, one may desire to evaluate the contributions of the past, future, or both signal values to determine satisfaction at the current time. Therefore, the success may depend on the accumulation of signal values from different time instants. The need for the definition of such success criteria inside the same predicate is still there, and this can be met by using the integral of a signal over a given bounded time interval in a new operator/predicate to assess satisfaction.

Example 3.1.1. *Consider a continuous signal function that represents the speed,* $\|v\|$*. Let the trajectory of* $\|v\|$ *be the same with the signal* $\mathbf{x}$ *in Fig. 3.1. Assume that at* $t = 20$*, the system is desired to travel more than* 11 m *within the last* 10 s*. In other words, integration of* $\|v\|$ *over* $[10, 20]$ *must be at least* 11*. One may check the satisfaction of this specification at* $t = 20$ *by inspecting the red curve in Fig. 3.1 obtained by the integration of* $\|v\|$ *(i.e.,* $\mathbf{x}$*) over the window of the previous* 10 s *at each time instant.*

The interval of interest over which the integrated signal may not depend only on the past values. One may also specify thresholds on the integrals defined partially or fully over the future signal values. In Fig. 3.1, sample intervals defined in the past, future, or both are partially shown with red, magenta, and orange, respectively. For each time instant, the window of 10 s *is shifted throughout the time axis to generate the integral curves. While at* $t = 50$*, the total distance traveled between* $[45, 55]$ *(depending on both past and future values) is more than* 11 m*, at* $t = 80$*, this time, the distance traveled during* $[80, 90]$ *(depending on only future values) is less than the given threshold.*

In the next section, novel STL predicates are introduced whose satisfaction can depend on the part of the past, present, or future of the signal. The new predicates are not the same as using the standard predicates with temporal operators. For example, consider some specifications defined between $t = 80$ and $t = 90$ over $\mathbf{x}$ represented in Fig. 3.1. Trajectory of $(\mathbf{x}, 80)$ does not satisfy the predicate of $\mathbf{x} \geq 5$ because $\mathbf{x}_{80} < 5$. Moreover, the trajectory $(\mathbf{x}, 0)$ does not satisfy an STL specification $F_{[80,90]}x \geq 5$ because there is no $t \in [80, 90]$ such that $\mathbf{x}_t \geq 5$. Now, consider another predicate including an integral over time whose satisfaction does not depend only on the current time step but is determined by the signal values within the bounds of the integral. The

same signal $(\mathbf{x}, 0)$ satisfies such a predicate $\int_{80}^{90} \mathbf{x}_\tau d\tau \geq 5$ as shown in Fig. 3.1 with magenta. Furthermore, temporal operators can also be used with such a predicate to create more complex specifications by shifting the integral interval.

One may try to represent such specifications by simply considering the integral or derivative of the original signal and using it with conventional predicates. For example, again consider the magenta signal in Fig. 3.1. Let the specification be "the integrations of $\mathbf{x}$ within the time windows of $[t+a, t+b]$ need always to be at most 11 for $t \in [70, 90]$, i.e., $G_{[70,90]} \int_a^b \mathbf{x}_\tau d\tau \leq 11$. This specification could be expressed by first defining a new signal depicting the integral of $\mathbf{x}$, i.e., $K(t) := \int_0^t \mathbf{x}_\tau d\tau$. Then, the requirement can be expressed $G_{[70,90]} K(t+b) - K(t+a) \leq 11$. While such predicates with multiple time instants (i.e., $K(t+b) - K(t+a)$) are not commonly used, they can be accommodated by the syntax and semantics of STL [50]. Alternatively, in the next section an *integral predicate* is introduced that can explicitly define certain progress over the given time interval, together with a *derivative predicate* to evaluate local characteristics of the signal. These predicates take time bounds as input and shift these bounds in accordance with the outer temporal operators. Accordingly, they do not require creating the integral and derivative of $\mathbf{x}$ as additional signals, which necessitates less memory and computations over the time interval of interest only.

3.2 Integral and Derivative Predicates

First, the integral predicate is introduced to express a specific amount of progress in preemptable and cumulative properties via STL specifications.

Definition 3.2.1 (Integral Predicate)**.** *An integral predicate over a bounded time interval $[a, b] \subset \mathbb{R}$ with $b > a$ is defined as:*

$$\mu^i_{[a,b]} := \int_a^b p(\mathbf{x}_\tau) d\tau \geq \delta \tag{3.1}$$

where $\delta \in \mathbb{R}$ is a constant, $\mathbf{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ is the signal, and $p : \mathbb{R}^n \to \mathbb{R}$ is an integrable function.

Notice that the integral predicate requires the input of the time bounds as the temporal operators like finally and globally. With the new integral predicate, one may

explicitly define these integral bounds and specify some progress threshold over a certain time interval.

Example 3.1.1 (*Cont'd*)**.** *The specifications mentioned in Example 3.1.1 and shown in Fig. 3.1 can now be represented with the integral predicates as follows. Both* $\mu^i_{[-10,0]}(t) = \int_{t-10}^{t} \|v\| \; d\tau \geq 11$ *defined on the past values (in red) and* $\mu^i_{[-5,5]}(t) = \int_{t-5}^{t+5} \|v\| \; d\tau \geq 11$ *depending on both past and future values (in orange) are satisfied for* $t = 20$ *and* $t = 50$*, respectively. However, for* $t = 80$ *the integral predicate* $\mu^i_{[0,10]}(t) = \int_{t}^{t+10} \|v\| \; d\tau \geq 11$ *that uses a future time interval (in magenta) is violated.*

In addition to cumulative properties, volatility over the signal is also possible. To explicitly limit such behavior, the derivative predicate is introduced.

Definition 3.2.2 (Derivative Predicate)**.** *A derivative predicate is defined to specify the queries on the rate of change of the signal function* $p(\mathbf{x})$ *with the time by the first derivative of it as follows:*

$$\mu^d_+ := \frac{dp(\mathbf{x})}{dt^+} \geq \delta \qquad \Big| \qquad \mu^d_- := \frac{dp(\mathbf{x})}{dt^-} \geq \delta,$$

where μ^d_+ *and* μ^d_- *denote the right and left derivative.*

By using the newly defined predicates together with the standard STL predicates, we can define diverse specifications related to the local and global characteristics of the signals according to the following STL syntax:

$$\phi ::= \mu \mid \mu^i_{[a,b]} \mid \mu^d_{+,-} \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid F_{[a,b]}\phi,$$

where the satisfactions of $\mu^i_{[a,b]}$ and $\mu^d_{+,-}$ are determined for any signal $\mathbf{x} : \mathbb{R}_{\geq 0} \to \mathbb{R}^n$ as follows:

$$\begin{aligned}
&(\mathbf{x}, t) \models \mu^i_{[a,b]} \iff \int_{t+a}^{t+b} p(\mathbf{x}_\tau) d\tau \geq \delta, \\
&(\mathbf{x}, t) \models \mu^d_+ \iff \frac{dp(\mathbf{x}_t)}{dt^+} \geq \delta, \\
&(\mathbf{x}, t) \models \mu^d_- \iff \frac{dp(\mathbf{x}_t)}{dt^-} \geq \delta, \\
&(\mathbf{x}, t) \models \neg\mu^i_{[a,b]} \iff \neg\big((\mathbf{x}, t) \models \mu^i_{[a,b]}\big), \\
&(\mathbf{x}, t) \models \neg\mu^d_{+,-} \iff \neg\big((\mathbf{x}, t) \models \mu^d_{+,-}\big).
\end{aligned}$$

Note that the signal $\mathbf{x}$ is undefined for $t < 0$; therefore, $t + a \geq 0$ is assumed throughout the chapter. By preserving the quantitative semantics for the common operators defined in (2.3), we can quantify the satisfaction of the new predicates similarly as:

$$\begin{aligned}
\rho(\mathbf{x}, \mu^i_{[a,b]}, t) &:= \int_{t+a}^{t+b} p(\mathbf{x}_\tau) d\tau - \delta, \\
\rho(\mathbf{x}, \mu^d_{+}, t) &:= \frac{dp(\mathbf{x}_t)}{dt^+} - \delta, \\
\rho(\mathbf{x}, \mu^d_{-}, t) &:= \frac{dp(\mathbf{x}_t)}{dt^-} - \delta, \\
\rho(\mathbf{x}, \neg\mu^i_{[a,b]}, t) &:= -\rho(\mathbf{x}_t, \mu^i_{[a,b]}, t), \\
\rho(\mathbf{x}, \neg\mu^d_{+,-}, t) &:= -\rho(\mathbf{x}_t, \mu^d_{+,-}, t).
\end{aligned} \tag{3.2}$$

Similar to conventional robustness metric in (2.3), a nonegative robustness metric indicates the satisfaction of the predicates, e.g., $\rho(\mathbf{x}, \mu^i_{[a,b]}, t) \geq 0 \implies (\mathbf{x}, t) \models \mu^i_{[a,b]}$, while negative one represents a violation $\big(\rho(\mathbf{x}, \mu^i_{[a,b]}, t) < 0 \implies (\mathbf{x}, t) \not\models \mu^i_{[a,b]}\big)$.

Remark 3.2.1. *In STL control synthesis, the satisfaction of temporal operators such as* finally *and* globally *at t are generally decided by the assessment of the future time steps $t' \geq t$. An integral predicate $\mu^i_{[a,b]}$ for $[a, b] \subset \mathbb{R}$, or the derivative one, $\mu^d_{+,-}$, can depend both on the past and future signal values by enabling negative time bounds.*

An integral predicate can have negative time bounds with respect to the bounds of the outer temporal operators (e.g., finally and globally). It can also be used alone by using nonnegative time bounds or defining it with the negative time bounds at future time steps. The horizon definition of STL [37] can be extended for the integral predicate and nested use of it with the temporal operators as follows:

$$\mu^i_{[a,b]} = \int_a^b p(\mathbf{x}_\tau) d\tau \geq \delta \implies hrz(\mu^i_{[a,b]}) := \max\big(|a|, b, b-a\big),$$
$$\phi = G_{[t_1,t_2]}\mu^i_{[a,b]} \text{ or } \phi = F_{[t_1,t_2]}\mu^i_{[a,b]} \implies hrz(\phi) := \max(t_2, t_2 + b),$$

where $t_1, t_2 \in \mathbb{R}_{\geq 0}$ and $a, b \in \mathbb{R}$ are time bounds with $t_2 \geq t_1$, $b > a$, and a constraint of $t_1 + a \geq 0$ for the sake of nonnegative global time.

Remark 3.2.2. *For discrete-time signals, we define the integral and derivative predicates at the discrete time, with a slight abuse of notation denoting discrete step numbers*

by $t \in \mathbb{Z}_{\geq 0}$, as follows:

$$\mu^{i}_{[a,b]} = \sum_{\tau=t+a}^{t+b-1} p(\mathbf{x}_\tau)\,\Delta t \geq \delta\,,$$
$$\mu^{d}_{+} = p(\mathbf{x}_{t+1}) - p(\mathbf{x}_t) \geq \delta\,\Delta t$$
$$\mu^{d}_{-} = p(\mathbf{x}_t) - p(\mathbf{x}_{t-1}) \geq \delta\,\Delta t,$$

where Δt is the time step.

3.2.1 Comparison with the Existing Metrics

The conventional robustness metric of STL evaluates a signal with respect to the critical time instants and neglects the remaining parts of the signal. To overcome this issue, the authors of [41] and [42] defined new metrics to capture the duration of predicate satisfaction compared to the best/worst value of the satisfaction. Similarly, the authors in [43] proposed a temporal operator to specify how long a predicate must be satisfied within a bounded time interval. Yet, the definition of success in such studies depends on the multiple instantaneous satisfactions of the predicates and does not include a notion of cumulative success.

Modified robustness metrics [47–49, 51] may also be used to quantify average and cumulative properties. These metrics may be used as predicates to specify thresholds over cumulative properties. However, they are defined by modifying the quantitative semantics of existing temporal operators such as "globally" and "finally". In this work, instead of modifying the semantics of existing temporal operators and losing standard capabilities, integral predicate — a new operator with its own qualitative and quantitative semantics— is used to quantify such properties, which can easily be combined with the standard STL syntax and semantics. For instance, consider a signal $\mathbf{x} = \{1, 1, 1, 1, 1, 2, \epsilon\}$, where ϵ is a small arbitrary number since $\mu^{i}_{[a,b]}$ does not consider the last signal value in discrete-time signals. Suppose that the sum of any two consecutive signal values needs to eventually be at least 3. The proposed integral predicate can be used to define this as $\varphi = F_{[0,4]}\mu^{i}_{[0,2]} \geq 3$ where $\mu^{i}_{[0,2]} = \int_0^2 \mathbf{x}\,dt$, for which the robustness of $\mathbf{x}$ with respect to φ is $\rho(\mathbf{x}, \varphi) = 0$. Thus, $\mathbf{x}$ barely satisfies the specification φ. Now, consider the *cumulative robustness metric*, $\rho^{+}(\cdot)$ in [47] that is the closest measure to the idea proposed in this chapter. One can express the task in φ by using

the cumulative robustness metric as $\varphi' = F_{[0,4]}\big(\rho^+(\mathbf{x}, F_{[0,1]}x \geq 0) \geq 3\big)$ for the same signal, which has $\rho^+(\mathbf{x}, \varphi') = -4$ implying a violation while $\mathbf{x}$ actually satisfies φ.

Furthermore, the difference is more dramatic when compared with [48,49,51], which also modify the quantitative semantics of STL, e.g., using "geometric mean." This yields a failure to decide the satisfaction of cumulative properties even with the thresholds applied to the robustness metrics. Moreover, depending on the sign of the signal values, modifications are required such as combining different metrics or using different temporal operators to represent the same cumulative properties for different signals.

Overall, no specific syntax in STL can capture the cumulative properties, and the proposed integral predicate is introduced to facilitate the definition of cumulative signal properties via STL.

3.2.2 Optimal STL Control Synthesis Problem

Consider a discrete-time system $\mathbf{x}^+ = f(\mathbf{x}, \mathbf{u})$ evolving over a continuous state space with state and input vectors of $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$, respectively. Accordingly, the signal takes the shape of a finite state trajectory, $\mathbf{x} = \begin{bmatrix}\mathbf{x}_0 \ \mathbf{x}_1 \ \cdots \ \mathbf{x}_H\end{bmatrix} \in \mathbb{R}^{n\times(H+1)}$ where H is the length of the mission horizon.

State trajectory $\mathbf{x}$ is desired to achieve an STL specification, Φ. Note that the mission horizon has to be longer than the specification horizon, i.e., $H \geq hrz(\Phi)$, to determine the satisfaction or violation of Φ.

Problem 3.2.1 (STL Control Synthesis)**.** *Given a system with discrete-time dynamics,* $\mathbf{x}^+ = f(\mathbf{x}, \mathbf{u})$*, find the optimal control policy over the horizon* $\mathbf{u}^* = \begin{bmatrix}\mathbf{u}_0^* \, \mathbf{u}_1^* \, \cdots \, \mathbf{u}_{H-1}^*\end{bmatrix} \in \mathbb{R}^{m\times H}$ *to achieve a global specification* Φ*:*

$$\begin{aligned} \mathbf{u}^* = &\arg\min \sum_{t=0}^{H-1} \mathcal{J}\big(\mathbf{x}_t, \mathbf{u}_t\big) \\ s.t. \quad &\mathbf{x}^+ = f(\mathbf{x}, \mathbf{u}), \\ &\mathbf{x}_0 = \mathrm{x}_0, \ \rho(\mathbf{x}, \Phi, 0) \geq 0, \end{aligned} \tag{3.3}$$

where $\mathcal{J}\big(\mathbf{x}_t, \mathbf{u}_t\big)$ *is a running cost as a function of state vector and control inputs,* $\mathcal{J} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$*;* x_0 *is the initial state vector, and* $\rho(\mathbf{x}, \Phi, 0) \geq 0$ *enforces the satisfaction of the temporal logic constraints by requiring the robustness metric to be nonnegative.*

3.3 Mixed-Integer Encoding of the STL Control Synthesis Problem

One major challenge in control synthesis problems with STL is representing the satisfaction of the desired STL specification in the optimization problem. For example, the satisfaction of Φ which is captured in (3.3) by $\rho\big(\mathbf{x}, \Phi, 0\big) \geq 0$ (implying $(\mathbf{x}, 0) \models \Phi$) where the space robustness function $\rho(\cdot)$ is nonlinear and nonsmooth $\big($due to the recursive use of min and max functions depending on the specification as defined in (2.3)$\big)$. In the literature, one frequently used approach to handle such discontinuities in the optimization problem is encoding them with a set of mixed-integer constraints [14]. Moreover, if the objective function $\mathcal{J}(\cdot)$ is assumed piece-wise linear (or quadratic), and predicates and dynamics are also linear, then the optimization problem becomes a mixed-integer linear (or quadratic) program that can be solved by off-the-shelf tools [52]. Alternatively, the space robustness function can be approximated by smooth functions (e.g., [27, 31]), and gradient descent techniques can be used to solve the optimization problem where nonlinear dynamics and predicates can be defined (i.e., nonlinear programming (NLP) problem) different than the mixed-integer programming (MIP) approach [53]. It is worth mentioning that such gradient descent-based techniques do not guarantee finding the global optimal trajectory. Therefore, the satisfaction of Φ is encoded as a set of constraints with binary variables in the form of $z_t^{\Phi} \in \{0, 1\}$ [4], [14]. For each predicate in the form of inequality, a couple of Big M constraints can be written depending on a binary variable as

$$\mu = p(\mathbf{x}_t) \geq \delta \iff \begin{cases} p(\mathbf{x}_t) - \delta \geq M(z_t^{\mu} - 1), \\ p(\mathbf{x}_t) - \delta \leq M z_t^{\mu} - \epsilon, \end{cases} \tag{3.4}$$

where $M, \epsilon \in \mathbb{R}^+$ are tuning parameters, being sufficiently large and small numbers, respectively. Similarly, the satisfaction of an integral predicate can also be encoded as

$$\phi = \mu^i_{[a,b]} \iff \begin{cases} \displaystyle\sum_{\tau=t+a}^{t+b-1} p(\mathbf{x}_\tau)\, \Delta t - \delta \geq M(z_t^{\phi} - 1), \\ \displaystyle\sum_{\tau=t+a}^{t+b-1} p(\mathbf{x}_\tau)\, \Delta t - \delta \leq M z_t^{\phi} - \epsilon. \end{cases} \tag{3.5}$$

The derivative predicate is encoded in a similar fashion as follows:

$$\mu_+^d = \frac{dp(\mathbf{x})}{dt^+} \geq \delta \iff \begin{cases} p(\mathbf{x}_{t+1}) - p(\mathbf{x}_t) - \delta\,\Delta t \geq M(z_t^{\mu_+^d} - 1), \\ p(\mathbf{x}_{t+1}) - p(\mathbf{x}_t) - \delta\,\Delta t \leq M z_t^{\mu_+^d} - \epsilon, \end{cases}$$

$$\mu_-^d = \frac{dp(\mathbf{x})}{dt^-} \geq \delta \iff \begin{cases} p(\mathbf{x}_t) - p(\mathbf{x}_{t-1}) - \delta\,\Delta t \geq M(z_t^{\mu_-^d} - 1), \\ p(\mathbf{x}_t) - p(\mathbf{x}_{t-1}) - \delta\,\Delta t \leq M z_t^{\mu_-^d} - \epsilon. \end{cases} \tag{3.6}$$

Starting with the predicates, the remaining binary constraints corresponding to an STL specification can be built into each other. The connections of the Boolean operators with other temporal operators and predicates can be constructed with the following rules and encoded as integer constraints considering two sample specifications ϕ and φ [14].

Negation: $\phi = \neg\varphi$

$$z_t^\phi = 1 - z_t^\varphi.$$

Conjunction: $\phi = \bigwedge_{i=1}^m \varphi_i$

$$\begin{aligned} z_t^\phi &\leq z_t^{\varphi_i}, \;\; i = 1, \ldots, m, \\ z_t^\phi &\geq 1 - m + \sum_{i=1}^m z_t^{\varphi_i}. \end{aligned}$$

Disjunction: $\phi = \bigvee_{i=1}^m \varphi_i$

$$\begin{aligned} z_t^\phi &\geq z_t^{\varphi_i}, \;\; i = 1, \ldots, m, \\ z_t^\phi &\leq \sum_{i=1}^m z_t^{\varphi_i}. \end{aligned}$$

Globally: $\phi = G_{[t_1,t_2]}\varphi$

$$z_t^\phi = \bigwedge_{\tau=t+t_1}^{t+t_2} z_\tau^\varphi.$$

Finally: $\phi = F_{[t_1,t_2]}\varphi$

$$z_t^\phi = \bigvee_{\tau=t+t_1}^{t+t_2} z_\tau^\varphi.$$

As a result, the satisfaction of the general specification Φ, which is previously implied by $\rho(\mathbf{x}, \Phi, 0) \geq 0$, is now rendered to $z_t^\Phi = 1$. As specifications over the state trajectory

are handled via temporal logic specifications, in the optimization, only the control input $\mathbf{u}_t$ is penalized. Therefore, the running cost is defined as the coordinate-wise absolute summation of the control input, i.e., $\mathcal{J}(\mathbf{u}_t) = \|\mathbf{u}_t\|_1$.

Nonlinear systems can be linearized around known equilibrium points. Furthermore, the mission constraints can be encoded as linear inequalities with binary variables as in (3.4), (3.5), (3.6). Accordingly, under linear dynamics and predicates, the constrained optimization problem in (3.3) can be posed as a mixed-integer linear program (MILP).

Problem 3.3.1 (MILP for STL Control Synthesis)**.**

$$\begin{aligned} \mathbf{u}^* = &\arg\min \sum_{t=0}^{H-1} \|\mathbf{u}_t\|_1 \\ s.t. \quad &\mathbf{x}_{t+1} = A\mathbf{x}_t + B\mathbf{u}_t, \\ &\mathbf{x}_0 = \mathrm{x}_0,\ z_0^{\Phi} = 1, \end{aligned} \tag{3.7}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are the system and input matrices of appropriate sizes, respectively.

Note that solving Problem 3.3.1 enforces the satisfaction of the STL specification (due to the existence of the constraint $z_0^{\Phi} = 1$). It can be solved via several off-the-shelf tools such as MATLAB's built-in integer programming solver, *intlinprog*, and Gurobi [54]. This problem can also be formulated for maximal satisfaction of the STL specification by relaxing each predicate with a slack variable to be minimized with the negation-free STL specifications [41], [42].

3.4 Case Study: STL Control Synthesis for Ground Robots

To illustrate the functionality of the new predicates, a complementary example is defined below.

Example 3.4.1. *Consider an autonomous robot with discrete-time double integrator dynamics*

$$\mathbf{x}_{t+1} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_t + \begin{bmatrix} 0.5\Delta t^2 & 0 \\ \Delta_t & 0 \\ 0 & 0.5\Delta t^2 \\ 0 & \Delta t \end{bmatrix} \mathbf{u}_t,$$

with the state vector $\mathbf{x} = [x, v_x, y, v_y]^T$ *where* v_x, $v_y \in \mathbb{R}$ *are the velocities in* x, $y \in \mathbb{R}$ *directions, respectively, and the input vector* $\mathbf{u} = [u_x, u_y]^T$ *where* u_x,$u_y \in \mathbb{R}$ *are the specific forces in given respective directions. The goal of the mission is to monitor the natural habitat in predefined areas by observing as much as possible without disturbing the members of it. Mission specifications are given as follows: i) Continuously service regions A and B for at least* 3 *s and* 6 *s, respectively, and while in B, travel inside it for at least* 2 *m in both directions. ii) Whenever in A or B, do not disturb the members of the natural habitat by limiting the acceleration to* 0.25 m/s^2 *in each direction. iii) Overall the robot cannot exceed an acceleration of* 0.5 m/s^2 *in any direction. iv) Since the habitants of region C are adversarial, whenever you are in C keep the horizontal velocity at least* 1 m/s *to leave the C immediately (its shortest edge lies in the horizontal direction). With a total mission time of* 20 *s, we can express these specifications as follows:*

$$\begin{aligned}\Phi = {} & F_{[0,17]}\big(G_{[0,3]}R_A\big) \wedge F_{[0,14]}\big(G_{[0,6]}R_B \wedge \mu^{i,1}_{[0,6]} \wedge \mu^{i,2}_{[0,6]}\big) \\ & \wedge G_{[1,20]}\big((R_A \vee R_B) \Rightarrow (\mu^{d,1}_{-} \wedge \mu^{d,2}_{-})\big) \\ & \wedge G_{[0,19]}\big(\mu^{d,3}_{+} \wedge \mu^{d,4}_{+}\big) \\ & \wedge G_{[0,20]}\big(R_C \Rightarrow |v_x| \geq 1\big),\end{aligned} \tag{3.8}$$

where the integral predicates are defined to constrain the total traveled distance as $\mu^{i,1}_{[0,6]} = \int_0^6 |v_x| d\tau \geq 2$ *and* $\mu^{i,2}_{[0,6]} = \int_0^6 |v_y| d\tau \geq 2$*; the derivative predicates limit the acceleration either temporarily or throughout the mission as* $\mu^{d,1}_{-} = |\dot{v}_x| \leq 0.25$, $\mu^{d,2}_{-} = |\dot{v}_y| \leq 0.25$, $\mu^{d,3}_{+} = |\dot{v}_x| \leq 0.5$, *and* $\mu^{d,4}_{+} = |\dot{v}_y| \leq 0.5$. *The reason for specifying left derivatives for the predicates inside A and B is to avoid an aggressive entrance into them. The regions in two-dimensional* $xy-$*plane can be represented with linear predicates as follows:*

$$\begin{aligned} R_A &= x \geq 1.5 \wedge x \leq 2 \wedge y \geq 4.75 \wedge y \leq 5.25, \\ R_B &= x \geq 4 \wedge x \leq 5 \wedge y \geq 1 \wedge y \leq 3, \\ R_C &= x \geq 2 \wedge x \leq 4 \wedge y \geq 1 \wedge y \leq 5. \end{aligned}$$

To bound the total traveled distance, integral predicates (Def. (3.2.1)) are defined to keep the distance in each particular direction at least 2 m by integrating the speeds in that direction. Moreover, the maximum acceleration in any direction is also constrained using derivative predicates defined in Def. 3.2.2. Although one can limit the control

inputs for this aim in this particular example, where the inputs are the acceleration, it is not generally the case for other systems. It is also worth mentioning that the expressions with the absolute value operators in (3.8) can be represented by multiple linear inequalities.

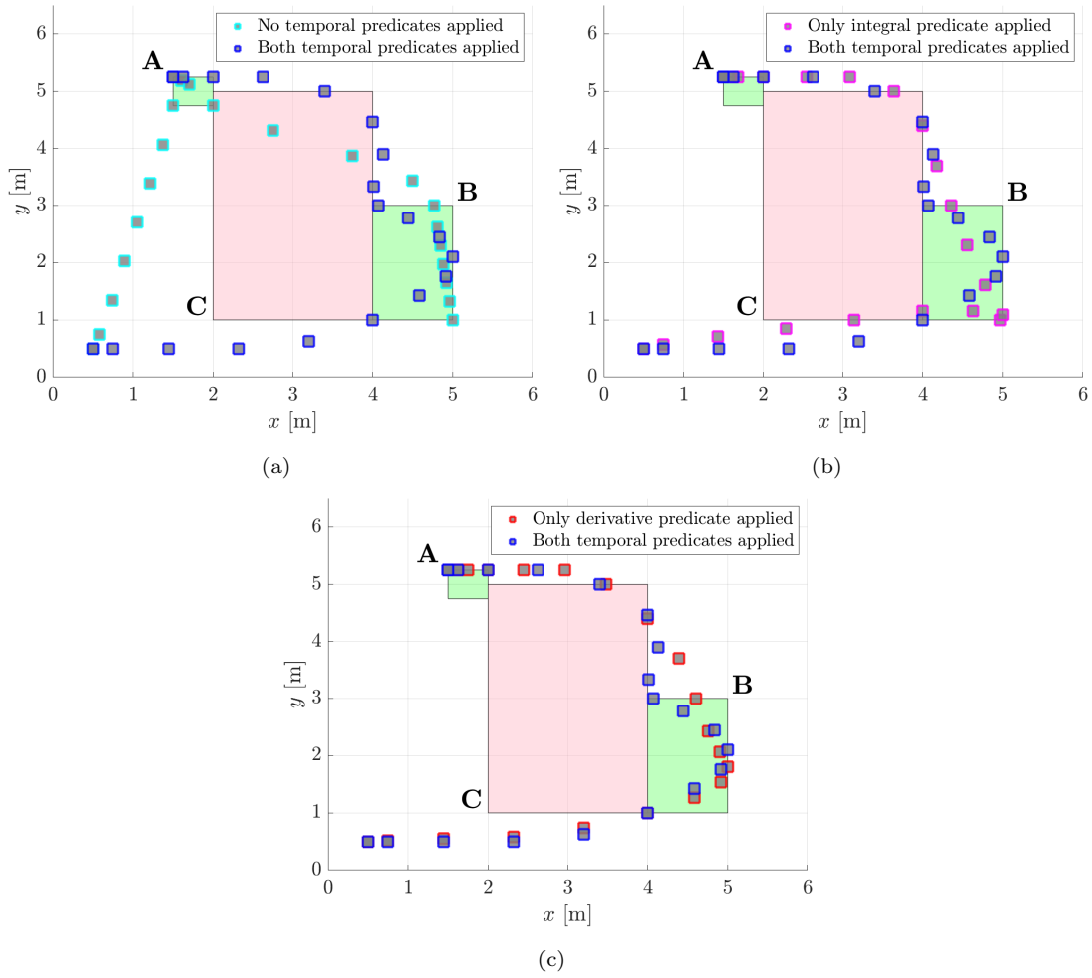
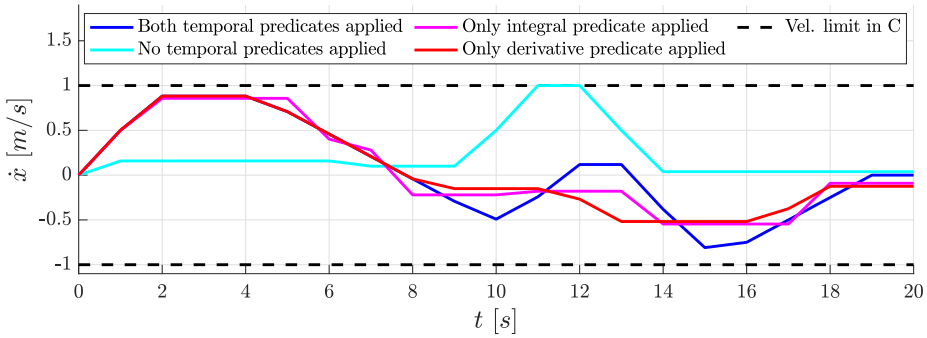


Figure 3.2: Comparisons of the blue trajectory that achieves the original specification with the ones achieving the same specification except a) the integral and derivative predicates (cyan), b) the derivative predicates (magenta) c) the integral predicates (red). All trajectories start from $(0.5, 0.5)$ m.

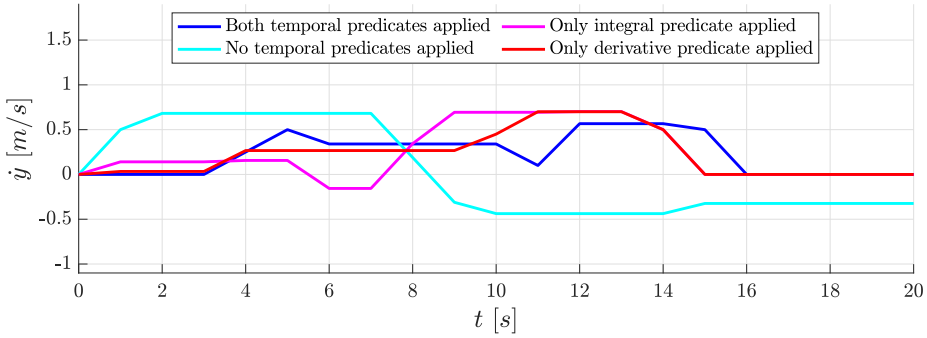
3.4.1 Simulation Results

A control synthesis tool that generates trajectories satisfying the given temporal logic specifications, including the new predicates, is developed. Specifications are not necessarily in negation-free form unlike [41], [42], and [47], in which the robustness metric is used directly in the algorithms. The specification is encoded as binary constraints as described in Sec. 3.3.

First, the system trajectory is computed according to the original specification (3.8). Then, for comparison, the system trajectories are computed to satisfy specifications 1) without the integral and derivative predicates defined in the green areas, 2) without the integral predicates, $\mu^{i,1}_{[0,6]}$ and $\mu^{i,2}_{[0,6]}$, and 3) without the derivative predicates, $\mu^{d,1}_{-}$ and $\mu^{d,2}_{-}$. The generated trajectories are illustrated in Fig. 3.2. It is evident from the figures that with the integral predicates, the robot does not only service B but is also required to travel inside the region more to monitor the natural habitat comprehensively. Moreover, the derivative predicates constrain the acceleration of the robot successfully both inside and outside of the regions. For instance, compared to no integral and derivative predicates case, given in cyan in Fig. 3.2a, when the region-based derivative constraint is applied, red in Fig. 3.2c, the robot cannot accelerate enough to enter region C and go around instead. This can also be observed in Fig. 3.3a where the only trajectory that enters the C is the cyan one by increasing its horizontal velocity to 1 m/s inside it. A similar effect of the derivative predicate can be seen in Fig. 3.4 in which the trajectories without it (cyan and magenta) violate the acceleration limit (0.25 m/s^2) in each direction inside the regions A and B. On the other hand, all four trajectories obey the general acceleration limit of 0.5 m/s^2. Finally, when there is no derivative predicate applied, the trajectory in magenta also avoids C since it has to slow down inside B to travel the minimum required distance. This makes the magenta trajectory a more aggressive one compared to the blue trajectory with both integral and derivative predicates inside B. The comparison of these four scenarios in terms of the control effort and computation time is also presented in Table 3.1.



(a)



(b)

Figure 3.3: For four different cases, change of velocities with time in a) x and b) y directions. The only trajectory that enters region C is the cyan one by increasing its speed to 1 m/s in x direction, which is a requirement for all cases.

Table 3.1: Results of the simulation scenarios in Fig. 3.2.

	Int. and Deriv. Predicates (blue)	Only Integral Predicates (magenta)	Only Derivative Predicates (red)	None of Them (cyan)
Total Cost $\mathcal{J}^*$ $[-]$	6.5363	4.8253	4.0749	3.9930
Solution Time $[s]$	4.53	15.43	1.13	2.29

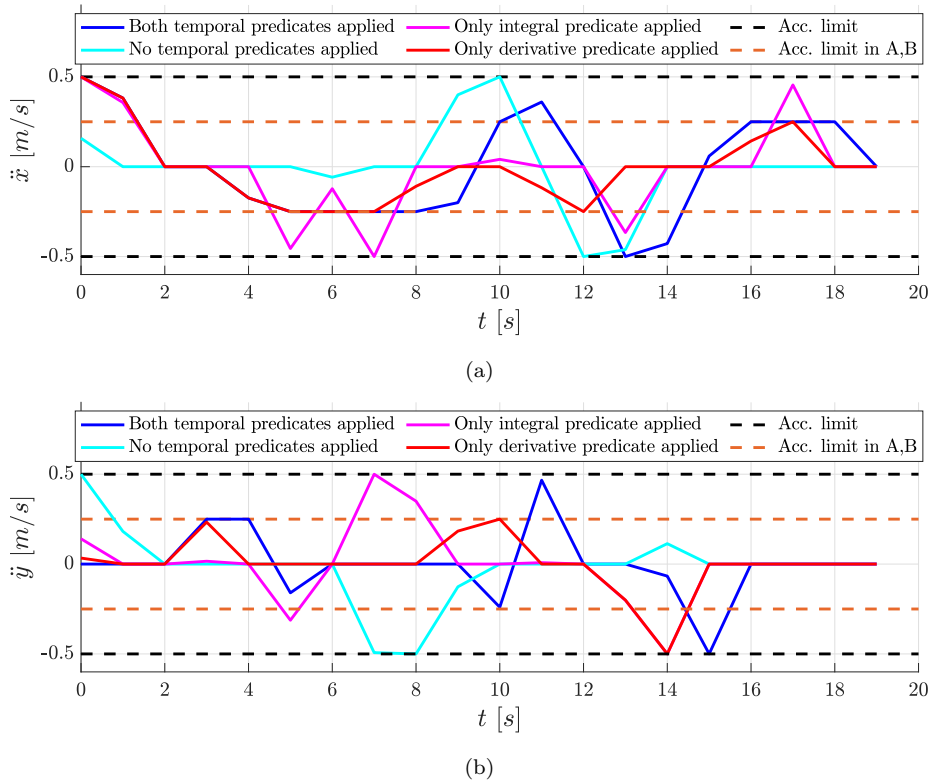


Figure 3.4: Change of accelerations with time in a) x and b) y directions, calculated via the right-derivative of the velocities. For all four cases, the general limit of 0.5 m/s^2 is obeyed. Moreover, blue and red trajectories satisfy a stricter limit of 0.25 m/s^2 defined by the left-derivative of the velocities inside the areas A and B.

3.5 Discussion

The proposed integral and derivative predicates for Signal Temporal Logic can be used to define specifications regarding the cumulative effects and the rate of change in a signal. As the new predicates can be encoded in terms of mixed-integer linear constraints, optimal trajectories satisfying the complex temporal logic specifications expressed by them can then be obtained by solving a mixed-integer linear program. The case study on the control of an autonomous robot in two-dimensional continuous space shows how the new predicates can be used in the design of the mission specifications more richly and expressively compared to the conventional STL. The next chapter investigates how multi-agent settings can benefit from these new capabilities.

Chapter 4

Scalable Operation of Heterogeneous Multi-Agent Systems under STL Specifications

This chapter addresses the problem of coordinating the trajectories of heterogeneous multi-agent systems under spatio-temporal specifications. In particular, global Signal Temporal Logic (STL) constraints are defined to express the number and type of agents that should be present at specific locations within the desired time windows. Moreover, integral predicates are used to specify cumulative progress that can be achieved asynchronously by multiple agents. In order to generate optimal trajectories, a mixed-integer linear program is formulated, whose objective is minimizing the agent movement subject to the heterogeneous abstracted dynamics of the agents and a global STL specification, including integral predicates.

In the second part of the chapter, energy constraints are also introduced to the coordination problem for agents assigned with cooperative tasks. Heterogeneous agents with stochastic energy dynamics are considered. The environment abstraction now involves the possible paths of different types of agents and ensures the availability of recharging within a certain distance. Then, another mixed-integer program over this abstraction is formulated to find the high-level paths of the agents. Next, the agents are steered in the environment according to the nominal plan under stochastic energy

consumption models and a recharging policy. Such stochastic energy dynamics cause deviations from the nominal plan and delay the completion of tasks. Accordingly, a preliminary metric is defined to evaluate the expected delay (temporal relaxation) in achieving the STL specification under the proposed solution.

This chapter is organized as follows: In Sec. 4.1, a brief literature survey is presented on the STL specifications for multi-agent settings. Next, service propositions are introduced in Sec. 4.2. Here we also discuss the environment model, multi-agent dynamics, and formulation of the heterogeneous multi-agent coordination problem under a global temporal logic constraint, along with a case study with simulations and a drone experiment. Then, Sec. 4.3 tackles the same problem subject to energy constraints and delays in task completion.

4.1 Introduction and Literature Review

In many applications, multi-robot systems with heterogeneous dynamics and capabilities must achieve tasks with complex specifications. For example, consider the following specification for a multi-robot surveillance mission: "At least one robot with a high-resolution camera needs to monitor region A every 5 minutes, and any two ground robots need to be at region B simultaneously within a time window $[t_1, t_2]$, and region C should be visited by a ground or aerial robot 10 times in total during the mission. On the other hand, existing work on multi-agent planning subject to STL constraints typically assumes initially given individual agent specifications or decomposes a team specification into individual ones, each of which is assigned to an agent (e.g., [7, 15]). However, as the number of agents increases, the complexity of planning significantly increases as well. In this regard, the work proposed in this chapter aims to satisfy team objectives as opposed to individual/local specifications. To achieve this, integral predicates are used in the STL specifications by allowing the definition of properties such as average or cumulative progress at desired time intervals. For example, some applications may require the agents to service a region multiple times within a given time window without requiring instantaneous and synchronous satisfaction. Multiple agents of different types can contribute to the progress over time in such *preemptable tasks*, which can be interrupted and continued later on. These specifications cannot

be defined via conventional STL predicates whose satisfactions depend only on a single time step. Unlike the recent studies of [47–49, 51], which propose modified quantitative semantics for STL, the new predicate definition preserves standard STL features and enables specifying cumulative properties.

There is a body of existing work focusing on the team specifications expressed via STL, e.g., [10, 29, 55–59]. Some of these work consider common specifications, assume a discrete abstraction of the environment, and solve an optimization problem that returns the optimal flow of agents from one discrete state to another. Such a formulation has the advantage of being independent of the number of agents. For example, the authors of [56, 57] control the flow of a specific number of agents in the presence of Linear Temporal Logic (LTL) specifications, which cannot define explicit time constraints. The authors of [55] allow for such deadlines; however, they consider identical agents. In [10, 59], the flow of agents that are heterogeneous only in terms of their capabilities is controlled, and this approach is extended with probabilistic transitions to account for uncertainties in robot motion in [58].

To summarize the main differences with the work highlighted in this chapter, none of those works can specify cumulative progress as captured by the integral predicates. Furthermore, most consider only identical agents and do not account for heterogeneity. Even though some consider heterogeneity in terms of agent capabilities, they do not account for heterogeneous dynamics, which is a significant challenge when planning the trajectories of different agent types. An approach is proposed in this chapter that allows for the definition of common objectives at the team level via integral predicates and the deployment of certain numbers of heterogeneous agents with specific capabilities and unique dynamics via STL specifications. An abstract model of the agent dynamics defining different velocities for different agent types is used to reflect heterogeneity among agents.

Moreover, the approach is also extended by considering a more realistic setting. In particular, a stochastic energy-consumption model is considered for each agent to ensure recharging availability while satisfying the given tasks. This is achieved by creating a discrete abstraction that guarantees the availability of recharging within a certain distance. Further, the plans are implemented under team-specific stochastic energy consumption throughout the mission. The proposed agent-level policies ensure the reachability of

charging stations in the environment whenever needed [60]. Although charging requirements may yield missing the deadlines of STL specifications, the expected amount of those delays are formally analyzed and an expected amount of delay, or *temporal relaxation*, is provided. Such a metric quantifies the needed relaxation of STL specifications with the introduced cumulative properties by formally accounting for the delays caused by agents deviating from their routes to get charged.

In [61], STL-satisfying trajectories are tried to be kept short, whose length can also serve as a proxy for energy consumption. However, none of these existing works address stochastic energy consumption and charging requirements in such multi-agent coordination problems. Furthermore, energy consumption is treated as the primary reason for stochasticity in this chapter. However, the approach is not restricted to this. For example, in another scenario, agents with limited storage capacities can collect an uncertain amount of samples and may need to empty their storage into stations frequently. Therefore, the framework presented in this chapter can be used for other risk- or resource-aware planning and execution strategies as well.

4.2 Multi-Agent Coordination Problem under STL Specifications with Integral Predicates

This chapter aims to generate coordinated trajectories for robots with heterogeneous capabilities and dynamics to achieve cooperative tasks with complex spatio-temporal specifications. For example, the agents can be asked to perform several missions in more than one region within different time windows. Also, some parts of the mission may require collaboration among agents of different types (such as drones with high-resolution cameras and ground robots with heat sensors).

Another type of mission that requires coordination is the achievement of preemptable tasks, i.e., tasks that require an accumulation of certain services over time. For example, an agent may start collecting data in a region and then leave before sampling the whole region to serve a different task. In that case, some other agent may later go to the same region and finish the data collection task by sampling the remaining parts.

4.2.1 Formulation of the Coordination Problem

Suppose that a heterogeneous multi-agent system with $n^{team} \in \mathbb{Z}^+$ different types of agents is initially distributed over an environment. Each team is comprised of $n_w^{agent} \in \mathbb{Z}^+$ identical agents with the single integrator dynamics of $\mathbf{x}_{w,i}(t{+}1) = \mathbf{x}_{w,i}(t){+}\mathbf{v}_{w,i}(t)\Delta t$ for $i \in [1, n_w^{agent}]$ and $w \in [1, n^{team}]$. Regions of interest over which the tasks are defined are denoted by P. Let Φ be a global STL specification with the horizon of $hrz(\Phi)$, including swarm service requirements represented via integral predicates. Note that using conventional STL, Φ cannot define tasks that include cumulative properties such as reward collection or any other preemptable task without the newly defined predicates.

An optimal control synthesis problem can be formulated to optimize an objective function (e.g., minimizing the total agent movement) subject to the swarm dynamics and the satisfaction of Φ.

Problem 4.2.1 (STL Satisfaction with Team-Level Tasks)**.** *Let a multi-agent system be comprised of n^{team} agent teams, each having n_w^{agent} identical agents, moving in an environment including a set of regions of interest P. Given a global STL specification Φ, find optimal agent velocities $\mathbf{v}^* = \big[\bigcup_w \bigcup_i \mathbf{v}_{w,i}(0) \ \ldots \ \bigcup_w \bigcup_i \mathbf{v}_{w,i}(H-1)\big]$ for $i \in [1, n_w^{agent}]$ and $w \in [1, n^{team}]$ subject to the STL constraints and agent dynamics:*

$$
\begin{aligned}
\mathbf{v}^* = \arg\min \ & \sum_{t=0}^{H-1} \mathcal{J}\Big(\bigcup_w \bigcup_i \mathbf{v}_{w,i}(t)\Big) \\
s.t. \quad & \Big(\big[\cup_w \cup_i \mathbf{x}_{w,i}(0) \ \ldots \ \cup_w \cup_i \mathbf{x}_{w,i}(H)\big], 0\Big) \models \Phi, \\
& \mathbf{x}_{w,i}(t+1) = \mathbf{x}_{w,i}(t) + \mathbf{v}_{w,i}(t)\Delta t, \\
& \forall t \in [0, H-1], \ \forall i \in [1, n_w^{agent}], \ \forall w \in [1, n^{team}],
\end{aligned}
$$

where $\bigcup_w \bigcup_i \mathbf{x}_{w,i}$ is the aggregated agent paths, $H \geq hrz(\Phi)$ is the overall mission horizon, $\mathcal{J}(\cdot)$ is the running cost as a function of aggregated agent inputs, and $\big([\cup_w \cup_i \mathbf{x}_{w,i}(0) \ \ldots \ \cup_w \cup_i \mathbf{x}_{w,i}(H)], 0\big) \models \Phi$ enforces satisfaction of the STL specification.

Objectives in a multi-agent mission can include requirements defined over space and time, i.e., spatio-temporal specifications. These can be expressed via temporal logic using the regions/nodes of interest combined with the desired capability and number of agents that are needed to satisfy the mission specifications. The Problem 4.2.1

will be solved approximately after defining the team-specific environment and swarm dynamics. Next, the agent-type-specific environment definition is given based on the respective regions of interest.

4.2.2 Environment Model

When planning the paths for agents, a set of regions of interest, including the base and the service areas, need to be visited by the agents of the respective team. Accordingly, several pairwise paths between these regions for each agent team are constructed via sampling-based algorithms. The partitioned environments are modeled for each team w as a graph with certain transition characteristics $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, d_w)$ where

- $\mathcal{V}_w$ is the set of nodes containing the regions of interest and the intermediate nodes,
- $\mathcal{E}_w \subseteq \mathcal{V}_w \times \mathcal{V}_w$ denotes the set of transitions that can be traversed between two nodes at a time step,
- $d_w : \mathcal{E}_w \to \mathbb{R}^+$ is the cost representing the Euclidean distance between two nodes.

Such a model allows for the definition of heterogeneous dynamics for single integrator agents by setting different transition costs for agents moving with different speeds, e.g., $d_w/\Delta t \leq \mathbf{v}_{w,\max}$. For example, if team 1 agents are twice as fast as team 2 agents, $\mathcal{G}_1$ and $\mathcal{G}_2$ are built with $2d_1 = d_2$.

A sampling-based algorithm (e.g., RRT* [62]) is used to generate such team-specific environment models with structural modifications to account for team dynamics, e.g., the maximum speed agents from different teams can attain. The modifications involve the definition of unique transition costs in accordance with the maximum speed agents from different teams can attain.

4.2.3 Swarm Flow Dynamics

Consider n^{team} agent teams each of which is comprised of n_w^{agent} identical agents where $w \in W$ for $W = \{1, \ldots, n^{team}\}$. Discrete-time vectors can be constructed to track the global behavior of the swarm as $N_w(t) = [n(w,1,t)\ n(w,2,t) \ldots n(w,|\mathcal{V}_w|,t)]^T \in \mathbb{Z}_{\geq 0}^{|\mathcal{V}_w|}$, where $n(w,v,t) : W \times \mathcal{V}_w \times \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ refers to the number of team w agents at node

$v \in \mathcal{V}_w$ at time t. Since each agent is considered a point mass with single integrator dynamics, it can move to one of its adjacent nodes within a time step. Moreover, it is assumed that the nodes are large enough to accommodate all the agents required to be in that node simultaneously without collision, and collision avoidance is kept out of the scope of this chapter.

Swarm flow dynamics can be constructed to formulate the evolution of $N_w(t)$ for each team $w \in W$. The number of team w agents moving from i^{th} to j^{th} nodes at t is denoted as $u_{w_{i,j}}(t) \in \mathbb{Z}_{\geq 0}$. For any agent team w, the flow over the team-specific environment graph $\mathcal{G}_w$ can be represented as:

$$\mathrm{u}_w(t) := \begin{bmatrix} u_{w_{1,1}}(t) & \cdots & u_{w_{1,|\mathcal{V}_w|}}(t) \\ \vdots & \ddots & \vdots \\ u_{w_{|\mathcal{V}_w|,1}}(t) & \cdots & u_{w_{|\mathcal{V}_w|,|\mathcal{V}_w|}}(t) \end{bmatrix}.$$

Using $\mathrm{u}_w(t)$, the number of team w agents leaving and entering specific nodes can be found as column-vectors

$$\mathrm{u}_w^{out}(t) := \mathrm{u}_w(t)\ \mathbf{1}_{|\mathcal{V}_w|\times 1} \text{ and } \mathrm{u}_w^{in}(t) := \mathrm{u}_w^T(t)\ \mathbf{1}_{|\mathcal{V}_w|\times 1},$$

respectively, with $\mathrm{u}_w^{out}(t), \mathrm{u}_w^{in}(t) \in \mathbb{Z}_{\geq 0}^{|\mathcal{V}_w|\times 1}$. The following expression can be obtained next to determine the evolution of $N_w(t)$:

$$N_w(t+1) = N_w(t) + \mathrm{u}_w^{in}(t) - \mathrm{u}_w^{out}(t). \tag{4.1}$$

Furthermore, the system should also satisfy the flow conservation and feasibility constraints below,

$$N_w(t) \geq \mathrm{u}_w^{out}(t), \tag{4.2a}$$

$$\mathcal{A}_{w_{i,j}} = 0 \Longrightarrow u_{w_{i,j}} = 0, \tag{4.2b}$$

where $\mathcal{A}_{w_{i,j}}$ indicates adjacency between i^{th} and j^{th} nodes of $\mathcal{G}_w$. While (4.2a) ensures that the number of agents leaving a node cannot be larger than the number of agents present at that node, (4.2b) prevents the flow of agents to the nodes which are infeasible to reach within a single time step.

4.2.4 Service Propositions

Although conventional STL can express rich time series, it is limited by the predicates whose satisfactions are determined considering the signal value at a single instant. In other words, these predicates are memoryless, and not all the signal values at multiple different time instants affect the satisfaction of a conventional predicate.

In the case of progressive events, on the other hand, evaluating the contributions of the past, future, or both signal values can help determine satisfaction at the current time. Generally, the success may depend on accumulating signal values at different time instants. In this regard, the integral predicate introduced in Chapter 3 can express a specific amount of progress in preemptable and cumulative properties via STL specifications.

Another notion that can be efficiently expressed via integral predicates over a given bounded time interval is preemptable tasks, which can be interrupted and continued later. A slightly modified version of the integral predicate in (3.1) is defined to account for multi-agent service tasks in discrete time.

Definition 4.2.1 (Integral Predicate for Service Tasks)**.** *For a signal $s : \mathbb{Z}_{\geq 0} \to \mathbb{R}^n$, an integral predicate over a bounded time interval $[a, b] \subset \mathbb{R}$ with $b \geq a$ is defined as $\mu^i_{[a,b]} = \sum_{\tau=a}^{b} prog(s(\tau)) \geq \delta$ where $\delta \in \mathbb{R}_{\geq 0}$ is a constant and $prog : \mathbb{R}^n \to \mathbb{R}$ is a function to evaluate the progress within $[a, b]$. The satisfaction of $\mu^i_{[a,b]}$ is determined at t as:*

$$(s,t) \vDash \mu^i_{[a,b]} \iff \sum_{\tau=t+a}^{t+b} prog(s(\tau)) \geq \delta,$$
$$(s,t) \vDash \neg\mu^i_{[a,b]} \iff \neg\big((s,t) \vDash \mu^i_{[a,b]}\big).$$

Integral predicates require the input of the time bounds as the temporal operators like finally and globally. The bounds of the integral can be explicitly defined, and a certain threshold on the progress over a certain time interval can be specified with the proposed integral predicate.

Remark 4.2.1. *An integral predicate $\mu^i_{[a,b]}$ for $[a, b] \subset \mathbb{R}$ can depend both on the past and future signal values by enabling negative time bounds. That said, as signals are*

undefined for $t < 0$; it is always assumed that $t + a \geq 0$ where $a \in \mathbb{Z}$ is the start of the time interval associated with any integral predicate.

Although the integral predicates can accommodate negative bounds, these are defined with respect to the time bounds of the outer temporal operators such as finally and globally. Additionally, the integral predicates can still be used alone by specifying nonnegative time bounds or defining them with the negative time bounds at future time steps. For a better understanding of the nested use of the integral predicate with the temporal operators, the reader can refer to (2.2), including the horizon definition for the STL specifications with integral predicate time bounds.

Definition 4.2.2 (Service Proposition)**.** *A service proposition, $S = \{p, c, \delta\}$, can be interpreted as: "within a given time interval, service c must be given in region p for at least δ many times." Consider multiple agent teams with $w \in W = \{1, \ldots, n^{team}\}$. The parameters used in the service proposition can be defined as:*

- *$p \in P$, the region to be serviced with P being the set of all regions of interest, i.e., $P = \bigcup_w P_w$,*
- *$c \in C$, the type of service to be given with C being the set of all service types,*
- *$\delta \in \mathbb{R}_{\geq 0}$, the total number of required service instances.*

Note that the service proposition S does not necessarily require synchronous and instantaneous satisfaction. Moreover, when combined with integral predicates as $\mu^i_{[a,b]}S$, the service propositions require a cumulative amount of the service within $[a, b]$. Then $\mu^i_{[a,b]}S$ is deemed satisfied as:

$$(s, t) \vDash \mu^i_{[a,b]}S \iff \sum_{\tau=t+a}^{t+b} prog(s(\tau)) \geq 1,$$

where, with a slight abuse of notation, $prog(s(\tau)) : W \times P \times \mathbb{Z}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denotes a progress function which maps each signal value at τ to a progress rate that needs to sum to at least 1 for satisfaction. The selection of $prog(s)$ depends on the requirements of the mission. For a heterogeneous multi-agent mission in which multiple agent teams can give the same service with different efficiency, it is defined as:

$$prog(s(t)) := \frac{\sum_{w=1}^{n^{team}} cap(w, c) \cdot n(w, p, t)}{\delta}, \tag{4.3}$$

where $n(w,p,t) : W \times P \times \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ is the number of agents from team w in region of interest p at t; $cap(w,c) : W \times C \to R_{\geq 0}$ maps how team w executes the task c. Although there may be several other metrics it can quantify, values of $cap(w,c) \in \{0,1,2\}$ are used in this work to indicate how fast the agents of team w can implement task c with its being zero implying the teams incapable of capability c.

The capability metric definition is similar to [59]; however, the formulation in (4.3), 1) assign how fast different agent types can perform a task, i.e., introduced weighted capabilities, 2) request the presence of certain agent types in addition to asking for random agent types with a specified capability. While $cap(w,c) = 0$ implies that the task c cannot be completed by type w agents, $cap(w,c) = 1$ and $cap(w,c) = 2$ denote the ability to service in regular and more efficient (twice faster) ways, respectively. Once again, the capability values are user-defined and subject to change as per task and agent type. For instance, let a specification be given as $\mu^i_{[0,10]}\{A, Take\ Picture, 20\}$ implying that region A must be imaged by agents equipped with a camera at least 20 times between $[0,10]$. Note that this cannot be satisfied by a single agent due to tight time constraints unless it can take two pictures at a time step. However, two agents would be able to satisfy it each by taking a picture per each time step in A for 10 time steps; or 20 agents could satisfy the specification by servicing A only for one time step. Such flexible specifications can be represented by the conventional STL syntax only in an impractical manner with the disjunction of all possible combinations as below:

$$
\begin{aligned}
\mu^i_{[0,10]} prog(s(t)) \geq 1 \cong & F_{[0,10]} prog(s(t)) \geq 1 \\
& \vee \big(F_{[0,9]} prog(s(t)) \geq 0.9 \wedge F_{[9,10]} prog(s(t)) \geq 0.1\big) \\
& \vee \big(F_{[0,9]} prog(s(t)) \geq 0.8 \wedge F_{[9,10]} prog(s(t)) \geq 0.2\big) \\
& \quad \vdots \\
& \vee \big(F_{[0,8]} prog(s(t)) \geq 0.9 \wedge F_{[8,10]} prog(s(t)) \geq 0.1\big) \\
& \vee \big(F_{[0,8]} prog(s(t)) \geq 0.8 \wedge F_{[8,10]} prog(s(t)) \geq 0.2\big) \\
& \quad \vdots
\end{aligned}
$$

Similarly, specification $\mu^i_{[0,20]}\{A, Spray, 40\}$ implies "region A must be sprayed by agents with at least 40 liters of pesticide between $t \in [0,20]$." Note that this task can be satisfied by an agent if it can spray two liters at a time step for 20 steps. Alternatively,

two agents would be able to satisfy it each by spraying one liter at each of 20 steps; or 40 similar agents could satisfy the task by servicing A simultaneously for one step.

Remark 4.2.2. *The use of integral predicate $\mu^i_{[a,b]}$ provides the flexibility of asynchronous achievement of the tasks. For the cases that require synchronicity, one can simply use $\mu^i_{[a,a]}$ which would imply the synchronous satisfaction of the task in the specified time $t = a$. In other words, if the synchronous presence of a certain number of agents with specific types or instantaneous achievement of a given task is needed in the time step a, one can use $\mu^i_{[a,a]}$ together with other timed operators instead of the conventional STL predicates.*

By combining agent types with previously given examples of tasks such as $C = \{Drone,\ Ground\ Vehicle,\ Take\ picture,\ Measure\ \ temperature,\ Collect\ sample\}$, and designating $cap^w_w = 1$ for all $w \in W$, type-specific missions can be defined as well. For instance, assume that 5 drones are desired to be in region B together at the current time step, then the specification could be written as $\mu^i_{[0,0]}\{B, Drone, 5\}$. Moreover, let region A need to be imaged 10 times at once (synchronously) in any time step within $[15, 25]$ then the specification would be $\phi = F_{[15,25]}\mu^i_{[0,0]}\{A, Take\ Picture, 10\}$.

With the newly defined tools, such as swarm flow dynamics and service propositions, Prob. 4.2.1 can be modified as follows.

Problem 4.2.2 (Cooperative STL Tasks over a Swarm)**.** *Let a multi-agent system comprising n^{team} agent teams evolve over the environment graphs $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, d_w)$ according to the dynamics of* (4.1), (4.2a), (4.2b) *for $w \in \{1, \dots, n^{team}\}$. Given a global STL specification Φ, find an optimal flow of agents $\mathrm{u}^*_w = \big[\mathrm{u}^*_w(0)\ \mathrm{u}^*_w(1) \cdots \mathrm{u}^*_w(H-1)\big] \in \mathbb{Z}^{|\mathcal{V}_w|\times|\mathcal{V}_w|H}_{\geq 0}$, $\forall w$ subject to the dynamics and STL constraints:*

$$
\begin{aligned}
\mathrm{u}^* = &\arg\min \sum_{t=0}^{H-1} \mathcal{J}\big(\mathrm{u}_1(t), \dots, \mathrm{u}_{n^{team}}(t)\big) \\
s.t.\quad & (4.1),\ (4.2a),\ (4.2b), \\
& N_w(0) = N_{w_0},\ \forall w \in \{1, \dots, n^{team}\}, \\
& \rho(N, \Phi, 0) \geq 0,
\end{aligned}
\tag{4.4}
$$

where $N = [N_1\ \dots\ N_{n^{team}}]$ and $\mathrm{u} = [\mathrm{u}_1\ \dots\ \mathrm{u}_{n^{team}}]$ are the aggregated agent distribution and flow policy, respectively, $H \geq hrz(\Phi)$ is the overall mission horizon,

$\mathcal{J}\big(\mathrm{u}_1(t),\ldots,\mathrm{u}_{n^{team}}(t)\big)$ *is the running cost as a function of agent flow,* N_{w_0} *is the initial agent distribution over the environment* $\mathcal{G}_w=(\mathcal{V}_w,\mathcal{E}_w,d_w)$*, and* $\rho(N,\Phi,0)\geq 0$ *enforces the satisfaction of STL specification by requiring the robustness value to be nonnegative.*

The Problem 4.2.2 is subject to cumulative success requirements defined as STL constraints via the integral predicates. Although positive space robustness (cf. (2.3) and (3.2)) enforces the satisfaction, it is computed via recursive definitions of computationally expensive min and max functions. Therefore, using it as a constraint in the optimization or feasibility problems makes the problem non-trivial again. An alternative way to represent the constraint of satisfying a specification, Φ, is encoding it as a set of mixed-integer constraints, as discussed next.

4.2.5 MILP Encoding of the Coordination Problem

After defining the agent-team-specific environment and swarm dynamics, the next task is searching for team plans over the built environment. These plans have to satisfy spatio-temporal mission specifications with cumulative success requirements defined via service propositions and integral predicates. The satisfaction of the STL specification Φ can be encoded as a set of constraints with binary variables, $z_t^{\Phi}\in\{0,1\}$ [4,14]. Similar to encoding of STL constraints via mixed-integer formulations in Sec. 3.3, satisfaction of integral predicates in Def. 4.2.1 is also encoded as:

$$\phi=\mu^i_{[a,b]}S \iff \begin{cases} \displaystyle\sum_{\tau=t+a}^{t+b} prog(s(\tau))-\delta\geq M(z_t^{\phi}-1), \\ \displaystyle\sum_{\tau=t+a}^{t+b} prog(s(\tau))-\delta\leq Mz_t^{\phi}-\epsilon. \end{cases}$$

The binary constraints corresponding to an STL formula can be built into each other, starting from the predicates. The remaining connections of the Boolean operators with the temporal operators can be constructed with the rules provided in Sec. 3.3 [4,14]. As a result, the overall satisfaction of the STL specification Φ reduces to $z_0^{\Phi}=1$.

In this work, minimization of the agent movement over the environment is considered while satisfying the desired STL specification for the sake of minimal energy consumption. To this end, a running cost is defined on the agent flow as $\mathcal{J}\big(\mathrm{u}_1(t),\ldots,\mathrm{u}_{n^{team}}(t)\big)=\sum_{w=1}^{n^{team}}\big\|\mathrm{u}_w(t)\big\|_1$. With the assumption of linear predicates, the optimization problem

previously presented in (4.4) including the robustness constraint can now take the form of a mixed-integer linear program (MILP) as follows.

$$\begin{aligned} \mathrm{u}^* = &\arg\min \sum_{t=0}^{H-1} \sum_{w=1}^{n^{team}} \left\| \mathrm{u}_w(t) \right\|_1 \\ s.t. \quad &(4.1),\ (4.2a),\ (4.2b), \\ &N_w(0) = N_{w_0},\ \forall w \in \{1, \ldots, n^{team}\}, \\ &z_0^{\Phi} = 1. \end{aligned} \tag{4.5}$$

The problem in (4.5) can also be solved via off-the-shelf tools such as Gurobi [54].

4.2.6 Case Study: A Cooperative Multi-Agent System

We consider three agent types with properties represented in Table 4.1, which are referred to by their shapes. The pairwise paths between regions of interest for each type are found via RRT* sampling-based algorithm (see Fig. 4.1 for sample paths) avoiding agent-type-specific obstacles on a 3.45×2.88 m environment represented in Fig. 4.2 [62].

Specifications for the multi-agent system can be summarized as follows: i) One diamond agent and one circle agent continuously and synchronously present at each bridge for any 8 seconds within $[10, 38]$. ii) A total of 50 temperature readings should be made on the service area in the middle within $[40, 60]$. iii) Top corner area should be imaged 15 times between $t = 40$ and $t = 60$. iv) Bottom corner area should be imaged

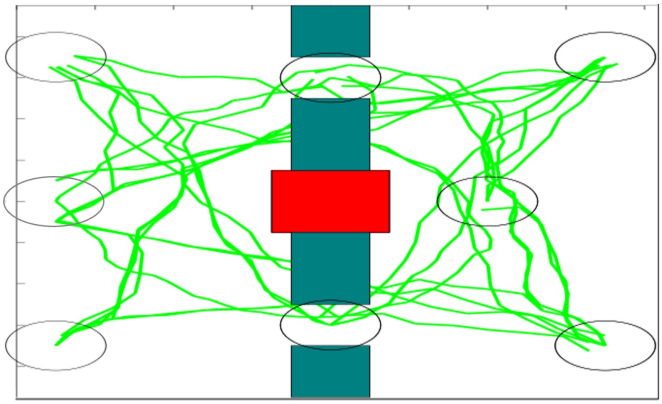


Figure 4.1: Pairwise obstacle-free RRT* paths between regions of interest.

Table 4.1: Properties of the three agent types.

Agent Type (Number)	Capability (per time step)	Regions of interest	Speed
Diamond (2)	Take picture ($\times 1$)	5	0.25 m/s
Triangle (2)	Take picture ($\times 1$) Measure temperature ($\times 1$)	4	0.25 m/s
Circle (2)	Measure temperature ($\times 2$)	4	0.125 m/s

15 times within $[40, 60]$. v) Whenever the diamond agents take pictures on both corners at the same time, all the diamond agents need to return to base within the next 10 seconds. Recall that n_w^{agent} is the number of type w agents. All of these specifications can be combined formally as,

$$\begin{aligned}\Phi =& F_{[10,30]} G_{[0,8]} \big(\mu^i_{[0,0]}\{Bridge_{top}, Circle, 1\} \\ & \wedge \mu^i_{[0,0]}\{Bridge_{top}, Diamond, 1\} \\ & \wedge \{\mu^i_{[0,0]} Bridge_{bot.}, Circle, 1\} \\ & \wedge \mu^i_{[0,0]}\{Bridge_{bot.}, Diamond, 1\}\big) \\ & \wedge \mu^i_{[40,60]}\{Target_{middle}, Measure\ temperature, 50\} \\ & \wedge \mu^i_{[40,60]}\{Target_{top\,corner}, Take\ picture, 15\} \\ & \wedge \mu^i_{[40,60]}\{Target_{bot.\,corner}, Take\ picture, 15\} \\ & \wedge G_{[0,60]}\Big(\big(\mu^i_{[0,0]}\{Target_{top\,corner}, Diamond, 1\} \\ & \qquad \wedge \mu^i_{[0,0]}\{Target_{bot.\,corner}, Diamond, 1\}\big) \\ & \qquad \implies F_{[0,10]}\mu^i_{[0,0]}\{Base, Diamond, n^{agent}_{diamond}\}\Big).\end{aligned} \tag{4.6}$$

Simulations are implemented for 70 seconds with $\Delta t = 2$, and assuming the presence of two agents of each type, i.e., $n_w^{agent} = 2,\ \forall w$. The distribution of agents moving over the environment is represented in Fig. 4.2 for selected time instants. Cumulative specifications defined by integral predicates are accomplished by the contribution of agents with different types. For instance, the imaging service in the bottom corner area is started by a diamond agent, then completed by the triangle agents, one of which also joined to the circle agents in the middle for temperature reading service. Moreover, the implication we specify caused the diamond agents not to service the different corner

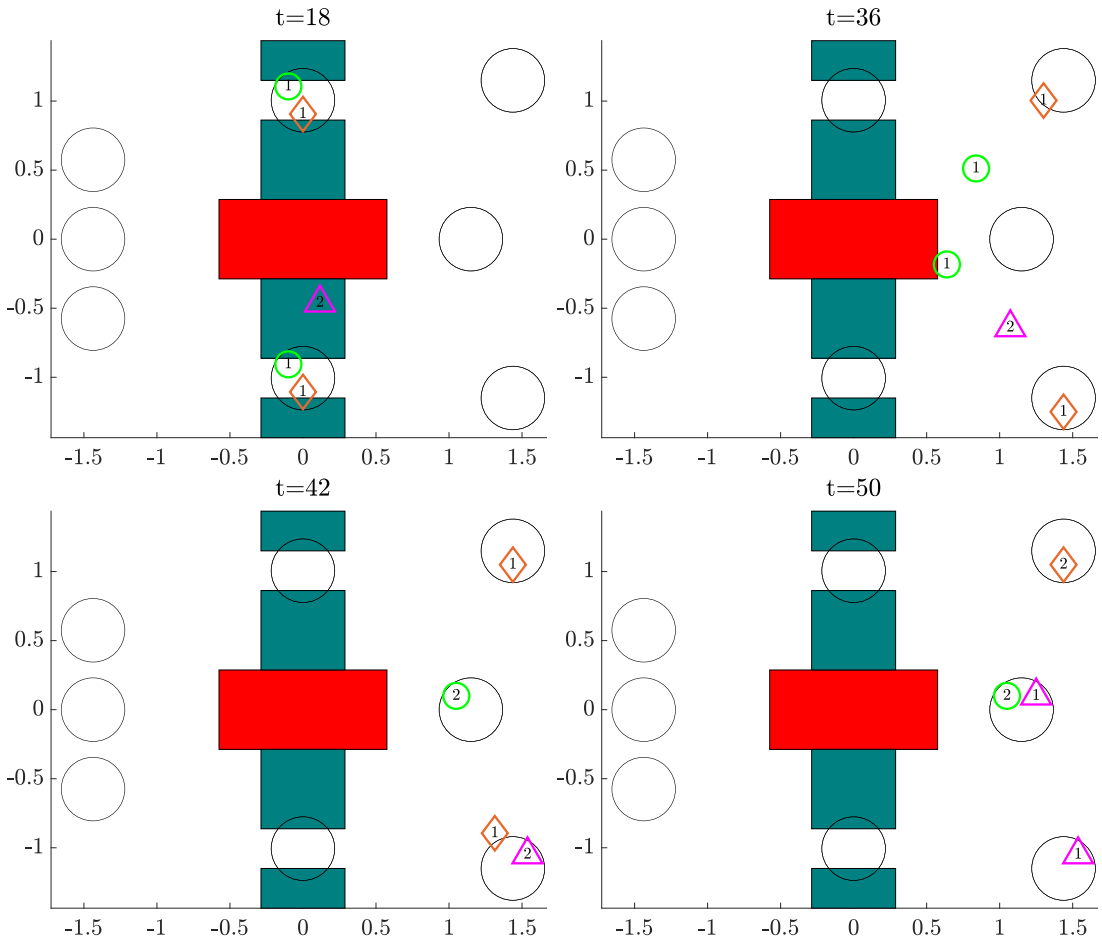


Figure 4.2: Team trajectories of diamond, triangle, and circle type agents at selected time instants (dark gray: obstacle for all agents, light gray: obstacles for circle agents only). While the values attached to markers indicate the number of agents from the respective type present at that node, larger circles represent all regions of interest.

regions at the same time.

To illustrate the scalability of our approach, the same mission (4.6) is executed by teams of different sizes, and the results are given in Table 4.2. We observe that the increasing number of agents has no significant adverse effect on the size of the problem and the complexity for the same STL specification and number of agent types. The number of constraints in the optimization problem is 2537 for the original specification and 1876 for the case without implication. These values remain the same with the changing number of agents. In the case of larger environments and more complex

specifications with long time horizons, the size of the problem would increase as expected though [14]. It is also clear from the Table 4.2 that the *implication* in (4.6) has a high impact on the solution time. This is mostly due to the *negation* operator we used to define the implication.

Although the results imply an inconvenience for real-time application, we consider high-level, offline planning, and the specification of rich tasks with cumulative properties is our main goal. Furthermore, with a limited number of integral predicates and without any implication or negation, we can define similar missions that are much easier to solve.

Table 4.2: Computation times for the different number of agents with three agent types designated as $n^{agent}_{diamond} + n^{agent}_{triangle} + n^{agent}_{circle}$.

Agent Number	$2+2+2$	$2+5+10$	$10+10+10$
Problem Construction Time $[s]$	0.93	0.94	1.01
Solver Time $[s]$	147.70	117.72	154.39
Problem Construction Time $[s]$ (w/o implication)	0.83	0.84	0.80
Solver Time $[s]$ (w/o implication)	64.95	52.59	64.76

An experiment is also conducted for the same scenario satisfying (4.6) for a similar environment via Crazyflie 2.0 platforms presented with a snapshot in Fig. 4.3.

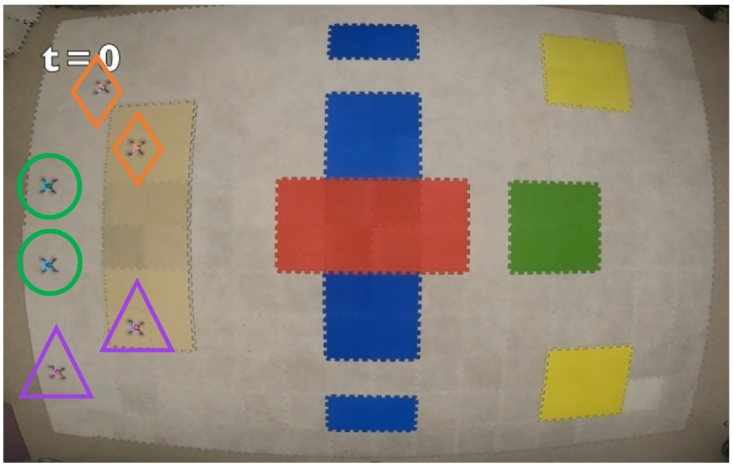


Figure 4.3: Snapshot from the experiment with six drones (two in each team).

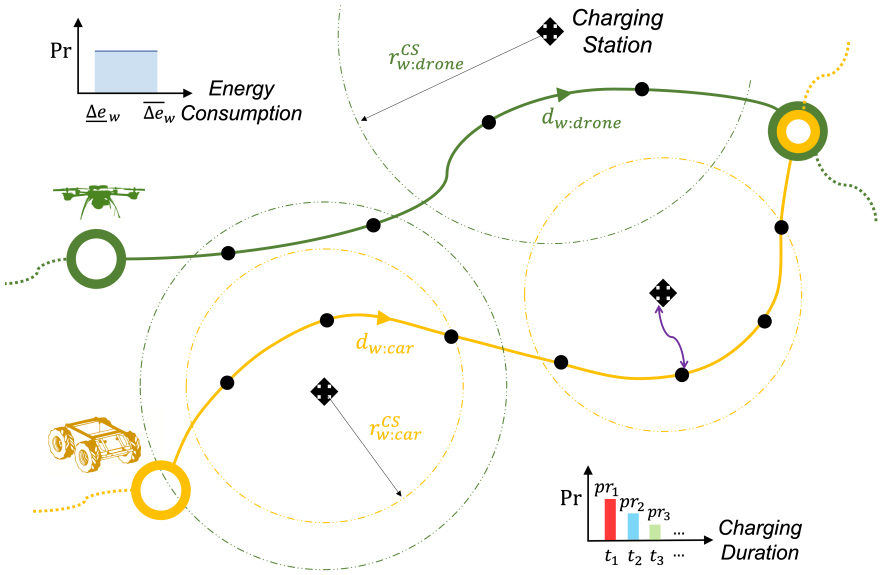


Figure 4.4: An illustration of heterogeneous agent paths with common objectives. Both the drone and car have paths such that when the battery is critical, they can reach a charging station within a given time by always staying within the radius of a station. Team-specific dynamics and stepwise energy consumption models determine the size of this radius. Energy consumption and charging duration models are stochastic, and whenever needed, agents get charged through the closest station, e.g., the purple route.

4.3 Energy-Aware Planning of Heterogeneous Multi-Agent Systems for Serving Cooperative Tasks

In this section, we extend the coordinated motion planning approach by considering a more realistic setting with a stochastic energy-consumption model for each agent and ensure recharge availability while satisfying the given tasks. The generated motion plans are energy-aware, requiring agents to be always in some vicinity of charging stations. Consideration of stochastic energy consumption in planning may yield missing the deadlines of STL specification. These expected amount of delays are then formally analyzed and quantified.

4.3.1 Coordination Problem with Energy Constraints

In addition to the Coordination Problem 4.2.1, here we also consider a certain charge capacity for each agent in team w that (stochastically) decreases with the agent motion.

Definition 4.3.1 (State of Charge)**.** *An agent i from team w at the time instant t has a charge capacity $e(t) \in [0, 100]$, $\forall t$ which evolves as $e_{w,i}(t+1) = \big(e_{w,i}(t) - \Delta e_{w,i}(t)\big)_+$ where $\Delta e \in \mathbb{R}_{\geq 0}$ is the energy consumption rate.*

Similar to the energy consumption models in [63, 64], we consider team-specific uniformly distributed consumption at each step as $\Delta e_{w,i} \sim U(\underline{\Delta e}_w, \overline{\Delta e}_w)$ where $\underline{\Delta e}_w, \overline{\Delta e}_w \in \mathbb{R}^+$ are the lower and upper limits of consumption for the respective team. We assume no energy consumption when agents are not moving. Given a set of stationary charging station positions (x_k^{CS}, y_k^{CS}) with k denoting the station id, the agents should visit the charging stations to get recharged to avoid running out of energy. Overall, we formulate an optimal control problem with an objective function (e.g., minimizing the total agent movement) subject to the swarm (motion and energy) dynamics and STL specification Φ.

Problem 4.3.1 (STL Satisfaction with Team-Level Tasks and Energy Constraints)**.** *Let a multi-agent system be comprised of n^{team} agent teams, each having n_w^{agent} identical agents, moving in an environment including a set of regions of interest P and a set of charging stations CS. Given a global STL specification Φ, find optimal agent velocities $\mathbf{v}^* = \big[\bigcup_w \bigcup_i \mathbf{v}_{w,i}(0) \ \dots \ \bigcup_w \bigcup_i \mathbf{v}_{w,i}(H-1)\big]$ for $i \in [1, n_w^{agent}]$ and $w \in [1, n^{team}]$ subject to the STL specification, agent dynamics, and energy constraints:*

$$
\begin{aligned}
\mathbf{v}^* = \arg\min \ & \sum_{t=0}^{H-1} \mathcal{J}\Big(\bigcup_w \bigcup_i \mathbf{v}_{w,i}(t)\Big) \\
s.t. \quad & \Big(\big[\cup_w \cup_i \mathbf{x}_{w,i}(0) \ \dots \ \cup_w \cup_i \mathbf{x}_{w,i}(H)\big], 0\Big) \models \Phi, \\
& \mathbf{x}_{w,i}(t+1) = \mathbf{x}_{w,i}(t) + \mathbf{v}_{w,i}(t)\Delta t, \\
& e_{w,i}(t) > 0, \ e_{w,i}(t+1) = \big(e_{w,i}(t) - \Delta e_{w,i}(t)\big)_+, \\
& \forall t \in [0, H-1], \ \forall i \in [1, n_w^{agent}], \ \forall w \in [1, n^{team}],
\end{aligned}
$$

where $\bigcup_w \bigcup_i \mathbf{x}_{w,i}$ is the aggregated agent paths, $H \geq hrz(\Phi)$ is the overall mission

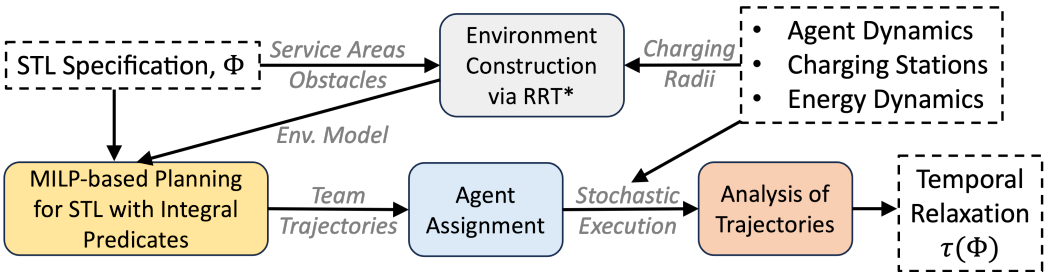


Figure 4.5: Proposed planning and execution scheme for energy-aware heterogeneous agents under STL specifications with integral predicates. Input and output to/from the system have dashed outlines.

horizon, $\mathcal{J}(\cdot)$ *is the running cost as a function of aggregated agent inputs,*

$$\Delta e_{w,i}(t) := \begin{cases} \sim U(\underline{\Delta e}_w, \overline{\Delta e}_w), & \mathbf{x}_{w,i}(t+1) \neq \mathbf{x}_{w,i}(t), \\ 0, & \textit{otherwise.} \end{cases}$$

is energy consumption, and $\big([\cup_w \cup_i \mathbf{x}_{w,i}(0) \ \ldots \ \cup_w \cup_i \mathbf{x}_{w,i}(H)], 0\big) \models \Phi$ *and* $e_{w,i}(t) > 0$ *enforce satisfaction of the STL specification without any agent being out of charge under energy consumption model in Def. 4.3.1.*

Note that this problem could be modified by including chance constraints and defined over the satisfaction probability. However, we require the completion of the mission without agents getting out of charge, and for this aim, we will relax the temporal requirements of Φ when necessary.

We propose a hierarchical and approximate solution to Problem 4.3.1 (see Fig. 4.5). First, a nominal plan is computed offline to ensure the satisfaction of the desired STL specifications at the team level. While we use a similar approach in Sec. 4.2 for efficiently generating paths, we also take into account here a set of charging stations and generate paths that are always within a certain distance of a charging station. This enables each agent to be sufficiently close to a charging station at any time during the mission. Once the team-level plans are generated, these plans are distributed to each agent. Agents follow their individual paths as well as a recharging policy, which might create some deviations from the original plan. Such deviations might delay task achievements. Hence, we also evaluate the expected delay and quantify a temporal relaxation of the specification Φ.

4.3.2 Team-Level Energy-Aware Planning

We aim to generate coordinated paths for limited-energy agents, which are heterogeneous in capability and dynamics, to satisfy cooperative tasks. Depending on agent capabilities, missions may require agents from different teams to collaborate and achieve preemptable tasks. Moreover, the paths need to be designed by considering the recharging needs.

The limited energy capacity of the agents with uncertain energy consumption models is another aspect to consider while designing energy-aware multi-agent trajectories. When the paths between service regions are long enough, the agent may need to get charged via charging stations en route.

We consider a set of regions of interest including the base and the service areas that need to be visited by the agents of the respective team. Moreover, there are additional regions associated with charging stations spread out over the environment, as illustrated in Fig. 4.4. Accordingly, we construct pairwise paths between these regions for each agent team via sampling-based algorithms under the constraint of always being within a certain distance from a charging station.

We model the partitioned environment for each team w as a graph with certain transition characteristics $\mathcal{G}_w = (\mathcal{V}_w, \mathcal{E}_w, d_w, \Delta e_w)$ where

- $\mathcal{V}_w := \mathcal{V}_w^{RoI} \cup \mathcal{V}_w^{CS}$ where $\mathcal{V}_w^{RoI}$ is the node set of nodes containing the regions of interest and the intermediate nodes, and $\mathcal{V}_w^{CS}$ represent the charging station nodes and intermediate nodes to reach them from $\mathcal{V}_w^{RoI}$,
- $\mathcal{E}_w := \mathcal{E}_w^{RoI} \cup \mathcal{E}_w^{CS}$ with $\mathcal{E}_w^{RoI} \subseteq \mathcal{V}_w^{RoI} \times \mathcal{V}_w^{RoI}$ and $\mathcal{E}_w^{CS} \subseteq \mathcal{V}_w^{CS} \times \mathcal{V}_w^{CS}$ denoting the set of transitions that can be traversed between two nodes at a time step,
- $d_w : \mathcal{E}_w \to \mathbb{R}^+$ is the cost representing the Euclidean distance between two nodes,
- $\Delta e_w : \mathcal{E}_w \to U(\underline{\Delta e}_w, \overline{\Delta e}_w)$ is the stochastically modeled state of charge decrease (energy consumption) rate, where $\underline{\Delta e}_w, \overline{\Delta e}_w \in \mathbb{R}^+$ are the limits of consumption.

Note that $\mathcal{V}_w^{RoI} \cap \mathcal{V}_w^{CS} \neq \emptyset$ since a path from/to a charging station contains node(s) in $\mathcal{V}_w^{RoI}$.

We can define heterogeneous dynamics for single integrator agents by setting different transition costs for agents moving with different speeds, similar to the previous

section. To illustrate, the step size of the drone is larger than that of the car in Fig. 4.4 as drones can travel longer distances within a time step.

We use again a sampling-based algorithm, RRT* [62], accounting for team dynamics and also recharging considerations. We first implement nominal planning that is constrained to keep charging available all the time without considering the actual recharging maneuvers as the energy consumption is stochastic. To this end, the nominal environment $\mathcal{G}_w^{RoI} = (\mathcal{V}_w^{RoI}, A_w^{RoI}, d_w)$ is generated such that at every instant, agents are within charging radius of a station.

Definition 4.3.2 (Charging Radius)**.** *Given a set of stationary charging station positions (x_k^{CS}, y_k^{CS}), the h-hop charging radius of the charging station k for the agent team w is defined by $r_{k,w}^{CS} = d_w h$.*

We constrain the value of $h \in \mathbb{Z}^+$ to satisfy $(2h+1)\overline{\Delta e}_w \leq 100$, $\forall w$, where $\overline{\Delta e}_w$ is the worst-case stepwise energy consumption for team w. This constraint ensures an agent's progress (at least for one transition) on its nominal path after getting recharged, hence $2h+1$ transitions are considered instead of $2h$. To enforce the charging radius constraint on the generation of the nominal environment $\mathcal{G}_w^{RoI}$, we apply the following modification on RRT* [62]: *"For each $v \in \mathcal{V}_w^{RoI}$, there must be a charging station k such that $\sqrt{(v_x - x_k^{CS})^2 + (v_y - y_k^{CS})^2} \leq r_{k,w}^{CS}$ where $[v_x, v_y] \in \mathbb{R}^2$ is the position of the node v."*

Intuitively, we want all agents to be able to reach a charging station in at most h-hop when necessary. The following assumption is required to ensure that the agent path is always within the charging radius of a charging station.

Assumption 4.3.1 (Charging Station Locations)**.** *Consider an ordered sequence of charging stations $CS_{ij} = \{CS_1, CS_2, \ldots\}$ with $(x_k, y_k) \in \mathbb{R}^2$ denoting all points within the charging radius of the station k, i.e., $\sqrt{(x_k - x_k^{CS})^2 + (y_k - y_k^{CS})^2} \leq r_{k,w}^{CS}$. For any two regions of interest of any team w denoted by the nodes v_i and v_j, there exist such a sequence of stations CS_{ij} with $\min(\sqrt{(x_{k-1}-x_k)^2+(y_{k-1}-y_k)^2}) \leq d_w$ for $k \in [2, |CS_{ij}|]$, $(x_1^{CS}, y_1^{CS}) = (v_{i_x}, v_{i_y})$, and $(x_{|CS_{ij}|}^{CS}, y_{|CS_{ij}|}^{CS}) = (v_{j_x}, v_{j_y})$.*

In other words, Assumption 1 requires that the environment should have a sufficient number of charging stations covering all the nodes. In that case, an agent cannot be in

a node from which a charging station is not reachable in at most h-hop (i.e., the agent can move from one circle to another at a step as illustrated in Fig. 4.4.

For a given set of charging station locations, we find the shortest pairwise paths in between the regions of interest via the RRT* method modified by charging considerations. Intuitively, the higher d_w or h, the longer the charging radius r_w^{CS} as depicted in Fig. 4.4; therefore, fewer stations are needed. That said, we do not choose the locations of charging stations, but we are provided with them under Assumption 4.3.1.

We can formulate the optimization problem with mixed-integer constraints to minimize agent movement while satisfying the desired STL specification, similar to (4.5). Moreover, since an unknown number of charging maneuvers will be required for each agent during the mission, it is favorable to keep delays in STL time intervals minimal by satisfying the tasks as soon as possible. To this end, we define the running cost via the sum-norm of the agent flow as $\mathcal{J}\big(\mathrm{u}_1(t),\dots,\mathrm{u}_{n^{team}}(t)\big) = \sum_{w=1}^{n^{team}} t\,\big\|\mathrm{u}_w(t)\big\|_1$ and penalize later transitions more to get the agents to move as early as possible.

As the service propositions and progress function are linear, the optimization problem takes the form of a mixed-integer linear program (MILP) as follows:

$$\begin{aligned} \mathrm{u}^* = &\arg\min \sum_{t=0}^{H-1}\sum_{w=1}^{n^{team}} t\,\big\|\mathrm{u}_w(t)\big\|_1 \\ s.t.\quad &(4.1),\ (4.2\mathrm{a}),\ (4.2\mathrm{b}), \\ &N_w(0) = N_{w_0},\ \forall w \in \{1,\dots,n^{team}\}, \\ &z_0^{\Phi} = 1. \end{aligned} \tag{4.7}$$

The solution of (4.7) gives the nominal plan for the multi-agent system satisfying an STL specification with preemptable service requirements. However, due to the stochastic energy consumption model, we do not know how many agents will need charging and when or how often these charging instances will occur. Next, we will investigate the probabilistic models to estimate these values and implement the given plan with a recharging policy for each agent.

4.3.3 Agent-Level Implementation under Charging Constraints

After obtaining the team-level plans with swarm flow throughout the environment $\mathcal{G}$, now we have to i) assign individual agents to perform flow requirements and ii) implement the transitions subject to energy consumption and charging considerations, i.e., get them charged whenever needed.

Agent Assignment

Given the agent flow $\mathrm{u}_w(t)$ with $u_{w_{i,j}}(t)$ denoting the number of agents required to move from node v_i to node v_j, let us denote the set of team w agents inside the node v_i by Σ_{w,v_i} and the position vector of these agents by $\beta_{w,v_i} = [\mathbf{x}_{w,1} \ \cdots \ \mathbf{x}_{w,|\Sigma_{w,v_i}|}]^T \in \mathbb{R}^{2|\Sigma_{w,v_i}|}$, we assign the agents via:

$$\beta^* = \arg \min_{\beta^* \subseteq \beta_{w,v_i}} \sum_{k=1}^{u_{w_{i,j}}(t)} (v_{j_x} - \beta^*_x(k))^2 + (v_{j_y} - \beta^*_y(k))^2.$$

The indices of β^* determine which agents inside the set Σ_{w,v_i} need to move to the node v_j.

Online Energy-Aware Decision-making

We consider a given nominal plan as the solution of (4.7), which includes the flow of agents through environment $\mathcal{G}_w$ for $w \in [1, n^{team}]$. By construction, the environment enables each agent to reach a charging station in at most h-hop (see Sec. 4.3.2). Given energy consumption distribution $\Delta e_{w,i} \sim U(\underline{\Delta e}_w, \overline{\Delta e}_w)$ we define the battery threshold e^* as follows.

Definition 4.3.3 (Battery Threshold)**.** *Given a user-defined probability of γ such that at any instant t, an agent can reach the next intended node and then a charging station in $h+1$-hop with a probability of $\gamma \in [0,1]$. Then the battery threshold e^* is defined obeying to* $\Pr\left(\sum_{\tau=1}^{h+1} \Delta e_w(\tau) \leq e^*\right) = \gamma$.

In other words, at each time t, agents have to check if moving to the next node would violate the reachability to a charging station in h-hop. Therefore an $h+1$-hop motion is assessed and used in the calculation of the battery threshold e^*. When the battery

level drops below the threshold, a 'charge-and-continue' route is calculated online with RRT* between the closest charging station and the current node. This planning is implemented over a much smaller environment similar to [65] (purple charging route in Fig. 4.4).

Note that $\gamma = 1$ implies that even if $\sum_{\tau=1}^{h+1} \Delta e_w(\tau) = (h+1)\overline{\Delta e}_w$, i.e., the worst energy consumption possible in $h+1$-hop occurs, the agent will still be able to reach the charging station. This also guarantees the recharging from the current time t whenever needed by Def. 4.3.2 and the fact that $e(t-1) \geq e^*$. That said, we keep $\gamma \in [0,1]$ user-defined and allow a risk-aware planning process.

Remark 4.3.1. *Summation of n uniformly distributed random values $U_i \sim U(0,1)$ yields Irwin–Hall distribution that can be approximated via $\sum_{i=1}^{n} U_k \sim N\big(\mu = \frac{n}{2}, \sigma^2 = \frac{n}{12}\big)$.*

Let an agent be fully charged, i.e., $e_0 = 100$, then the probability of the agent's state of charge dropping below the threshold and becoming critical after moving for n steps is

$$\Pr(e(n) < e^*) = 1 - \Pr\Big(\sum_{i=1}^{n} \Delta e(i) \leq 100 - e^*\Big),$$

via the cumulative distribution function,

$$\begin{aligned} F_{\sum_{i=1}^{n} \Delta e(i)}\big(100 - e^*\big) &= \Pr\Big(\sum_{i=1}^{n} \Delta e(i) \leq 100 - e^*\Big) \\ &= \frac{1}{\sqrt{\pi n/6}} \int_{-\infty}^{A} e^{\left(-\frac{(t-\frac{n}{2})^2}{n/6}\right)} dt, \end{aligned}$$

where $A = \frac{100 - e^* - n\,\underline{\Delta e}}{\overline{\Delta e} - \underline{\Delta e}}$. Moreover, we can find the battery threshold based on h to assess the reachability of charging stations after moving to the next node in the plan and the user-defined probability γ by the quantile function as:

$$e^*_w = (h+1)\overline{\Delta e}_w + (\overline{\Delta e}_w - \underline{\Delta e}_w) F^{-1}_{\sum_{i=1}^{h+1} \Delta e_w(i)}(\gamma), \ \forall w \in W.$$

Definition 4.3.4 (Charging Duration)**.** *Whenever an agent reaches a charging station node, it is recharged back to $e = 100$ in t_c steps with a probability of pr_{t_c}. Let the alternative charging durations be distributed as $CD = \{t_{c,1}, t_{c,2}, \ldots\}$ with the probabilities $\{pr_{t_{c,1}}, pr_{t_{c,2}}, \ldots\}$, then the expected duration in each station is $E[t_c] = \sum_{i=1}^{|CD|} t_{c,i} pr_{t_{c,i}}$.*

Temporal Relaxation of STL under the Proposed Solution

Due to stochastic energy consumption, agents can deviate from their nominal paths to get recharged. Regarding the policy discussed in Sec. 4.3.3, which includes returning to the node in $\mathcal{V}_w^{RoI}$ from which the agent leaves to get recharged, we define and quantify the expected temporal relaxation/violation in this section.

Given a path between two regions of interest, namely $v_i, v_j \in \mathcal{V}_w^{RoI}$, i.e., $path_{ij} = \{v_i, \ldots, v_j\}$, we denote the number of intermediate nodes as $n_{ij}^{int} = |path_{ij}| - 2$. Depending on the h-hop charging station reachability, the possible number of charging instances on $path_{ij}$ can be represented as $n_{ij}^{charge} \in \{1, \ldots, n_{ij}^{int} - h\}$. Then, we can find the probability of the number of charging instances while traveling between v_i and v_j as:

$$\Pr(n_{ij}^{charge} = l) = \sum_{m=1}^{\binom{n_{ij}^{int}-h}{l}} \Pr\Big(\bigwedge_{r \in L_m^l} e(r) < e^* \mid \bigwedge_{r \in [1, n_{ij}^{int}-h] \setminus L_m^l} e(r) \geq e^* \Big),$$

where L^l is the set of combinations of l nodes, i.e., $L^l = \{L \subseteq \{1, \ldots, n_{ij}^{int} - h\} \mid |L| = l\}$, and L_m^l is the m^{th} element of L^l. Moreover, the expected delay τ_{ij} while traveling from node v_i to node v_j would then be found as:

$$E[\tau_{ij}] = \sum_{l=1}^{n_{ij}^{int}-h} \Pr\big(n_{ij}^{charge} = l\big) \times l \times \big(2h + E(t_c)\big).$$

Definition 4.3.5 (Temporal Relaxation of STL Specification)**.** *Suppose an STL specification,*

$$\Phi := \phi_1 \wedge \ldots \wedge \phi_{n^{task}}, \tag{4.8}$$

where each task ϕ_i is associated with a region of interest $p_i \in P$. Moreover, let the nominal path of agent i be comprised of multiple ordered regions of interest as $path_i = \{p_1, p_2, \ldots\}$. We will construct the expected temporal relaxation $\tau(\Phi)$ starting from the individual agent paths as follows:

- *Calculate the delays in reaching each region of interest in $path_i$ that is supposed to be serviced by each agent i in each team w:*

$$\tau_{w,i}(p_j) = \tau_{w,i}(p_1) + \sum_{l=2}^{j} E[\tau_{path_i(l-1), path_i(l)}],$$
$$\text{for } j \in [1, |path_i|],\ i \in [1, n_w^{agent}],\ w \in [1, n^{team}].$$

- *Determine the node-specific relaxations for the whole system and overall temporal relaxation of Φ:*

$$\tau_{temp}(\Phi) := \max_{j\in[1,n^{task}]} \max_{w\in[1,n^{team}]} \max_{i\in[1,n_w^{agent}]} \tau_{w,i}(p_j). \tag{4.9}$$

The relaxation value above is not final, since the delay in the nominal agent paths does not necessarily imply the violation of the task deadlines. This is because the nominal plan can include teams satisfying tasks way before the given deadline. Hence, we find how early teams complete giving services in the nominal plan and reflect this time budget in calculating the temporal relaxation.

Let $\{t_{d,1}, \ldots, t_{d,n^{task}}\}$ be the task deadlines in (4.8). Moreover, we denote the completion step of the service specified in ϕ_j by t_j^*. Therefore, task-specific time budgets can be reflected in the formulation of temporal relaxation in (4.9) as:

$$\tau(\Phi) := \max_{j\in[1,n^{task}]} \Big(\big(\max_{w\in[1,n^{team}]} \max_{i\in[1,n_w^{agent}]} \tau_{w,i}(v_j) \big) - \big(t_{d,j} - t_j^* \big) \Big).$$

When periodic tasks are concerned, there may be multiple satisfaction instances, i.e., completion times, for the same single task in (4.8). We treat all such instances like separate tasks, as the deadline and completion times are still known a priori via the nominal plan. Furthermore, $\tau(\Phi)$ constitutes an "expected upper bound" on temporal relaxation as all charging maneuvers are assumed to have exactly h-hops as the worst-case scenario, whereas they can be less in reality.

4.3.4 Case Study: A Cooperative Energy-Aware Multi-Agent System

We consider a multi-agent system with three agent teams whose properties are shown in Table 4.3. We refer to heterogeneous agents by their shape and color. The pairwise team-specific paths between respective regions of interest for each agent team are found via RRT* sampling-based algorithm [62] with modifications to avoid team-specific obstacles and ensure availability of charging stations in at most h-hop on a 3×1.4 m environment represented in Fig. 4.6.

Mission specifications for the multi-agent system are summarized as follows: i) Bottom green circular region has to be imaged 30 times in total within $[10, 30]$ by the agents equipped with a camera, ii) 45 liters of pesticide need to be sprayed on the blue circular

Table 4.3: Properties of the three agent teams with energy constraints.

Agent Team (#Agents)	Capability (per time step)	Regions	#Obstacles	Speed	Energy Cons., $\Delta e_w \sim U(\underline{\Delta e}_w, \overline{\Delta e}_w)$
Triangle/Magenta (2)	Measure temperature (×2) Spray pesticide (×2)	4	3	0.10 m/s	$\sim U[6, 10]$
Diamond/Orange (2)	Take photo (×1) Measure temperature (×1)	4	2	0.15 m/s	$\sim U[10, 14]$
Circle/Green (2)	Take photo (×1) Spray pesticide(×1)	4	2	0.15 m/s	$\sim U[10, 14]$

region in the middle between $t = 20$ and $t = 40$, iii) 30 temperature readings should be made in the yellow region on top within $[10, 30]$. These specifications are expressed using the integral predicates as:

$$\begin{aligned}\Phi := \mu^i_{[10,30]}\{Target_{bottom/green}, Take\,photo, 30\} \\ \wedge\, \mu^i_{[20,40]}\{Target_{middle/blue}, Spray\,pesticide, 45\} \\ \wedge\, \mu^i_{[10,30]}\{Target_{top/yellow}, Measure\,temperature, 30\}.\end{aligned} \tag{4.10}$$

Simulations are implemented for an unknown length of mission horizon that is shaped according to stochastic charging requirements throughout the mission with $\Delta t = 1$. The distribution of agents moving over the environment is represented in Fig. 4.6. The specification in (4.10) is accomplished with some delays due to limited energy capacity. We illustrate that agents from different teams contribute to the satisfaction of preemptable tasks in Fig. 4.6. For instance, a diamond agent joins a circle agent to service the green area at $t = 26$. The same agent moves to the yellow region at $t = 39$ to collectively service with the other diamond and triangle agents. Throughout the mission, the agents are also subject to charging maneuvers and calculate their paths online (shown in black) to the closest station whenever needed.

For the scenario in Fig. 4.6, the charging station availability parameter h is 3 with the charging duration and availability probabilities of $\Pr(t_c = \{1, 2, 3\}) = \{0.8, 0.15, 0.05\}$ and $\gamma = 0.9$. We also investigate the impact of this user-defined probability threshold γ on the temporal relaxation of STL specification. Results are presented in Fig. 4.7. We set different γ levels and inspect both actual temporal relaxation and its expected value

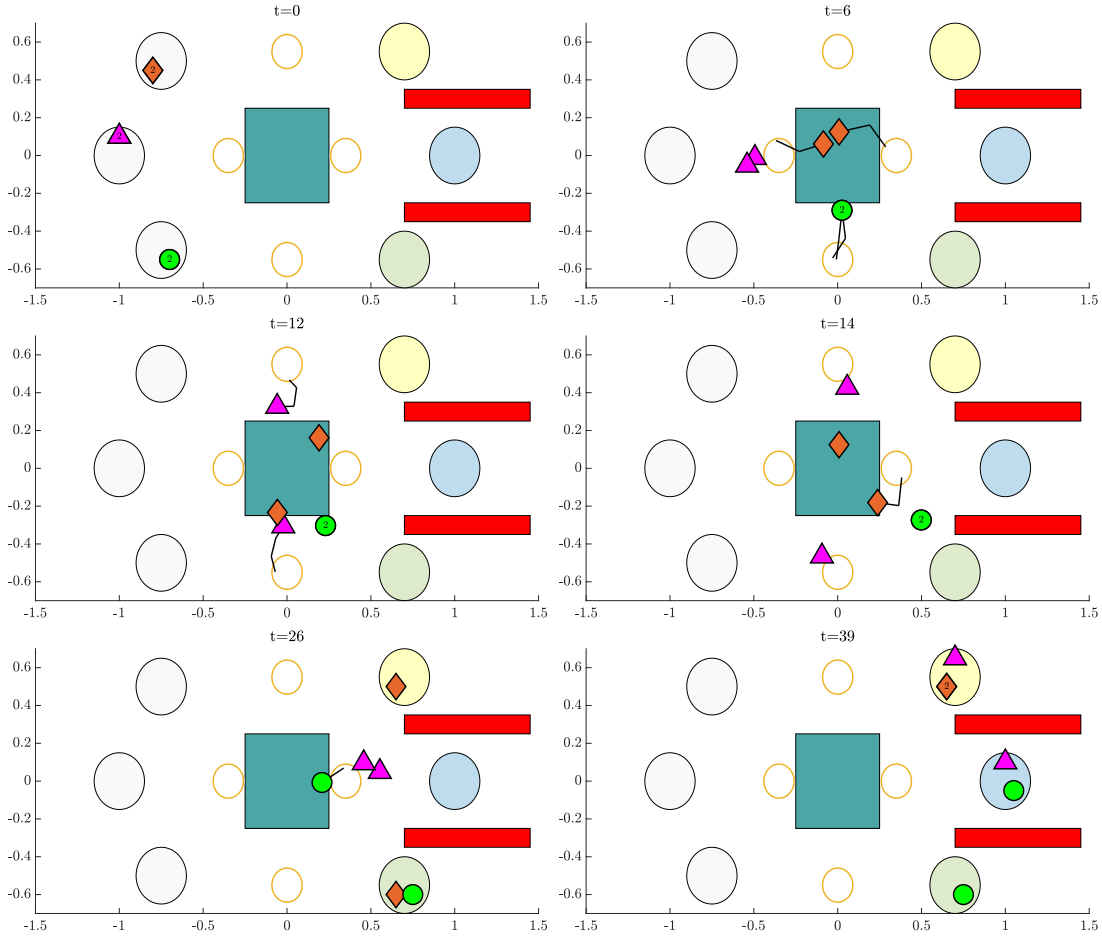


Figure 4.6: Team plans of the triangle (magenta), diamond (orange), and circle (green) agents at selected time instants (red rectangular regions: obstacles for all agents, blue rectangular region: obstacle for the triangle agents only). The values attached to markers (if any) indicate the number of agents from the same team at that node. Gray circles are the bases of teams. Target regions are colored according to (4.10). The orange circles illustrate charging stations. Some of the snapshots depict newly generated black paths between charging stations and individual agents when their states of charge get critical.

obtained as discussed in Sec. 4.3.3. As γ decreases, holding less charge becomes more tolerable due to decreased battery level threshold e^*. Consequently, agents take risk and go for charging less frequently, which results in smaller temporal relaxation. Moreover, the expected and actual ($n = 10$) temporal relaxations are close to each other, verifying the formal analysis made prior to the actual mission implementation in Sec. 4.3.3.

Since the plan is generated using a flow control problem, increasing the number of agents has no significant adverse effect on the solution times. In addition to 2+2+2 setting in Table 4.3 which takes $\sim 230\,s$ to generate the nominal plan, we tried 5+6+7 ($\sim 8\,s$) and 10+10+10 ($\sim 12\,s$) number of agents as well for the same specification in (4.10). These results imply that increasing the number of agents can even facilitate the solution of the problem since including more agents provides a larger solution space for the specifications with cumulative properties. On the other hand, when the nominal plan is executed subject to probabilistic charging requirements, we obtain the mean computation times of $0.11\,s$, $0.15\,s$, and $0.18\,s$, respectively, for these three multi-agent settings ($n = 100$), suggesting a linear increase.

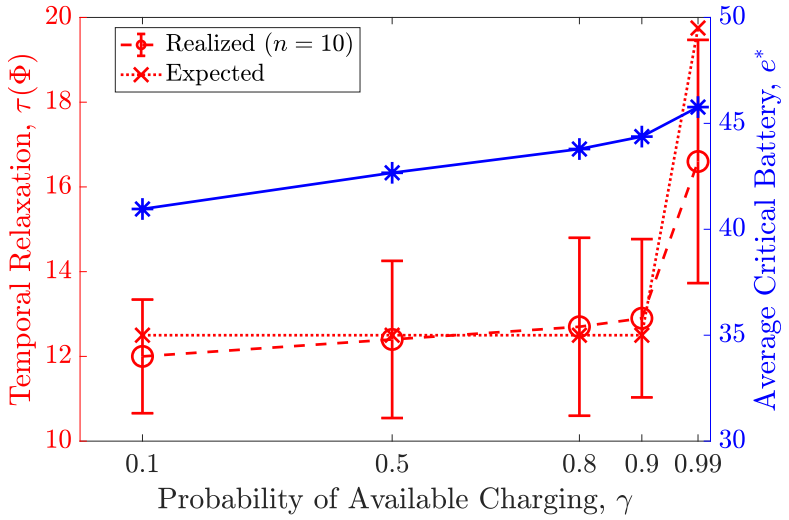


Figure 4.7: Results of how threshold γ on the probability of available charging affects the actual and expected temporal relaxation values due to changed critical battery level threshold. The actual (realized) mission implementation values are obtained by generating $n = 10$ samples.

A scenario similar to the one in Fig. 4.6 satisfying (4.10) is also conducted with a more realistic dynamical system using the Robotarium platform [66] depicted in Fig. 4.8.

4.4 Discussion

This chapter shows how integral predicates for STL specifications can be used in multi-agent systems to encode the cumulative progress in tasks. In the case of heterogeneous

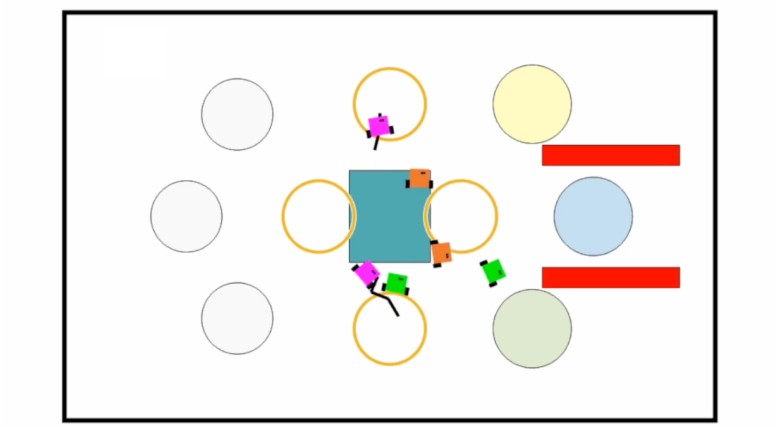


Figure 4.8: Snapshot from the Robotarium simulation with six ground robots.

agents, such a predicate can be used to define the progress within a specific time interval that can be asynchronously achieved by multiple agents during different time steps. We also show that the integral predicates can be encoded as mixed-integer linear constraints. Optimal trajectories are then obtained by solving a MILP problem. We show how the proposed predicate can be used in the planning of the multi-agent trajectories more richly and expressively compared to the conventional STL.

Moreover, an energy-aware planning approach is introduced using the proposed planning framework for heterogeneous agents. This involves a hierarchical solution to energy-aware planning and implementation at team and agent levels subject to stochastic energy consumption and charging. We formally quantify the expected amount of worst-case delays and relate them to the temporal relaxation of the STL specification. However, this quantification is done as a post-processing. The next chapter addresses the formulation of a formal metric to quantify temporal relaxation in STL specifications that are either initially infeasible or become infeasible later on during the mission. This new metric can be used in optimization routines for planning and control.

Chapter 5

Resilient Motion Planning under STL Specifications

This chapter addresses the problem of resilient motion planning for robots operating under STL specifications in dynamic environments. In such settings, unforeseen events—such as the emergence of dynamic obstacles—can render the original STL specification infeasible. To address this, we propose a reactive framework that enables local corrections or global replanning of the robot's trajectory in response to these unexpected occurrences. We introduce a metric that can quantify the temporal relaxation of Signal Temporal Logic (STL) specifications and facilitate resilient control synthesis in the face of infeasibilities. When the original STL specification becomes unsatisfiable, our framework involves minimally relaxing it by extending/shrinking time windows or removing some tasks as per user preferences. This strategy is designed to prevent arbitrarily long delays in mission completion and facilitate the satisfaction of the mission with minimal temporal relaxation. We present theoretical results supporting our framework and demonstrate its effectiveness through high-fidelity simulations, along with benchmark analysis.

The chapter is organized as follows: Section 5.1 introduces the relevant literature on STL control synthesis with temporal aspects and reactive planning under temporal logic specifications. A family of STL specifications that can be tackled with the proposed approach is identified in Sec. 5.2. Next, we introduce a metric that can formally quantify

the temporal relaxation of STL specifications in Sec. 5.3. A comparison between the proposed metric and the standard STL metrics is presented in Sec. 5.5. Section 5.4 involves an STL control synthesis framework minimizing the proposed metric. An online monitoring and reactive planning algorithm that continuously evaluates the robot's trajectory and dynamically adjusts it as well as the STL specification to minimize temporal relaxations is addressed in Sec. 5.6.

5.1 Introduction and Literature Review

As discussed in Chapter 2, there are two main STL robustness metrics, i.e., space and time robustness which capture the effect of shifting the signal in space and time on the satisfaction, respectively [26]. While most of the studies in the literature propose solution methods to optimize space robustness or a variant of it [14,16,27,41,42,47–49], there are also some works emphasizing the benefits of optimizing time robustness [67–70]. Moreover, allowing for negative robustness and maximizing it may imply minimal violation of an STL specification. Control synthesis with space-related metrics can allow for spatial relaxations (e.g., if Region A cannot be reached within 10 minutes, get as close as possible within that time frame). However, the time-related metrics may not provide a notion of temporal relaxation, especially when the failure becomes inevitable due to unexpected environmental changes.

In real-world applications, autonomous systems operate under various disturbances (e.g., internal, external, human-triggered), such as the sudden appearance of obstacles or human intervention, which can cause deviations from nominal plans. In more severe cases, the original mission may become unsatisfiable (e.g., a robot tasked with reaching region A in 10 minutes may be delayed due to the updated plan to navigate around unexpected obstacles). Moreover, when multiple tasks are involved, a delayed arrival at one region can cause cascading delays in subsequent tasks. In such cases, a resilient control synthesis requires to search for trajectories resulting in minimal violations of the original specification. One common way of achieving this is by relaxing the spatial requirements of the specification (e.g., [30]), such as satisfying $x \geq 4.7$ instead of $x \geq 5$. On the other hand, relaxing the time bounds by strictly enforcing the thresholds over signal values is not investigated much in the STL literature, although applications such

as manipulating objects in fixed regions would benefit from it. Such a notion of temporal relaxation has been introduced for Time Window Temporal Logic (TWTL) in [71], where an automata-theoretic approach is proposed to shrink or extend the corresponding time intervals of the tasks when needed.

Time robustness [26] in STL control synthesis is considered in [67], where the goal is to maximize the time robustness along with the space robustness for a limited family of STL specifications; however, the infeasibility of STL specifications is not discussed. Moreover, in [68] and [70], mixed-integer encoding of the right-time robustness and a left-right combined time robustness metric are introduced, respectively. These metrics, however, may not facilitate a reactive response needed to operate in dynamic environments due to focusing primarily on critical trajectory segments that might be in the past and dominate the overall performance. Similarly, time robustness under time shifts in the stochastic signals is assessed along with a risk measure of STL failure in [69]. While maximizing time robustness may lead to satisfaction of the STL specification even when the signal is shifted along the time (e.g., delay in the signal), it may not necessarily result in minimal temporal relaxation of the STL specifications in the presence of violations. This is because the quantity of time robustness metric is dominated by the critical violations (i.e., use of min/max functions), which makes it hard to differentiate individual task relaxations. Furthermore, maximizing time robustness, especially in the violation cases [67, 68], requires a significant computational effort, which is also illustrated later on in the following sections.

STL specifications can be infeasible by nature. In such scenarios, the specifications can be repaired [72] or partially satisfied [73]. In [74, 75] partial satisfaction is achieved through a two-fold optimization framework. In [72] repair of infeasible STL specifications is addressed, where an iterative algorithm is proposed to repair infeasible STL specifications by relaxing thresholds over signal values, penalizing relaxations, and determining possible interval modifications based on the relaxed thresholds. However, this work is not equivalent to minimizing temporal relaxation and its iterative nature demands more computational effort. Moreover, a notion of temporal relaxation is investigated in [21], where an algorithm is proposed to plan trajectories that satisfy iteratively changing STL specifications (i.e., shifting STL). In particular, the original STL specification is considered but its time intervals are from the current time instant

to the end of the time interval. However, this work is limited to safety and persistence behaviors. In general, these studies provide offline solutions and cannot adapt to unexpected changes/disturbances in a dynamic environment. Therefore, recovery from infeasibility due to environmental changes, disturbances, or actuator limits and the impact of such failures on the realized specifications remain a challenge.

For robots to fulfill temporal logic specifications in dynamic environments, it is essential that they can react to unexpected events by adjusting their trajectories in real time. These events might include potential collisions [76–78], deviations caused by external disturbances [67], or newly defined specifications [79]. Replanning the entire trajectory after each such event can be computationally expensive, particularly in cluttered environments where frequent replanning leads to increased computational overhead [78]. To address this, local trajectory corrections are proposed as a more efficient alternative for minor deviations, allowing the robot to continue its mission without needing a complete replanning.

Several studies explore reactive motion planning under temporal logic constraints. For example, deviation from STL-satisfying trajectories is addressed in [67] via local Model Predictive Control (MPC) corrections, which drives the robot to return to its nominal trajectory after disturbances. Efficient local corrections in dynamic environments are tackled in [76] and [77], where the robot adapts to obstacles detected in real time.

Although local corrections offer some benefits, they may sometimes be inadequate or impractical, necessitating full replanning in certain cases. Approaches like [80] improve the efficiency of replanning by reconstructing the state graph only partially to accommodate changes in the environment. Similarly, [77] employs mixed-integer programming to handle failures by only considering the affected/violated constraints, reducing the overall computational burden. Our work similarly proposes efficient reactive planning, but we can tackle infeasible tasks by minimally violating specifications.

Overall, the lack of online synthesis methods that can react to unexpected changes in the environment and even allow for minimal violations of the original STL specification (by strategically modifying the time windows or removing tasks) motivates us to address the challenges of resilient motion planning under minimally relaxed STL specifications.

5.2 A Family of STL Specifications for Temporal Relaxation

We consider the following STL fragment to express the desired system behaviors:

$$\begin{aligned}\Phi &::= \phi \mid \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid F_{[a,b]}\Phi \mid G_{[a,b]}\Phi,\\ \phi &::= F_{[a,b]}\varphi \mid G_{[a,b]}\varphi,\\ \varphi &::= \mu \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2,\end{aligned} \tag{5.1}$$

where Φ, ϕ, φ are STL specifications. Although the syntax in (5.1) is expressive enough to specify various rich tasks, it does not allow for conjunction/disjunction of the temporal operators with bare predicates as in $\Phi = F_{[a,b]}(\mu_1 \wedge G_{[c,d]}\mu_2)$.

Example 5.2.1. *Consider two signals $\mathbf{x}'$ and $\mathbf{x}''$ starting from $t = 0$ and illustrated in Fig. 5.1. Suppose that the specification is $\Phi_1 = G_{[15,60]}\mathbf{x} \geq h_1$ meaning that the signal has to satisfy $\mathbf{x}(t) \geq h_1$ for all $t \in [15, 60]$. While both signals violate Φ_1, $\mathbf{x}'$ (blue) stays in the desired region longer than $\mathbf{x}''$ (red). However, the (right) time robustness notion cannot differentiate this due to using* min *function in its computation as in* (2.3)*. Specifically, $\theta^+(\mathbf{x}', \Phi_1, 0) = \theta^+(\mathbf{x}'', \Phi_1, 0) = \min_{t\in[15,60]} \theta^+(\mathbf{x}, \mathbf{x}(t) \geq h_1, t)$, hence both signals have the same time robustness that is negative due to violation, and its length is illustrated by the purple bar in Fig. 5.1.*

Now, suppose the specification is updated as $\Phi_2 = \Phi_1 \wedge F_{[75,120]}\mathbf{x} \leq h_2$ meaning that the signal needs to satisfy $\mathbf{x}(t) \leq h_2$ for some time $t \in [75, 120]$ in addition to satisfying Φ_1. Despite the blue signal's satisfaction of the second task, the violation of Φ_1 dominates the computation of time robustness of Φ_2. Furthermore, the signal $\mathbf{x}''$ slightly violates the second task, but the initial violation still dominates. As a result, the violation in the purple area indifferently determines the time robustness score of both signals and specifications with $\theta^+(\mathbf{x}', \Phi_1, 0) = \theta^+(\mathbf{x}'', \Phi_1, 0) = \theta^+(\mathbf{x}', \Phi_2, 0) = \theta^+(\mathbf{x}'', \Phi_2, 0)$. Similar examples are straightforward to illustrate for the left time robustness as well.

Note that the standard space robustness in (2.3) may be insufficient to differentiate signal behaviors as well. For instance, $\rho(\mathbf{x}', \Phi_1, 0) = \rho(\mathbf{x}'', \Phi_1, 0) = -\rho^*$. However, this issue is addressed by several studies via modified space robustness metrics and extended STL syntax (e.g., [16,41,42,47–49]). Nonetheless, such metrics do not assess satisfaction

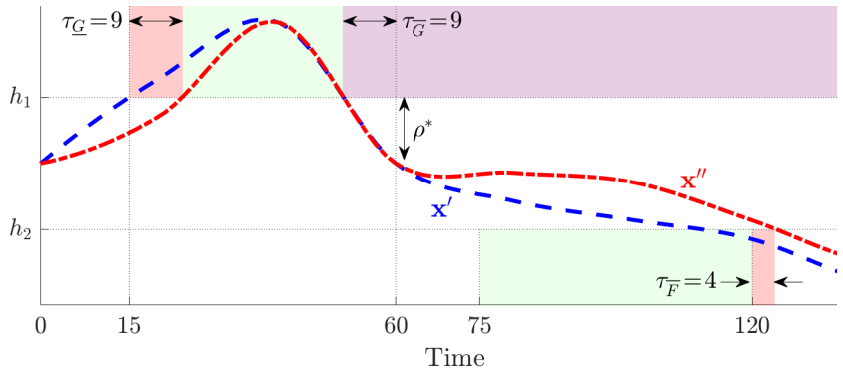


Figure 5.1: Two signals that violate the given STL specifications by different amounts. Green regions are the targets the system must reside in to achieve the specifications. The areas of violation are denoted by the color of the respective signal, while the purple region depicts violation by both signals.

beyond the original time intervals, which motivate us to introduce a temporal relaxation metric for STL specifications.

One may also consider using average time robustness values over the intervals instead of min/max functions similar to existing space robustness modifications. However, this may not result in desired relaxed notions. For instance, let $\Phi_3 = G_{[75,120]}\mathbf{x} \geq h_2$. As depicted in Fig. 5.1, $\mathbf{x}'$ leaves the $\mathbf{x} \geq h_2$ region before the deadline of $t = 120$. This yields the time robustness values between the violation instant and $t = 120$ to become very small (negative) since there is no satisfaction instant remaining until the end of the mission. Hence, arbitrarily large mission horizons yield violated time robustness values to be arbitrarily small. In this regard, averaging them with satisfactory positive small values does not provide a good understanding of the signal behavior.

The spatial relaxation of STL specifications is generally possible by allowing the system to minimally violate its spatial requirements (i.e., maximizing the space robustness while letting it be negative). For example, if the specification is entering a desired region, the spatial relaxation will be approaching the region as close as possible given the time interval. However, spatial relaxation may not be feasible for some missions, e.g., manipulating an object in a fixed region. Therefore, extending or shrinking the time intervals to allow for late or early satisfaction can be practical in some scenarios. However, the temporal relaxation of STL is not broadly investigated, and the standard time robustness cannot differentiate partial satisfactions as illustrated in Example 5.2.1.

5.3 Temporal Relaxation Metric

Suppose that a task ϕ in the STL specification Φ (as per the syntax in (5.1)) is defined over a time interval of $I = [a, b]$. For every such task, we have the temporally relaxed versions $\phi^\tau = F_{[a-\tau_{\underline{F}}, b+\tau_{\overline{F}}]}\varphi$ and $\phi^\tau = G_{[a+\tau_{\underline{G}}, b-\tau_{\overline{G}}]}\varphi$ where $\tau_{\underline{F}}, \tau_{\overline{F}}, \tau_{\underline{G}}, \tau_{\overline{G}} \in \mathbb{Z}_{\geq 0}$ are temporal relaxation parameters. That is, the system may *eventually* satisfy φ within a longer time interval and *always* satisfy it within a shorter interval, both of which imply the relaxation of the original requirement. Moreover, when multiple temporal operators are nested, temporal relaxation is enabled for the innermost ones. For example, if the original specification is $\Phi = F_{[c,d]}G_{[a,b]}\varphi$, then the relaxed version of it is considered as $\Phi^\tau = F_{[c,d]}G_{[a+\tau_{\underline{G}}, b-\tau_{\overline{G}}]}\varphi$.

Our approach will result in the satisfaction of the predicates and/or their combinations only within the original *globally* interval I_G (potentially within an interval shorter than the original) and remove the task otherwise. On the other hand, the interval of the *finally* operator I_F is extended, and when not bounded, this extension may yield excessive relaxations, e.g., satisfying μ at $t = 100$ where the specification is $\phi = F_{[5,10]}\mu$. For this reason, our formulation allows the user to specify acceptable bounds to relax the original interval $I = [a, b]$. Accordingly, the new interval bounding the maximal relaxation is defined as $\overline{I}_F = (a - \lceil \gamma_F \cdot |I_F| \rceil, b + \lceil \gamma_F \cdot |I_F| \rceil)$ where $\gamma_F > 0$ is a user-defined tolerance parameter.

A similar tolerance $0 < \gamma_G \leq 1$ can be defined for the *globally* as well where the relaxation is allowed over $\overline{I}_G = [a, b] \setminus \left(\left[a, a + \left\lfloor \gamma_G \cdot \frac{|I_G|}{2} \right\rfloor \right) \cup \left(b - \left\lfloor \gamma_G \cdot \frac{|I_G|}{2} \right\rfloor, b \right] \right)$. Note that, $\overline{I}_F$ and $\overline{I}_G$ are not the relaxed intervals, but they exclusively bound the allowable relaxation according to user preferences.

Next, we define the temporal relaxation metrics that we want to keep minimum. The temporal relaxation metric simply comprises the amount of violation with respect to the maximum allowable relaxation. We define two normalized metrics below to measure the temporal relaxation among the tasks with *finally* and *globally* operators, respectively.

Definition 5.3.1 (Temporal Relaxation Metric)**.** *Given an STL specification with the syntax in* (5.1)*, the temporal relaxation metrics for the finally and globally operators are defined as:*

Finally

$$\tau(\mathbf{x}, F_{[a,b]}\varphi, t) := \begin{cases} \dfrac{\max(\tau_{\underline{F}}, \tau_{\overline{F}})}{\gamma_F \cdot |I_F|}, \ \textit{if} \ \exists t' \in t \oplus [a - \tau_{\underline{F}}, b + \tau_{\overline{F}}] \\ \textit{s.t.} \ \gamma_F \cdot |I_F| \geq \tau_{\underline{F}}, \tau_{\overline{F}} \geq 0 \ \textit{and} \ (\mathbf{x}, t') \models \varphi, \\ \quad 1, \qquad\qquad\qquad\qquad \textit{otherwise.} \end{cases} \tag{5.2}$$

Globally

$$\tau(\mathbf{x}, G_{[a,b]}\varphi, t) := \begin{cases} \dfrac{\tau_{\underline{G}} + \tau_{\overline{G}}}{\gamma_G \cdot |I_G|}, \qquad \textit{if} \ \forall t' \in t \oplus [a + \tau_{\underline{G}}, b - \tau_{\overline{G}}] \\ \textit{s.t.} \ \gamma_G \cdot \dfrac{|I_G|}{2} \geq \tau_{\underline{G}}, \tau_{\overline{G}} \geq 0 \ \textit{and} \ (\mathbf{x}, t') \models \varphi, \\ \quad 1, \qquad\qquad\qquad\qquad \textit{otherwise.} \end{cases} \tag{5.3}$$

Proposition 5.3.1 (One-sided Relaxation of Finally Tasks)**.** *Suppose that* $\tau_{\underline{F}}, \tau_{\overline{F}} \in \mathbb{Z}_{\geq 0}$*. Let* $\phi = F_{[a,b]}\,\varphi$ *be a finally task endowed with the temporal relaxation metric* $\tau(\mathbf{x}, F_{[a,b]}\varphi, t)$ *defined according to* (5.2)*. When* $\tau(\mathbf{x}, F_{[a,b]}\varphi, t)$ *is minimized, the resulting relaxed interval* $[a - \tau_{\underline{F}}, b + \tau_{\overline{F}}]$ *can have at least one of* $\tau_{\underline{F}}$ *and* $\tau_{\overline{F}}$ *equal to zero.*

Proof. Without loss of generality, let $t = 0$ and the required single satisfaction happen at $b+\tau_{\overline{F}}$. Suppose that the minimally relaxed interval can only be obtained as $[a-\tau_{\underline{F}}, b+\tau_{\overline{F}}]$ with $\tau_{\underline{F}} \neq 0$ and $\tau_{\overline{F}} \neq 0$. Now, consider another interval $[a, b + \tau_{\overline{F}}]$ with $\tau_{\underline{F}} = 0$. Since $(\mathbf{x}, t) \models F_{[a,b+\tau_{\overline{F}}]}\varphi \implies (\mathbf{x}, t) \models F_{[a-\tau_{\underline{F}}, b+\tau_{\overline{F}}]}\varphi$ via the STL semantics in (2.3), $[a - \tau_{\underline{F}}, b + \tau_{\overline{F}}]$ with $\tau_{\underline{F}} \neq 0$ and $\tau_{\overline{F}} \neq 0$ cannot be the minimally relaxed interval. Hence, at least one of $\tau_{\underline{F}}$ and $\tau_{\overline{F}}$ can be zero when the interval is minimally relaxed. □

For instance, consider $\Phi = F_{[5,10]}\varphi$, and the closest satisfaction is $(\mathbf{x}, 15) \models \varphi$. The relaxed specification would be $\Phi^\tau = F_{[5,15]}\varphi$. Alternatively, if $(\mathbf{x}, 3) \models \varphi$ is the closest satisfaction instant, then the relaxed specification would be $\Phi^\tau = F_{[3,10]}\varphi$. When $\tau_{\underline{F}} = \tau_{\overline{F}} = 0$, then the original specification is satisfied, i.e., $(\mathbf{x}, t) \models F_{[a,b]}\varphi$, without any temporal relaxation, i.e., $\tau(\mathbf{x}, F_{[a,b]}\varphi, t) = 0$.

Note that the temporal relaxation metrics in (5.2) and (5.3) are normalized and take values within $[0, 1]$. That is, when no relaxation is needed and the original specification is achieved, the relaxation metric is 0 for each operator, and when the allowable relaxation amount is exceeded or the task is completely failed, the metric becomes 1 and the task is removed. Hence, the aim will be minimizing the temporal relaxation during the satisfaction of an STL specification Φ in the face of infeasibilities.

Building on top of $\tau(\mathbf{x}, F_{[a,b]}\varphi, t)$ and $\tau(\mathbf{x}, G_{[a,b]}\varphi, t)$ in (5.2) and (5.3), we recursively define a general temporal relaxation metric $\tau(\mathbf{x}, \Phi, t)$ for the STL specification Φ as follows:

$$\tau(\mathbf{x}, \Phi_1 \wedge \Phi_2, t) := \frac{\tau(\mathbf{x}, \Phi_1, t) \ + \ \tau(\mathbf{x}, \Phi_2, t)}{2}, \tag{5.4a}$$

$$\tau(\mathbf{x}, \Phi_1 \vee \Phi_2, t) := \min\big(\tau(\mathbf{x}, \Phi_1, t), \tau(\mathbf{x}, \Phi_2, t)\big), \tag{5.4b}$$

$$\tau(\mathbf{x}, G_{[a,b]}\Phi, t) := \max_{t' \in t \oplus [a,b]} \tau(\mathbf{x}, \Phi, t'), \tag{5.4c}$$

$$\tau(\mathbf{x}, F_{[a,b]}\Phi, t) := \min_{t' \in t \oplus [a,b]} \tau(\mathbf{x}, \Phi, t'). \tag{5.4d}$$

By definition, the temporal relaxation is allowed for the innermost temporal STL operators via (5.2) and (5.3) when multiple of them are nested. In other words, when $\Phi = F_{[a,b]}\varphi$ or $\Phi = G_{[a,b]}\varphi$, i.e., $\Phi = \phi$ as per (5.1), we use temporal relaxation metrics in (5.2) and (5.3), respectively. However, when there are multiple nested temporal operators, e.g., $\Phi = G_{[a,b]}F_{[c,d]}\varphi$, we use the definitions in (5.4c) and (5.4d) for the outer ones together with (5.2) and (5.3) inside them.

Corollary 5.3.1. *For a given STL specification Φ with the syntax in* (5.1)*, the overall relaxation metric defined according to* (5.4) *is bounded such that $0 \leq \tau(\mathbf{x}, \Phi, t) \leq 1$, $\forall t$.*

Proof. This follows directly from the semantics of temporal relaxation metric in (5.4) together with (5.2) and (5.3). □

It is worth noting that unlike the standard STL quantifying metrics (e.g., [25]), we use min for the disjunction or *finally* (in (5.4b) and (5.4d)), and max for the *globally* (in (5.4c)). This is because the lower temporal relaxation metric implies closer satisfaction to the original specification. Moreover, the averaging in (5.4a) enables us to measure collective performance that is not dominated by critical values.

Proposition 5.3.2 (Soundness of the Temporal Relaxation Metric)**.** *For a given STL specification Φ with the syntax in* (5.1)*, the temporal relaxation metric defined according to* (5.4) *is sound in the sense that $\tau(\mathbf{x}, \Phi, t) = 0 \Longrightarrow (\mathbf{x}, t) \models \Phi$*[1]*.*

[1]The reverse $\tau(\mathbf{x}, \Phi, t) = 0 \Longleftarrow (\mathbf{x}, t) \models \Phi$ may not always be true as the satisfaction over the original time intervals implies satisfaction over arbitrarily relaxed time intervals as well with nonzero relaxation parameters according to (5.2) and (5.3).

Proof. By recursive definition in (5.4) together with (5.2) and (5.3), no temporal relaxation, i.e., $\tau(\mathbf{x}, \Phi, t) = 0$, implies that all tasks in Φ expressed according to the syntax in (5.1) have either $\tau(\mathbf{x}, F_{[a,b]}\varphi, t') = 0$ or $\tau(\mathbf{x}, G_{[a,b]}\varphi, t') = 0$ with $t' \geq t$. Meaning, for all relaxation parameters in Φ, we have $\tau_{\underline{F}} = \tau_{\overline{F}} = \tau_{\underline{G}} = \tau_{\overline{G}} = 0$ as per (5.2) and (5.3) which, by definition, yields satisfaction within the original interval $[a, b]$ for all tasks. Hence, it follows by construction that the overall specification is satisfied on time, $(\mathbf{x}, t) \models \Phi$. $\square$

Proposition 5.6.1 along with Corollary 5.3.1 can be interpreted as follows: In the best case, the given STL specification is satisfied within the original time intervals with $\tau(\mathbf{x}, \Phi, t) = 0$. Therefore, an extra performance such as achieving tasks for a longer time than required is not demanded, unlike the optimization of other quantifying metrics of STL. However, when it is not feasible to achieve Φ, the minimization of $\tau(\mathbf{x}, \Phi, t)$ leads to the satisfaction of spatial requirements specified in Φ with a minimal temporal relaxation. Furthermore, this minimal relaxation may potentially yield structural changes on the Φ such as compromising highly demanding tasks which may jeopardize the success of the others.

Example 5.2.1 (*Cont'd*)**.** *Consider again the two signals* $\mathbf{x}'$ *(blue) and* $\mathbf{x}''$ *(red) in Fig. 5.1. Recall that both signals have the same time robustness value even for different STL specifications* $\Phi_1 = G_{[15,60]}\mathbf{x} \geq h_1$ *and* $\Phi_2 = \Phi_1 \wedge F_{[75,120]}\mathbf{x} \leq h_2$*. In this regard, the time robustness metric may not capture the signal behaviors completely. We now examine the performance of the new temporal relaxation metric for the same scenario. Let* $I_1 = [15, 60]$ *and* $I_2 = [75, 120]$ *with* $|I_1| = |I_2| = 46$*. Assume that the tolerance parameters are defined as* $\gamma_F = 1$ *and* $\gamma_G = 1$*, and note the temporal relaxation parameters* $\tau_{\overline{G}} = \tau_{\underline{G}} = 9$ *and* $\tau_{\overline{F}} = 4$ *from Fig. 5.1. As both signals violate* Φ_1*, the violation amounts can be calculated using the proposed temporal relaxation metric as* $\tau(\mathbf{x}', \Phi_1, 0) = 9/46$ *and* $\tau(\mathbf{x}'', \Phi_1, 0) = (9 + 9)/46 = 18/46$ *implying the temporal relaxation made for* $\mathbf{x}'$ *is lower than that for* $\mathbf{x}''$*. Similarly, when quantifying the violations for* Φ_2*, the ones for* Φ_1 *should neither dominate the computation nor be completely useless, but contribute to the calculation of the cumulative relaxation. Since* $\mathbf{x}'$ *completely satisfies the finally task, the only violation comes from* Φ_1 *but averaged due to total number of two tasks as* $\tau(\mathbf{x}', \Phi_2, 0) = \frac{9}{2 \cdot 46}$*. On the other hand,* $\mathbf{x}''$ *violates both tasks leading to* $\tau(\mathbf{x}'', \Phi_2, 0) = \left(\frac{18}{46} + \frac{4}{46}\right)/2 = \frac{11}{46}$*. These temporal relaxations for two signals yield the following relaxed specifications:* $\Phi_2^{\tau'} = G_{[15,60-\tau_{\underline{G}}]}\mathbf{x} \geq h_1 \wedge F_{[75,120]}\mathbf{x} \leq h_2$ *for* $\mathbf{x}'$

requiring less relaxation than that of $\Phi_2^{\tau''} = G_{[15+\tau_{\underline{G}},60-\tau_{\overline{G}}]}\mathbf{x} \geq h_1 \wedge F_{[75,120+\tau_{\overline{F}}]}\mathbf{x} \leq h_2$, *the relaxed specification for* $\mathbf{x}''$.

Definition 5.3.2 (Time Interval Similarity[2])**.** *The similarity between two time intervals* I_1 *and* I_2 *is measured via*

$$\mathcal{S}(I_1, I_2) := \frac{|I_1 \cap I_2|}{\max\big(|I_1|, |I_2|\big)}.$$

Proposition 5.3.3 (Minimally Modified Intervals)**.** *Let* ϕ *be an STL task with a single temporal operator over an original interval of* $I = [a, b]$ *as in* (5.1)*, e.g.,* $\phi = F_{[a,b]}\,\varphi$ *or* $\phi = G_{[a,b]}\,\varphi$*. Suppose that two feasible minimal relaxations of* ϕ*,* $\tau'(\mathbf{x}, \phi, t)$*,* $\tau''(\mathbf{x}, \phi, t) \in [0, 1]$*, are obtained over the relaxed intervals of* I' *and* I''*, respectively. If* $\tau'(\mathbf{x}, \phi, t) \leq \tau''(\mathbf{x}, \phi, t)$ *under the same user tolerances* γ_F *and* γ_G*, then* $\mathcal{S}(I', I) \geq \mathcal{S}(I'', I)$.

Proof. ***Finally Case:*** As shown in Prop. 5.3.1, minimal temporal relaxation of finally operator enables one of the relaxation parameters $\tau_{\underline{F}}$ or $\tau_{\overline{F}}$ to be equal to zero. Without loss of generality, suppose that the relaxations take place at the right-hand side (future) of the original intervals, i.e., $\tau'_{\underline{F}} = \tau''_{\underline{F}} = 0$. Then the condition in the premise implies $\tau'_{\overline{F}} \leq \tau''_{\overline{F}}$ according to (5.2). It follows from the relaxed intervals $I' = [a, b + \tau'_{\overline{F}}]$ and $I'' = [a, b + \tau''_{\overline{F}}]$ that $\mathcal{S}(I', I) \geq \mathcal{S}(I'', I)$ as per Def. 5.3.2.

Globally Case: Suppose that the relaxed intervals are $I' = [a + \tau'_{\underline{G}}, b - \tau'_{\overline{G}}]$ and $I'' = [a + \tau''_{\underline{G}}, b - \tau''_{\overline{G}}]$. Then the condition in the premise implies $\tau'_{\underline{G}} + \tau'_{\overline{G}} \leq \tau''_{\underline{G}} + \tau''_{\overline{G}}$ according to (5.3); thereby, $\mathcal{S}(I', I) \geq \mathcal{S}(I'', I)$ as per Def. 5.3.2. □

Hence, keeping the temporal relaxation minimum is desired to achieve a relaxed specification as close as possible to the original one.

Problem 5.3.1 (Minimization of Temporal Relaxation)**.** *Given an initial state* $\mathbf{x}_0$*, an STL specification* Φ*, some user-defined relaxation tolerances* γ_F *and* γ_G*, the minimal temporal relaxation problem for a dynamical system can be formulated as an optimization problem as follows:*

$$\begin{aligned} \mathbf{u}^* = \arg\min\ & \tau(\mathbf{x}, \Phi, 0) \\ s.t.\quad & \mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t)), \\ & \mathbf{x}(t) \in \mathcal{X},\ \mathbf{u}(t) \in \mathcal{U},\ \forall t, \end{aligned} \tag{5.5}$$

[2]Note that this similarity measure is a form of Jaccard similarity index when one interval is contained in the other.

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ denotes the dynamics of the system; $\mathcal{X}$ and $\mathcal{U}$ are the admissible state and control sets, respectively.

If Φ can be satisfied by a trajectory starting from the given initial state $\mathbf{x}_0$, then solving Problem 5.3.1 generates a feasible trajectory. Otherwise, solving it results in a trajectory that minimally relaxes the time bounds of Φ under the allowable relaxations provided by the user. Such a trajectory is different than the one that can be found by maximizing space robustness (which does not modify the original time bounds) and time robustness.

5.4 Temporally Relaxed STL Control Synthesis

Mixed-integer encoding of STL constraints is a common approach in control synthesis under an STL specification [14]. If the dynamics $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is linear (e.g., $\mathbf{x}(t+1) = A\,\mathbf{x}(t) + B\,\mathbf{u}(t)$ where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are the system matrices), the mixed-integer encoding of the STL specification Φ and the temporal relaxation metric $\tau(\mathbf{x}, \Phi, 0)$ renders the optimization problem in (5.5) a mixed-integer linear program (MILP) as long as the predicate function $p(\mathbf{x}(t))$ is linear as well.

The authors of [68] use the framework in [14] to maximize right time robustness by counting the consecutive satisfactions and violations via MILP encoding. While we use a similar idea of counting constraints, there are substantial differences as the counting takes place within particular time intervals and the counters are used to encode the temporal relaxation metric according to (5.2), (5.3), and (5.4).

Next, we will present the MILP encoding of variables that count the consecutive time instances of satisfaction (i.e., $z_t^\varphi = 1$) and violation (i.e., $z_t^\varphi = 0$), respectively. Such an encoding is proposed in [68] only forward in time to calculate the right time robustness. To see signal behaviors such as cumulative temporal relaxation (as introduced in (5.4a)), we formulate a similar satisfaction/violation counting mechanism both forward and backward in time. Then we follow the recursive definition of the general temporal relaxation metric proposed in (5.2), (5.3), and (5.4) to build MILP constraints representing them.

We will use the variables of $r_t^1(\varphi), l_t^1(\varphi) \in \mathbb{Z}_{\geq 0}$ and $r_t^0(\varphi), l_t^0(\varphi) \in \mathbb{Z}_{\leq 0}$ which denote the number of instants with consecutive satisfaction (1) and violation (0) of φ starting

from t toward the future (r) and past (l), respectively. The numbers of consecutive past satisfaction and violation instants of φ starting from t (for $t' \leq t$) are defined forward in time, respectively, as follows:

$$\begin{aligned} l_t^1(\varphi) &= z_t^{\varphi} \cdot (l_{t-1}^1(\varphi) + 1), \\ l_t^0(\varphi) &= (1 - z_t^{\varphi}) \cdot (l_{t-1}^0(\varphi) - 1). \end{aligned} \tag{5.6}$$

Similarly, the numbers of consecutive satisfaction and violation instants of φ at future starting from t (for $t' \geq t$) are defined backward in time, respectively, as below

$$\begin{aligned} r_t^1(\varphi) &= z_t^{\varphi} \cdot (r_{t+1}^1(\varphi) + 1), \\ r_t^0(\varphi) &= (1 - z_t^{\varphi}) \cdot (r_{t+1}^0(\varphi) - 1). \end{aligned} \tag{5.7}$$

By construction, $r_t^1(\varphi)$ and $l_t^1(\varphi)$ count the maximum numbers of consecutive instants with $z_{t'}^{\varphi} = 1$ for $t' \geq t$ and $t' \leq t$, respectively. On the other hand, $r_t^0(\varphi)$ and $l_t^0(\varphi)$ count the maximum numbers of consecutive instants with $z_{t'}^{\varphi} = 0$ and multiply these values by -1.

For the given time intervals $I_F = I_G = [a, b]$, boundary conditions for the calculations in (5.6) and (5.7) are determined based on the type of temporal operator. For *finally*, the calculations in (5.8) start forward and backward in time with $l^0_{t+a-\lceil \gamma_F \cdot |I_F| \rceil}(\varphi) = 0$ and $r^0_{t+b+\lceil \gamma_F \cdot |I_F| \rceil}(\varphi) = 0$, respectively, where $\lceil \gamma_F \cdot |I_F| \rceil$ is the exclusive bound for the allowable relaxation. On the other hand, the boundary conditions for the calculations for *globally* in (5.9) is started forward and backward in time with $l^1_{t+a-1} = 0$ and $r^1_{t+b+1} = 0$, respectively.

We continue with the MILP encoding of temporal relaxation metrics for *finally* and *globally* operators constructed in (5.2) and (5.3), respectively.

Finally

$$\tau(\mathbf{x}, F_{[a,b]}\varphi, t) = (z_t^{F_{[a,b]}\varphi} - 1) \cdot \frac{\max\left(l^0_{t+a}(\varphi), r^0_{t+b}(\varphi)\right)}{\lceil \gamma_F \cdot |I_F| \rceil}, \tag{5.8}$$

where the variable $z_t^{F_{[a,b]}\varphi}$ denotes the Boolean satisfaction of the original task, i.e., $z_t^{F_{[a,b]}\varphi} = 1 \iff (\mathbf{x}, t) \models F_{[a,b]}\varphi$, and its MILP encoding is as follows:

$$z_t^{F_{[a,b]}\varphi} = \bigvee_{t'=t+a}^{t+b} z_{t'}^{\varphi} \ .$$

Hence, if the *finally* task is already satisfied there is no need to consider relaxation beyond the original interval or demanding more satisfactory time instants. This way, we keep the original requirement of satisfying φ at any single instant within $[a, b]$ if possible, or at an instant as close as possible to the original interval otherwise.

Globally

$$\tau(\mathbf{x}, G_{[a,b]}\varphi, t) = 1 - z_t^{G_{[a+\beta,b-\beta]}\varphi} \cdot \left(1 - \frac{|I_G| - l^1_{t+\lfloor \frac{a+b}{2} \rfloor}(\varphi) - r^1_{t+\lceil \frac{a+b}{2} \rceil}(\varphi)}{\gamma_G \cdot |I_G|}\right), \tag{5.9}$$

where $\beta = \left\lfloor \gamma_G \cdot \frac{|I_G|}{2} \right\rfloor$ and the fraction is the ratio of failed time instants to the allowed relaxation. The variable $z_t^{G_{[a+\beta,b-\beta]}\varphi}$ denotes the Boolean satisfaction of the task under maximum allowable relaxation, and its MILP encoding is as follows:

$$z_t^{G_{[a+\beta,b-\beta]}\varphi} = \bigwedge_{t'=t+a+\beta}^{t+b-\beta} z_{t'}^{\varphi} \ .$$

Therefore, the *globally* task can be satisfied under an allowable relaxation, and otherwise removed. Note that when $\gamma_G = 1$, the Boolean satisfaction variable becomes redundant as even a single satisfied time instant is an acceptable relaxation for the $\gamma_G = 1$ case. The encoding in (5.9) enforces the satisfaction at the middle and possible relaxations at both ends as required by the definition in (5.3). As $\lfloor \frac{a+b}{2} \rfloor$ and $\lceil \frac{a+b}{2} \rceil$ denote the same time step when $a+b$ is even, for such cases, we consider the satisfaction at only one of these two instants in the numerator of (5.9) to prevent double counting.

After encoding the temporal relaxation metrics of $\tau(\mathbf{x}, F_{[a,b]}\varphi, t)$ and $\tau(\mathbf{x}, G_{[a,b]}\varphi, t)$, the MILP formulation of $\tau(\mathbf{x}, \Phi, t)$ for the overall STL specification Φ can be done according to the recursive definitions in (5.4) and using the recipe in [14] for quantitative encoding of min/max functions.

Unlike the time robustness, the temporal relaxation metric is not defined throughout the whole mission horizon. In this regard, much fewer variables are used in MILP encoding compared to time robustness yielding substantially faster solutions in addition to obtaining different signal behaviors. Moreover, counting the violations only for the *finally* and the satisfactions only for the *globally* operator within predetermined bounds ($\overline{I}_F$ and $\overline{I}_G$) keeps the number of optimization variables low and contribute to the computational efficiency.

Algorithm 1: STL Control Synthesis under Minimal Temporal Relaxation using MILP encoding

input : Initial state $\mathbf{x}_0$, linear dynamics $f(\mathbf{x}, \mathbf{u})$, STL specification Φ, and user tolerance inputs γ_F and γ_G.

1 Define constraints of the dynamics along the mission horizon H;

2 Formulate MILP constraints on z_t^{φ} for each task φ over $t \in [0, H]$ (i.e., predicates and their conjunction/disjunction);

3 Formulate STL constraints to define $\tau(\mathbf{x}, \Phi, 0)$ built recursively upon the core temporal relaxation metrics $\tau(\mathbf{x}, F_{[a,b]}\varphi, t)$ in (5.2) and $\tau(\mathbf{x}, G_{[a,b]}\varphi, t)$ in (5.3) according to (5.4);

4 Solve (5.5) in MILP format to extract state and input sequences;

output : Relaxed STL specification Φ^{τ} with the amount of temporal relaxation on each task, state trajectory $\mathbf{x}$, and input policy $\mathbf{u}^*$ that achieve Φ^{τ}.

Theorem 5.4.1 (Completeness)**.** *For an STL specification Φ defined using the syntax in* (5.1) *with linear predicates, if any task Φ_j in $\Phi = \bigwedge_{i=1}^{k} \Phi_i$ is feasible alone, then Alg. 1 returns $\tau(\mathbf{x}, \Phi, t) < 1$ implying the spatial requirement of at least one task is satisfied.*

Proof. MILP encoding of the temporal relaxation in lines 2-3 of Alg. 1 implies by construction (via Eqs. (5.8) and (5.9)) that the minimizer of the optimization problem solved in line 4 chooses the lowest cumulative temporal relaxation metric among the alternatives. Suppose that Φ_j is a feasible task in Φ. Now we will consider two cases based on the satisfaction status of the Φ_j:

Case 1 (Φ_j is completed on time): Since no relaxation is needed for Φ_j, even if all other tasks are completely violated (i.e., $\tau(\mathbf{x}, \Phi_i, 0) = 1,\ \forall i \neq j$), the resultant cumulative temporal relaxation is bounded as $\tau(\mathbf{x}, \Phi, 0) \leq \frac{\tau(\mathbf{x}, \Phi_j, 0) + k - 1}{k}$. It follows from $\tau(\mathbf{x}, \Phi_j, 0) = 0$ that $\tau(\mathbf{x}, \Phi, 0) \leq \frac{k-1}{k} < 1$. Let us denote $\tau(\mathbf{x}, \Phi, 0)$ in this case as τ_{case_1} for future reference.

Case 2 (Φ_j is temporally relaxed or entirely removed): Suppose that the feasible task Φ_j is relaxed, and the algorithm still returns $\tau(\mathbf{x}, \Phi, 0) = \frac{\tau(\mathbf{x}, \Phi_j, 0) + \sum_{i \neq j} \tau(\mathbf{x}, \Phi_i, 0)}{k} = 1$ meaning no spatial requirement is achieved within the relaxed time intervals. However, for the Alg. 1 to return a temporal relaxation for Φ_j, instead of τ_{case_1}, we need to have $\frac{\tau(\mathbf{x}, \Phi_j, 0) + \sum_{i \neq j} \tau(\mathbf{x}, \Phi_i, 0)}{k} \leq \tau_{case_1} \leq \frac{k-1}{k}$ which is a contradiction and yields $\tau(\mathbf{x}, \Phi, 0) < 1$ since $0 \leq \tau(\mathbf{x}, \Phi, 0) \leq 1$ as presented in Corollary 5.3.1. □

5.5 Temporal Relaxation vs. Time Robustness

Given an initial state $\mathbf{x}_0$, if Φ cannot be satisfied by a trajectory starting from $\mathbf{x}_0$, the trajectories found by minimizing temporal relaxation and maximizing time robustness are not the same. In the literature, either right or left time robustness is maximized, and addressing both past and future relaxations at the same time is missing. Even if a unified approach can be proposed to account for this, there exists a major difference regarding the computation of time robustness, which uses min/max functions at each level. This leads to the major violation cases dominating the value of the overall time robustness and cannot differentiate the satisfaction/violation of the other tasks (as illustrated in Example 5.2.1). Furthermore, the time robustness becomes inconclusive when it is equal to zero. This may potentially yield problems in differentiating between a near miss and a tight satisfaction. For instance, let a specification be given as $\phi = F_{[5,10]}\mu$ with two scenarios: either $(\mathbf{x}, 10) \models \mu$ (tight satisfaction) or $(\mathbf{x}, 11) \models \mu$ (near miss). As the (right) time robustness is calculated by $\theta^+(\mathbf{x}, \phi, 0) = \max_{t\in[5,10]} \theta^+(\mathbf{x}, \mu, t) = \theta^+(\mathbf{x}, \mu, 10) = 0$ for both cases, we have the same time robustness score for them. Hence, satisfying ϕ within the specified time interval but at the very last step or violating it by one time step becomes indifferent. Hence, determining the need to relax each task is not possible by using the notion of time robustness.

Moreover, maximizing time robustness in the presence of a violation leads to checking the overall mission horizon to ensure the satisfaction of the corresponding task (or predicate) at least once so that a finite negative time robustness can be obtained. However, this can cause the mandatory satisfaction instant to be arbitrarily far from the original interval which may not be desirable for some missions. On the other hand, the proposed formulation can accommodate user input for limiting maximum relaxations and minimizes temporal relaxation under the allowable relaxation tolerances. If a task requires relaxation more than the allowable relaxation, then that task is removed from the specification. Note that a similar task removal by maximizing time robustness might be possible. However, that becomes a more computationally intensive process as it requires iteratively computing trajectories and their corresponding time robustness values and deciding to remove a task if the resulting time robustness is smaller than a user-defined threshold. On the other hand, the proposed approach of minimizing temporal

relaxation does not require iterative trajectory computation and solves the problem in a single shot while still incorporating the user preferences.

5.5.1 Benchmark Analysis

We develop a control synthesis tool that generates trajectories satisfying the given STL specification under minimal temporal relaxation by solving the problem in (5.5). We use $\gamma_F = \gamma_G = 1$ to bound the relaxations over *finally* and *globally* tasks, respectively. When the task intervals are close to $t = 0$ or $t = H$, we shift the signals as they are defined on $\mathbb{Z}_{\geq 0}$ or extend the mission horizon H accordingly so that the allowable relaxation bounds $\overline{I}_F$ and $\overline{I}_G$ are captured within $[0, H]$.

To illustrate the benefits of the new temporal relaxation metric, here we address control synthesis for an autonomous robot with discrete-time double-integrator dynamics as

$$\mathbf{x}(t+1) = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}(t) + \begin{bmatrix} 0.5\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & 0.5\Delta t^2 \\ 0 & \Delta t \end{bmatrix} \mathbf{u}(t),$$

with the state vector $\mathbf{x} = [x, v_x, y, v_y]^T$ where v_x, $v_y \in \mathbb{R}$ are the velocities in x, $y \in \mathbb{R}$ directions, respectively; and the input vector $\mathbf{u} = [u_x, u_y]^T$ where u_x,$u_y \in \mathbb{R}$ are the accelerations along the given directions with the limit of $|u_x|$,$|u_y| \leq 2.2$. The mission scenario requires the robot to visit R_A and R_B some time within $[32, 42]$ and $[77, 87]$, respectively, and always stay inside R_C within $[47, 67]$. These requirements are expressed by the STL specification:

$$\Phi_{case} = F_{[32,42]} R_A \wedge F_{[77,87]} R_B \wedge G_{[47,67]} R_C, \tag{5.10}$$

where visiting regions can be captured by the conjunction of linear predicates as follows:

$$R_A = x \geq 4 \wedge x \leq 8 \wedge y \geq 6 \wedge y \leq 10,$$
$$R_B = x \geq 16 \wedge x \leq 20 \wedge y \geq 6 \wedge y \leq 10,$$
$$R_C = x \geq 10 \wedge x \leq 14 \wedge y \geq -6 \wedge y \leq -2.$$

The STL control synthesis problem under Φ_{case} is solved by i) minimizing the proposed temporal relaxation metric $\tau(\cdot)$ using Alg. 1 and encoding in Sec. 5.4, ii) maximizing right $\left(\theta^+(\cdot)\right)$ and left $\left(\theta^-(\cdot)\right)$ time robustness metrics via the approach in [68] for

comparison. Results are given in Table 5.1, and realized trajectories with the marked satisfactory instances and the original intervals are presented in Fig. 5.2. The use of the proposed metric results in smaller temporal relaxation compared to time robustness maximizing trajectories. Furthermore, the fewer number of constraints and variables in our encoding yields more efficient computation time as well. As a result, the captured signal behaviors via all three approaches in Table 5.1 lead to the following realized STL specifications with marked relaxations in the time intervals:

$$\Phi^{\tau}_{case} = F_{[\mathbf{28^*},42]}R_A \wedge F_{[77,\mathbf{89^*}]}R_B \wedge G_{[\mathbf{53^*},67]}R_C,$$
$$\Phi^{\theta^+}_{case} = F_{[32,42]}R_A \wedge F_{[77,\mathbf{95^*}]}R_B \wedge G_{[\mathbf{59^*},\mathbf{65^*}]}R_C,$$
$$\Phi^{\theta^-}_{case} = F_{[\mathbf{21^*},42]}R_A \wedge F_{[77,87]}R_B \wedge G_{[\mathbf{50^*},\mathbf{62^*}]}R_C.$$

As depicted above, using standard time robustness can assess relaxations toward either right or left. Therefore, one of the *finally* tasks had to be satisfied when maximizing each time robustness metric. Moreover, relaxation on one end of the *globally* interval enables a smaller relaxation on the other end without additional cost, hence causing unnecessary relaxation. The proposed metric, on the other hand, cumulatively assesses the relaxations on both ends and among other tasks, hence satisfying an STL specification with time intervals as close as possible to the original ones.

Table 5.1: Simulation results and optimization parameters for temporal relaxation $\big(\tau(\mathbf{x}, \Phi_{case}, 0)\big)$ minimization and standard time robustness $\big(\theta^+(\mathbf{x}, \Phi_{case}, 0)$ and $\theta^-(\mathbf{x}, \Phi_{case}, 0)\big)$ maximization.

Optimized Metric	# Constraints	# Variables		Computation Time $[s]$		Temporal Relax. $\in [0, 1]$
		Continuous	Integer	YALMIP [81]	Solver	
$\tau(\mathbf{x}, \Phi_{case}, 0)$	5505	310	1821	1.66	39.2	0.277
$\theta^+(\mathbf{x}, \Phi_{case}, 0)$	8199	970	1861	1.98	1118.6	0.465
$\theta^-(\mathbf{x}, \Phi_{case}, 0)$	8175	964	1861	2.32	632.9	0.461

Relaxation via Structural Changes: Different from standard STL robustness metrics, the proposed temporal relaxation metric may result in task removal to avoid significant cascaded delays in task achievement. For instance, consider the specification: $\Phi_{case'} = G_{[15,25]}x \geq 5 \wedge F_{[35,45]}x \leq -5 \wedge F_{[75,85]}x \geq 5$. As the time robustness can be arbitrarily small (and negative), even an utter delay is better than not achieving the task. Hence, maximizing it returns the trajectory in Fig. 5.3 which fails all tasks. In fact, removing one task may lead to the satisfaction of others on time. However, the trajectory

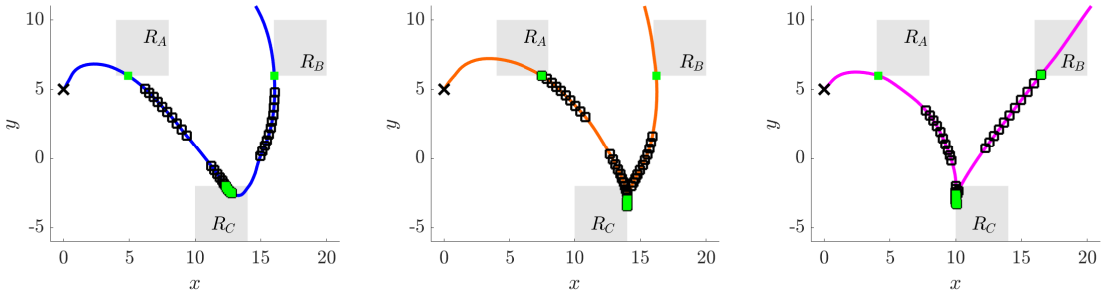


Figure 5.2: Trajectories of the robot under the proposed minimal temporal relaxation control approach (on the left) together with right and left time robustness maximizing approaches (on the middle and right, respectively). The mission requirements are defined in (5.10). The initial position is marked by $\times$. The original *globally* and *finally* intervals are denoted by discrete black squares where satisfaction instants within or closest to them are represented via green fillings.

still tries to bound the violation which yields cascaded delays in the remaining tasks. On the other hand, minimizing the temporal relaxation metric requires the removal of *globally* task and enables the completion of others within the original intervals. Hence, Alg. 1 returns relaxed specification of $\Phi^{\tau}_{case'} = F_{[35,45]}x \leq -5 \wedge F_{[75,85]}x \geq 5$ without any relaxation on the time intervals. Moreover, the total solution time for minimizing temporal relaxation is $2.3\,s$ with $\tau(\mathbf{x}, \Phi_{case'}, 0) = 0.33$, i.e., one-third of the specification is compromised, where the time robustness maximization takes much longer, $59.8\,s$.

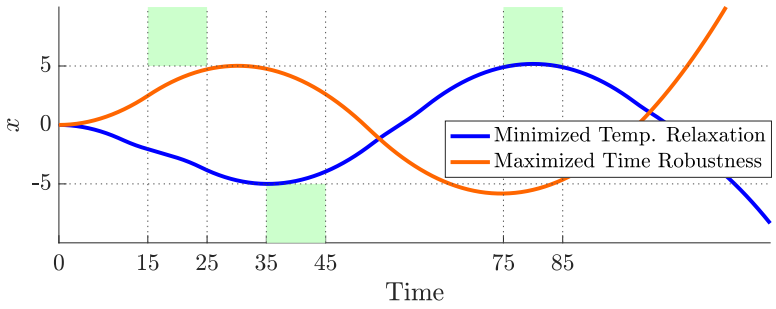


Figure 5.3: Time history of state x. Green regions are desired to be visited via $\Phi_{case'} = G_{[15,25]}x \geq 5 \wedge F_{[35,45]}x \leq -5 \wedge F_{[75,85]}x \geq 5$.

5.6 Resilience via Reactive Motion Planning

The existing time robustness of STL [26] may not facilitate a reactive response due to focusing primarily on critical trajectory segments that might be in the past and still dominate the metric. Moreover, its lack of ability to compromise/remove tasks may yield cascaded delays that exceed user-defined violation limits. Therefore, the proposed temporal relaxation metric is promising in the face of unforeseen events that require a reaction from the robot. Particularly, we can use the proposed metric with online synthesis methods that can react to unexpected changes in the environment and even allow for minimal violations of the original STL specification (by strategically modifying the time windows or removing tasks) for the sake of resilience in motion planning.

5.6.1 Planning Problem in Dynamic Environments

In this section, we consider a robot whose dynamics is modeled by a family of affine holonomic systems as $\dot{\mathbf{x}} = \mathbf{u}$ with an admissible input set $\mathcal{U} := \{\mathbf{u} \in \mathbb{R}^n \mid \|\mathbf{u}\| \leq u_{\max}\}$ where $u_{\max} \in \mathbb{R}_{\geq 0}$ is the maximum input. We also consider a dynamic environment with modes unknown to the robot.

Definition 5.6.1 (Modes of Dynamic Environment)**.** *Transitions between different environment modes with unknown probabilities are represented via a stochastic weighted transition system as a tuple $\mathcal{G}_{env} = (\mathcal{V}, \mathcal{E}, \omega)$ where:*

- *$\mathcal{V}$ is a finite set of environment modes comprising different obstacle locations $\mathcal{O}$;*
- *$\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of transitions/environmental changes;*
- *$\omega : \mathcal{E} \to pr_{env}$ is the transition weight where $pr_{env} \in [0, 1]$ is the probability of switching to a new environment.*

Let $env_i \in \mathcal{V}$ be the i^{th} mode of the environment. We denote the set of obstacles in env_i as $\mathcal{O}_i$. A transition from mode env_i to env_j is possible if $(env_i, env_j) \in \mathcal{E}$. We use row-stochastic transition probabilities such that $\sum_{j \in \mathcal{V}} pr_{\text{env}}^{i,j} = 1, \ \forall i \in \mathcal{V}$, which are unknown to the robot. An example transition system of environment modes is depicted in Fig. 5.4. Such dynamic obstacles or no-go zones can be the case for various scenarios ranging from warehouses with flexible human workstations to agricultural robots avoiding livestock.

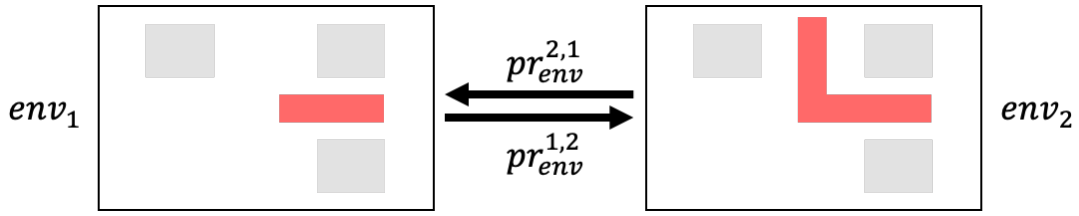


Figure 5.4: Two sample environment modes with fixed goal regions (gray) and dynamic obstacle settings (red).

Consider an H-horizon length STL specification $\Phi := \Phi_{\text{goal}} \wedge \Phi_{\text{safety}}$ such that $\Phi_{\text{safety}} = \bigwedge_{j \in \mathcal{O}} G_{[0,H]} \neg \phi_{\text{obs},j}$ where $\neg \phi_{\text{obs},j}$ implies the avoidance from the obstacle $j \in \mathcal{O}$ with $j \subset \mathbb{R}^n$. Since the obstacles change in the environment during the mission, each environment mode will result in a different safety specification as $env_i \iff \Phi_{\text{safety},i}$ (Fig. 5.4).

In this work, we consider the detection of potential collisions due to environmental changes and the arising need for correction, as the major event that may imply a significant spatial and temporal deviation from the nominal trajectory[3].

Definition 5.6.2 (Triggering Event)**.** *A triggering event is a major event leading to a reaction from the robot. It is a couple* (k, t_k) *with an index* $k \in \{1, 2, \dots k_{\max}\}$ *and occurs at* $t_k \in \mathbb{Z}_{\geq 0}$ *such that* $t_k < t_{k+1}$*. A triggering event implies* $(\mathbf{x}, 0) \not\models \Phi_{safety}$ *for a nominal trajectory* $\mathbf{x}$ *and* $(\mathbf{x}', 0) \models \Phi_{safety}$ *for a new trajectory* $\mathbf{x}'$ *updated at* t_k*.*

The current environment after such an event can be tracked via a function $\sigma : [1, k_{\max}] \to [1, |\mathcal{V}|]$ that maps each event to an environment mode $env_{\sigma(k)} \in \mathcal{V}$ for $t_k \leq t \leq t_{k+1}$.

Note that the mapping via $\sigma(\cdot)$ is onto but not injective, as the same environment can be reactivated later. Considering a finite set of environment modes $\mathcal{V}$, we can obtain the set of all possible specifications $\{\Phi_k := \Phi_{\text{goal}} \wedge \Phi_{\text{safety},\sigma(k)} \mid \forall k\}$ where $\Phi_{\text{safety},\sigma(k)} := \bigwedge_{j \in \mathcal{O}_{\sigma(k)}} G_{[t_k,H]} \neg \phi_{\text{obs},j}$ as different environments are concerned after each triggering event.

[3]The proposed reactive framework can also be used in the face of other events arising from any unmodeled external or internal disturbances.

Assumption 5.6.1 (Feasibility of Safety Tasks)**.** *Let* $\mathbf{x}(t)$ *be the robot position and a triggering event occur at* t_k*. Then* $\mathbf{x}(t_k) \cap \{j \subset \mathbb{R}^n \mid j \in O_{\sigma(k)}\} = \emptyset$*, implying that an obstacle cannot suddenly appear at the current robot location.*

Assumption 5.6.1 implies that an obstacle cannot suddenly appear at the current robot location.

Proposition 5.6.1 (Invariance of Temporal Relaxation)**.** *Consider a trajectory* $\mathbf{x}$*, and an STL specification* Φ *with changing safety features according to the events defined in Def. 5.6.2. If* $\mathbf{x}$ *has no collision, e.g.,* $(\mathbf{x}, 0) \models \bigwedge_j G_{[t_k, t_{k+1}]} \neg \phi_{obs,j},\ \forall k$*, then* $\tau(\mathbf{x}, \Phi_k, 0) = \tau(\mathbf{x}, \Phi_{goal}, 0),\ \forall k$*.*

Proof. It directly follows from the definition of triggering events in Def. 5.6.2 and recursive temporal relaxation definition in (5.4) as $\tau(\mathbf{x}, \Phi_{\text{safety},\sigma(k)}, 0) = 0,\ \forall k$. □

The minimal temporal relaxation of an STL specification can be denoted as $\tau(\mathbf{x}, \Phi, 0) = \tau(\mathbf{x}, \Phi_{\text{goal}}, 0)$, given that no collision takes place in the course of $\mathbf{x}$. Due to the time-varying environment and STL specifications, mission satisfaction requires a reactive control strategy. The planning problem to be solved reactively after an event is formulated as below:

Problem 5.6.1 (Reactive Temporal Relaxation in Dynamic Environments)**.** *Consider a robot moving in an uncertain dynamic environment that potentially changes at each time step as per Def. 5.6.1. By considering the given user tolerances* γ_F *and* γ_G *in Def. 5.3.1, find trajectories that minimally relax the dynamic STL specifications* $\Phi_k = \Phi_{goal} \wedge \Phi_{safety,\sigma(k)}$ *changing with the events defined in Def. 5.6.2. That is, for each event at* t_k *solve the following:*

$$\begin{aligned} \mathbf{u}^* = \arg\min\ & \tau(\mathbf{x}, \Phi_{goal}, 0) \\ s.t.\quad & \dot{\mathbf{x}} = \mathbf{u},\ \mathbf{x}(t) = \mathbf{x}_t,\ \forall t \in [0, t_k], \\ & \mathbf{x} \in \mathcal{X},\ \mathbf{u} \in \mathcal{U},\ (\mathbf{x}, 0) \models \Phi_{safety,\sigma(k)}, \end{aligned} \tag{5.11}$$

where $\mathcal{X}$ *and* $\mathcal{U}$ *are the admissible state and control sets, respectively,* $\mathbf{x}_{t_k}$ *is the known current state when* k^{th} *event takes place, and* $(\mathbf{x}, 0) \models \Phi_{safety,\sigma(k)}$ *implies avoidance from all obstacles that are active within the interval* $[t_k, t_{k+1})$*.*

Let Φ^τ be the temporally relaxed specification after solving Problem 1. That is, the time intervals of Φ_{goal} might be modified in Φ^τ. According to Def. 5.3.1, if Φ can be completely satisfied, then solving Problem 5.6.1 in response to events generates a satisfactory trajectory, i.e., $\Phi^\tau = \Phi$ and $\tau(\mathbf{x}, \Phi_{\text{goal}}, 0) = 0$. Otherwise, its solutions result in minimal violation of the time bounds of Φ under the allowable relaxations provided by the user, i.e., $\tau(\mathbf{x}, \Phi_{\text{goal}}, 0) > 0$. However, neither satisfactory nor minimally relaxed trajectories are guaranteed prior to execution due to unforeseen environmental changes. Therefore, a runtime monitoring and reaction strategy has to be developed.

5.6.2 Reactive Motion Planning with Temporal Relaxation

A replanning strategy under a triggering event can vary from replanning globally to applying local corrections. Figure 5.5 shows a diagram that includes different replanning strategies and when to trigger them. Here, we introduce a hierarchical two-fold optimization for temporal relaxation and space robustness. The robot initially follows the nominal trajectory. In the face of environmental changes, we monitor the trajectory and assess the potential failures. Whenever there exists a risk, we either locally correct the trajectory via CBFs or globally replan the trajectory via mixed-integer linear programs (MILP). A highlight of our approach is shown in Fig. 5.6 and detailed in Alg. 2.

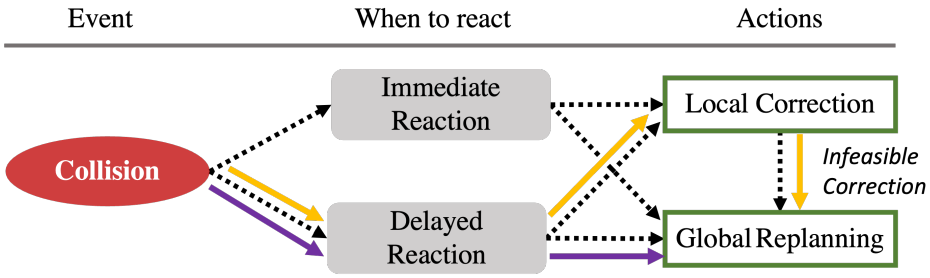


Figure 5.5: Potential recovery strategies in the face of a major event defined in Def. 5.6.2. While black dashed arrows represent all possible options, this section presents approaches shown in purple and yellow arrows.

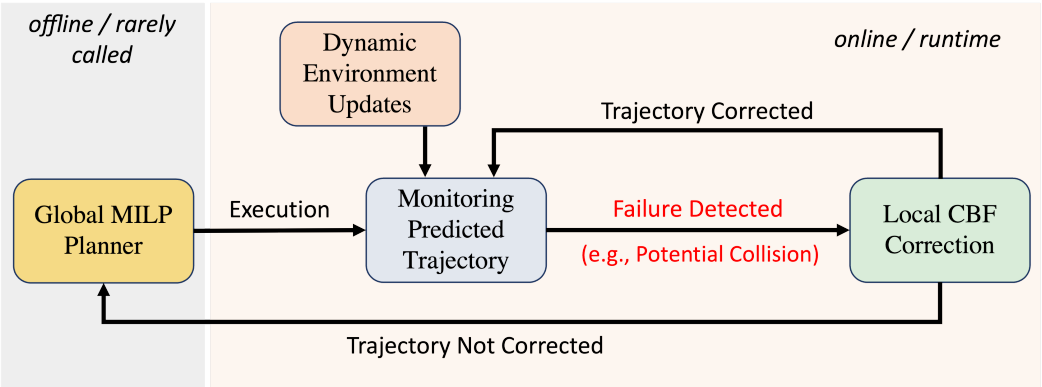


Figure 5.6: Proposed planning and execution scheme.

Algorithm 2: Initial Trajectory Generation Followed by Runtime Monitoring and Reactions

input : STL Specification Φ, input bound $u_{\max}$, user tolerances γ_F, γ_G

Obtain $(\mathbf{x}_{\text{plan}}, \mathbf{u}_{\text{plan}})$ via global MILP planner, minimizing temporal relaxation $\tau(\mathbf{x}_{\text{plan}}, \Phi_{\text{goal}}, 0)$,;

Maximize the space robustness for Φ^τ achieved by $\mathbf{x}_{\text{plan}}$ while minimizing control effort,;

while *Any task remains undecided,* $\mathcal{D} \neq \mathcal{T}$ **do**

  Advance the robot for one time step;

  Get any environment update, $env_{current}$;

  Monitor $\tau(\mathbf{x}_{\text{pred}}, \Phi_{\text{goal}}, 0)$,;

  Check collisions throughout $\mathbf{x}_{\text{pred}}$ over $env_{current}$;

  if $\mathbf{x}_{pred}$ *has a qualifying collision in* $env_{current}$ **then**

    if *Feasible CBF* **then**

      Correct trajectory via CBFs,

    else

      Replan via two-fold MILP planner,

    end

  end

end

return : Realized Specification, Φ^τ

Co-optimization of Temporal Relaxation and Space Robustness

We solve the planning problem in two steps. First, a trajectory with minimal temporal relaxation is found subject to Boolean satisfaction of predicates. Then, we solve a space

robustness maximization problem to obtain trajectories preserving the same temporal relaxation but resilient to disturbances. Both solutions are based on mixed-integer linear programs (MILP). This two-step approach trades off optimality with computational performance.

The first step, i.e., obtaining a temporally relaxed trajectory is already addressed in Sec. 5.4. This solution returns potentially relaxed intervals of the STL tasks in Φ_{goal}. However, it does not take into account the space robustness, i.e., requires only Boolean satisfaction of predicates. On the other hand, this approach is prone to "barely satisfying" trajectories, e.g., touching the target once and immediately leaving. Therefore, even a small, momentary disturbance can yield the violation of the entire task. For this reason, we maximize the space robustness within the potentially relaxed intervals obtained in Sec. 5.4. Similarly, minimization of the control input is also handled at this level to avoid potentially additional temporal relaxation for the sake of less control effort. In [74], a similar hierarchical framework is proposed, however, robustness values of all satisfactory predicates are treated equally which may not fully capture the task-specific success.

Consider the relaxed specification Φ^τ obtained by solving (5.11) with potentially relaxed time intervals or removed tasks. We next solve the following problem:

$$\begin{aligned} \max \;\; & \rho(\mathbf{x}, \Phi^\tau, 0) - \gamma\, \mathcal{J}(\mathbf{u}) \\ s.t. \;\; & \dot{\mathbf{x}} = \mathbf{u},\, \mathbf{x} \in \mathcal{X},\; \mathbf{u} \in \mathcal{U}, \\ & \rho(\mathbf{x}, \Phi^\tau, 0) \geq 0, \end{aligned} \tag{5.12}$$

where $\mathcal{J}(\cdot)$ is a cost function, $\gamma \in \mathbb{R}^+$ is a weight parameter. The MILP encoding of (5.12) can be done similarly to [14].

Proposition 5.6.2 (Conservation of Temporal Relaxation)**.** *Given a temporally relaxed STL specification Φ^τ, solving* (5.12) *does not yield a higher temporal relaxation.*

Proof. It follows from temporal relaxation (5.4) and space robustness (2.3) definitions. Consider a trajectory $\mathbf{x}'$ yielding $\tau(\mathbf{x}', \Phi, 0) = \tau^* > 0$ and a relaxed Φ^τ. Since the constraint in (5.12) implies $\rho(\mathbf{x}, \Phi^\tau, 0) \geq 0 \Rightarrow \tau(\mathbf{x}, \Phi^\tau, 0) = 0$ for any $\mathbf{x}$, we have $\tau(\mathbf{x}, \Phi, 0) = \tau(\mathbf{x}', \Phi, 0) = \tau^*$ by definition. □

Runtime Monitoring

Let $\mathbf{x}_{\text{plan}} = (\mathbf{x}_0, \ldots, \mathbf{x}_H)$ and $\mathbf{u}_{\text{plan}} = (\mathbf{u}_0, \ldots, \mathbf{u}_{H-1})$ be the solution of the two-fold optimization in (5.12). We denote the realized trajectory up to time t' as $\mathbf{x}_{\text{real}} = (\mathbf{x}_0, \ldots, \mathbf{x}_{t'})$. For the rest of the mission, we predict the future as

$$\begin{aligned} \mathbf{x}_{\text{pred}}(t) &= \mathbf{x}_{\text{real}}(t), && \text{for } t \in [0, t'], \\ \mathbf{x}_{\text{pred}}(t+1) &= \mathbf{x}_{\text{pred}}(t) + \Delta t\, \mathbf{u}_{\text{plan}}(t), && \text{for } t \in [t', H]. \end{aligned} \tag{5.13}$$

In other words, $\mathbf{x}_{\text{pred}}$ provides a combination of observed (past) and predicted (future) states. Accordingly, we use this prediction in the evaluation of overall temporal relaxation $\tau(\mathbf{x}_{\text{pred}}, \Phi, 0)$. The primary event in this section that can trigger the correction/replanning scheme is a potential collision with obstacles, i.e., $\exists t' > t$ such that $\mathbf{x}_{\text{pred}}(t') \cap \{j \subset \mathbb{R} \mid j \in O_{\sigma(k)}\} \neq \emptyset$ where $t_k \leq t \leq t_{k+1}$. We then reactively try to reestablish a safe and acceptable performance in the next section.

Definition 5.6.3 (Decided Tasks)**.** *Given a specification in the form of* $\Phi_{goal} := \bigwedge_i \Phi_{goal,i}$*, if the temporal relaxation of any* $\Phi_{goal,i}$ *can be decided by the realized states* $\mathbf{x}_{real}$ *until* t'*, then* $\Phi_{goal,i}$ *is a decided task. Moreover, the set of decided tasks are defined as* $\mathcal{D} = \{\Phi_{goal,i} \mid \tau(\mathbf{x}_{real}, \Phi_{goal,i}, 0) \in \mathbb{R}\}$*.*

Let a function map the predicted trajectory to satisfied target labels with the order of satisfaction such that $\mathbf{x}_{\text{pred}} \to \mathbf{S}$, e.g., $\mathbf{S} = \{\Phi_3, \Phi_1, \Phi_3, \Phi_2\}$ with a periodic task Φ_3, and $I_{\Phi_i} \subset \mathbb{R}$ be the time interval during when the predicates associated with Φ_i is satisfied. Moreover, let a collision be predicted to happen at t_{obs} such that $\mathbf{x}_{\text{pred}}(t_{\text{obs}}) \cap \{j \subset \mathbb{R} \mid j \in O_{\sigma(k)}\} \neq \emptyset$. The two tasks in Φ_{goal} between which the collision occurs can be obtained by using $s = \arg\min_{s \in [1, |\mathbf{S}|]} |t_{\text{obs}} - \max(I_{\mathbf{S}(s)})| + |\min(I_{\mathbf{S}(s+1)}) - t_{\text{obs}}|$. Then, we do not trigger a collision event until $\mathbf{S}(s) \in \mathcal{D}$ and $\mathbf{S}(s+1) \notin \mathcal{D}$. That is, we consider a collision only when the danger is imminent (see yellow and purple paths in Fig. 5.5).

Trajectory Correction via Control Barrier Functions

Whenever a detected event causes a potential collision, we first try to correct the trajectory locally without further temporal relaxation. If this is not feasible, we do global replanning.

For local correction, we use time-varying CBFs under actuator constraints, ensuring the robot's convergence to the desired target region within a finite time while temporarily moving away from the target if necessary. The target region (next in $\mathbf{S}$) is obtained from φ_i associated with $\Phi_{\text{goal},i}$ as per (5.1).

Definition 5.6.4 (Remaining Time to Next Target)**.** *Whenever a correction is triggered at t_k, we find when the next task region was supposed to be visited by $\mathbf{x}_{pred}$ for the first time, $t_{target} > t_k$. Then the target location is assigned a remaining time $r : \mathbb{Z}_{\geq 0} \to \mathbb{Z}_{\geq 0}$ such that $r(t) = t_{target} - t$ and $r(t_{target}) = 0$.*

We formulate the requirement of reaching the target within the deadline by the following time-varying CBF:

$$\mathfrak{h}(\mathbf{x}, t) := r - \frac{\|\mathbf{x} - \mathbf{x}_{\text{target}}\|}{u_{\max}} \geq 0.$$

Note that the remaining time r strictly decreases, and we want $\|\mathbf{x} - \mathbf{x}_{\text{target}}\| \to 0$ as $r \to 0$. In this regard, we can define a time-varying set $\mathcal{C}_{\text{target}} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathfrak{h}(\mathbf{x}, t) \geq 0\}$ which shrinks as getting closer to the deadline, and the robot must always stay inside $\mathcal{C}_{\text{target}}$. Moreover, we can incorporate collision avoidance into the CBF constraints. As collision avoidance is not time-sensitive, we can use a regular zeroing CBF as $\mathfrak{s}(\mathbf{x}) := \|\mathbf{x} - \mathbf{x}_{\text{obs}}\| \geq 0$ requiring the robot to always reside inside $\mathcal{C}_{\text{safe}} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathfrak{s}(\mathbf{x}) \geq 0\}$.

As the CBF constraints in (2.6) are linear in control for a constant state, if we consider a quadratic running cost, then the problem to be solved reduces to a quadratic program (QP) with explicit input and STL-related CBF constraints to be solved sequentially over the discrete time:

$$\begin{aligned}
&\min_{\boldsymbol{u} \in \mathcal{U}} \ \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u} \\
&s.t. \ \frac{\partial \mathfrak{h}}{\partial \mathbf{x}}^T \boldsymbol{u} + \frac{\partial \mathfrak{h}}{\partial t} + \alpha_{\mathfrak{h}}(\mathfrak{h}) \geq 0, && \text{(5.14a)} \\
&\quad \frac{\partial \mathfrak{s}}{\partial \mathbf{x}}^T \boldsymbol{u} + \alpha_{\mathfrak{s}}(\mathfrak{s}) \geq 0, && \text{(5.14b)}
\end{aligned}$$

where we have $\partial\mathfrak{h}/\partial t = -\partial\mathfrak{h}/\partial r$ since $\partial r/\partial t = -1$.

The feasibility of the CBF correction maneuvers in terms of timely arrival to the target can be verified by simulating those maneuvers. However, due to short time and large or additional obstacles, we may end up having $\mathfrak{h}(\mathbf{x}, t) < 0$ for some $t \geq t_k$.

Although negative zeroing barrier functions have convergence properties [39], since the convergence time is not explicitly guaranteed in such cases, we call the MILP replanning routine as explained next.

Replanning via MILP

When CBF correction fails or is not preferred, we do a global replanning again for a new trajectory for the rest of the mission that minimizes temporal relaxation and maximizes space robustness in temporally relaxed trajectories.

Remark 5.6.1. *Let the already decided tasks at t_k be given as $\mathcal{D}$ (Def. 5.6.3) and the set of all STL tasks as $\mathcal{T}$. We can replace the $\tau(\mathbf{x}, \Phi_{goal}, 0)$ with $\frac{\sum_{i\in\mathcal{T}\setminus\mathcal{D}} \tau(\mathbf{x},\Phi_{goal,i},0)}{|\mathcal{T}\setminus\mathcal{D}|}$ in* (5.11) *as per the definition in* (5.4) *and Proposition 5.6.1.*

In other words, we use $\mathbf{x}_{\text{pred}}$ in (5.13) with the known and unknown trajectory segments without considering the optimization over the already decided tasks for the sake of a reduced computational burden. The time spent for getting new MILP solutions can be represented by an upper bound $\bar{t}_{\text{replan}}$ [34]. In this regard, the trajectory can be modified from (5.13) as $\mathbf{x}_{\text{pred}}(t+1) = \mathbf{x}_{\text{pred}}(t)$, for $t \in [t_k, t_k + \bar{t}_{\text{replan}} - 1]$ to account for this additional expense.

Case Studies

We present numerical simulations to validate our approach in Robotarium [82], showing its efficacy in reacting to unforeseen events by modifying robot trajectories. We compare our results with a more conservative approach without reactive planning.

We consider various environments with several potential rectangular obstacles. Whenever CBF correction is active, the obstacles are outer-approximated via ellipsoids. We use $\gamma_F = \gamma_G = 1$ to bound the relaxations.

To illustrate the benefits of the new temporal relaxation metric, we first considered the following complex scenario: *In addition to the obstacle avoidance requirement expressed as Φ_{safety}, the robot needs to 1) visit region A at least once within* $[20, 30]$*; 2) visit and stay at B and D within* $[76, 86]$ *and* $[100, 105]$*, respectively; and 3) visit and stay at C flexibly for any* 20 *consecutive steps within* $[20, 120]$. The following specification

expresses these requirements:

$$\Phi_{\text{goal}} = F_{[20,30]}\,\text{A} \wedge G_{[76,86]}\,\text{B} \wedge F_{[20,100]}G_{[0,20]}\,\text{C} \wedge G_{[100,105]}\,\text{D},$$

where visiting regions can be captured by the conjunction of linear predicates, e.g., $\text{A} = x \geq 4 \wedge x \leq 8 \wedge y \geq 6 \wedge y \leq 10$.

Predicted and actual trajectories of the robot modified after detecting collisions are presented in Fig. 5.7. In this case, we define three different environment modes with obstacles of various sizes. The nominal plan at $t=0$ has some temporal relaxation in three of the tasks due to actuation limits and obstacles, leading to the relaxed specification of

$$\Phi^{\tau'}_{\text{goal}} = F_{[20,30]}\text{A} \wedge G_{[76,\mathbf{81^*}]}\text{B} \wedge F_{[20,100]}G_{[0,\mathbf{6^*}]}\text{C} \wedge G_{[\mathbf{101^*},105]}\text{D}.$$

At $t = 11$, a new obstacle on the way is detected. However, the robot takes no action until the collision becomes immediate, which happens at $t = 22$ after fulfilling the requirements in A. Due to the size and location of the obstacle, local correction fails, and global replanning yields a new nominal path and the new relaxed specification of

$$\Phi^{\tau}_{\text{goal}} = F_{[20,30]}\,\text{A} \wedge G_{[76,86]}\,\text{B} \wedge F_{[20,100]}G_{[\mathbf{9^*},20]}\,\text{C},$$

removing the task in D completely for the sake of success in the remaining tasks. Different from standard STL metrics, the temporal relaxation metric has the advantage of task removal to avoid significant cascaded delays. This, in fact, improves the satisfaction in B (original interval in Φ) and C (closer to the original interval) compared to the previous plan with $\Phi^{\tau'}$, at the cost of removing the task in D. At $t = 31$, another obstacle is added, which immediately triggers a reaction and is locally avoided (e.g., $t = 58$) without compromising the timely execution of the current plan via local CBF corrections. At $t = 71$, timely rendezvous with the nominal plan is completed, and the robot continues to its plan to satisfy Φ^{τ}. The complete trajectory with marked satisfactory instances and original intervals are presented at $t = 120$.

Comparison of Recovery Strategies: A similar reactive planning problem is solved by using i) the proposed global planning and local correction framework (yellow path in Fig. 5.5), ii) the proposed reactive framework using global MILP planner only (purple path in Fig. 5.5), and iii) a strategy that sticks to the offline plan and only avoids

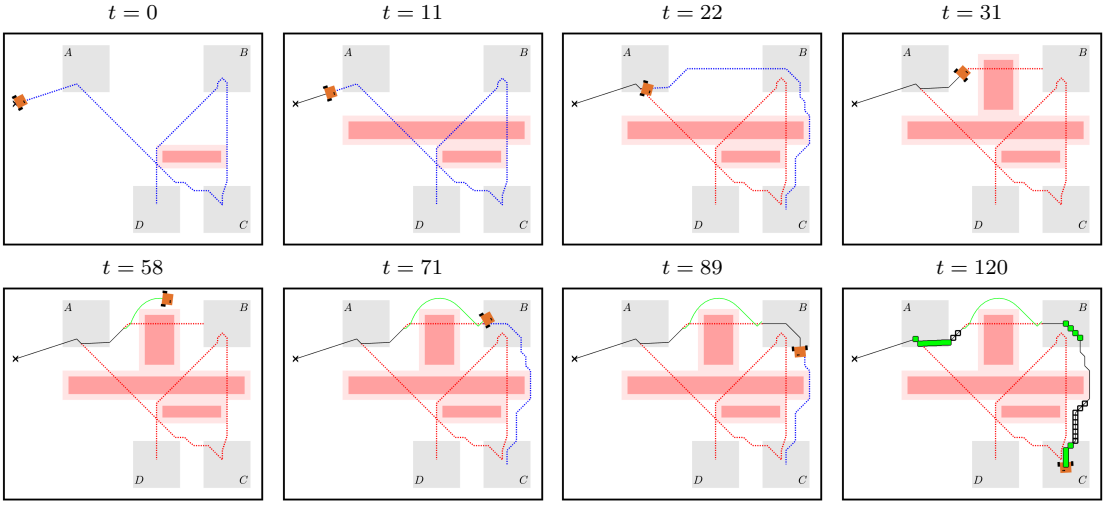


Figure 5.7: Robot progression at various times. The blue dashed path represents the current nominal plan. If the path becomes infeasible, it turns red. The path traversed by the robot is shown partially in black and green, denoting the segments generated by MILP or CBF, respectively. The red regions represent obstacles, with an offset accounting for the robot geometry. The last figure also depicts the (partially) satisfied time intervals of the given STL specification.

Table 5.2: Mean values with 95% confidence intervals of the total of 300 randomized trials. Although temporal relaxation is the primary metric for performance evaluation, how many tasks are compromised on purpose (or inevitably failed for the last strategy) or relaxed is also presented.

Recovery Strategies	Temporal Relax. $\tau(\mathbf{x}, \Phi, 0) \in [0, 1]$	# Relaxed Tasks	# Failed / Removed Tasks	# Triggering Events	# MILP Replannings	# CBF Corrections	MILP Soln. Time $[s]$	CBF Soln. Time $[s]$
Global Plan & Local Correction	0.37±0.04	0.38±0.11	1.22±0.15	1.50±0.15	0.84±0.17	0.66±0.11	1.04±0.16	0.025±0.004
Global Plan & Global Replan	0.24±0.02	0.29±0.09	0.94±0.10	2.14±0.27	2.14±0.27	N/A	1.98±0.60	N/A
Global Plan w/o Reaction	0.46±0.03	0.18±0.08	1.74±0.11	1.52±0.12	N/A	N/A	0.69±0.06	N/A

obstacles without caring about missing deadlines. In addition to the collision avoidance requirement, we use the following specification:

$$\Phi_{\text{goal}'} = F_{[32,42]}\,\mathrm{A} \wedge G_{[77,87]}\,\mathrm{B} \wedge G_{[47,67]}\,\mathrm{D}.$$

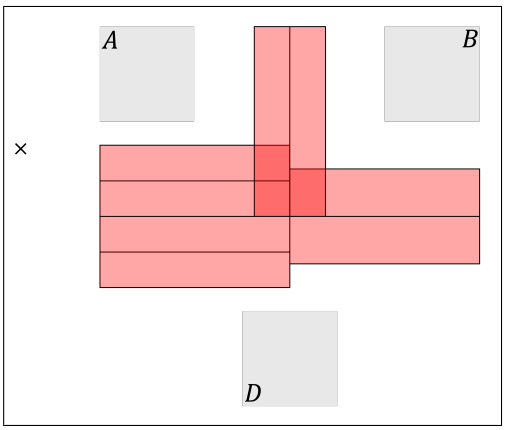


Figure 5.8: Eight potential obstacles that emerge under the specification $\Phi_{\text{goal}'}$.

To see the importance of the reactions clearly, this case involves eight large obstacles with more frequent changes in the environment modes. In particular, the obstacles in Fig. 5.8 are used to build eight environment modes with $pr_{env}^{i,i} = 0.65$ and $pr_{env}^{i,j} = 0.05$ if $i \neq j$, i.e., we expect an environmental change at each time step with 35% chance. Results are given in Table 5.2. While the global plan and replan strategy perform the best, given the relatively high computational cost of solving MILP on the fly, local CBF corrections seem as a viable option. That said, both strategies with reactive planning outperform the conservative "stick-to-the-plan" approach. While the proposed framework removes tasks on purpose, failing tasks in the other case is not optional. That is, the robot still tries to satisfy them but fails. This results in cascaded delays/failure in the remaining tasks.

Although MILP replan is needed after most events in this case due to large obstacles and frequent environmental changes, CBF corrections also play a significant role. Moreover, the local CBF corrections are evidently more efficient in terms of computation time and less likely to undermine the time constraints in the event of reactive planning.

5.7 Discussion

We demonstrate that the proposed formulation for temporal relaxation is computationally efficient and leads to the satisfaction of modified STL specifications by minimally

modifying the time intervals under the allowable relaxation limits. Moreover, we introduce a reactive motion planning framework for robots operating under STL specifications in dynamic environments. Our approach resiliently addresses unforeseen events, such as time-varying obstacles, by implementing runtime monitoring and reaction that dynamically replans or corrects trajectories to minimize temporal violations. Our results show the significance of having sophisticated reactive planning in the face of failure. We also ensure a balance between temporal relaxation and spatial robustness by hierarchical optimization, enhancing both computational efficiency and robust satisfaction, making it a promising solution for resilient robots in complex environments.

Chapter 6

Real-Time Resilient Control Synthesis under STL Specifications

In this chapter, we propose control barrier functions (CBFs) for a family of dynamical systems to satisfy a broad fragment of Signal Temporal Logic (STL) specifications. The specifications may include tasks with nested temporal operators or time-conflicting requirements (e.g., achieving periodic tasks or tasks defined within the same time interval). The proposed CBFs take into account the actuation limits of the dynamical system as well as a feasible sequence of STL tasks. They define time-varying feasible sets of states the system must always stay inside.

We investigate this problem under time-varying tasks. The tasks can be defined in locations changing with time (i.e., dynamic targets), and their future motions are not known a priori. This unpredictability requires an online control approach which motivates us to use control barrier functions (CBFs). We show the feasible sequence generation process that even includes the decomposition of periodic tasks and alternative scenarios due to disjunction operators. The sequence is used to define CBFs, ensuring STL satisfaction.

We show some theoretical results on the correctness of the proposed methods. We illustrate the benefits of our method with CBFs, analyze its performance via simulations

and experiments with drones, and compare it with existing methods.

The chapter is organized as follows: Section 6.1 introduces the relevant literature on CBFs and their use in STL control synthesis. A family of STL specifications that can be tackled with the proposed approach is identified in Sec. 6.2. Next, we state an optimal control problem subject to an STL specification with dynamic targets in Sec. 6.3. Section 6.4 presents our solution with the definition of CBFs over a sequence of STL tasks, how this sequence is generated, and the implementation of the sequential CBFs for STL control synthesis. Then, we discuss the results of the simulations and experiments showing our method's efficacy and real-time applicability in Sec. 6.5. We compare the proposed method with existing tools for stationary targets in Sec. 6.6.

6.1 Introduction and Literature Review

Existing STL control synthesis methods are typically used to solve large-scale optimization problems in an offline fashion for satisfactory robot trajectories. Then, these trajectories are tracked in real time. However, robots often operate in dynamic and/or partially known environments, and they may need to react to unforeseen events during their mission by updating their trajectories. In such cases, existing optimization-based methods may not provide new plans necessarily fast. This gap motivates the researchers to develop more efficient STL synthesis methods with real-time applicability.

In the last decade, there has been significant attention to using control barrier functions (CBFs), which can provide tractable feedback control laws for dynamical systems to ensure staying in "safe sets" of states by avoiding excessive reachability analysis, e.g., [39, 83]. Moreover, maximal control input policies might be used in CBFs to define such sets. Control-dependent CBFs for the sets changing with inputs are considered in [84, 85]. Similarly, the forward invariance (ensuring stay in the safe set forward in time) is established for time-varying sets via CBFs in [40]. Finite-time convergent CBFs [86] are also utilized to reach target sets before a preset deadline. Therefore, besides the safety-critical systems, CBFs can also be used to achieve rich spatial and temporal tasks in a computationally efficient manner.

Recently, there has been a great interest in investigating the use of CBFs to satisfy temporal logic specifications, which typically involve multiple tasks. For example, an

LTL fragment is considered in [87], and a CBF is defined for each task while the order of the task satisfaction is assigned randomly. In [88], tailored time windows are assigned for each task in order to avoid conflicts. While LTL specifications do not include explicit time constraints, specifications with deadlines can be expressed by STL. For instance, the authors of [35] use time-varying CBFs to ensure the satisfaction of a family of STL specifications. In particular, all the unsatisfied tasks are kept active throughout the mission, and the method ensures continuous progress to the satisfaction of all tasks. Instead of activating all CBFs, [89] monitors the tasks in the receding horizon and activates the CBFs accordingly. However, if any two tasks have conflicting requirements (e.g., visiting two distinct regions within overlapping time windows expressed as "go to region A and/or B within the same time window"), these methods become inapplicable as continuous progress towards the satisfaction of such tasks may not be guaranteed. In order to avoid such conflicts, STL tasks at different regions are defined over distinct time windows in [90]. Alternatively, an event-triggered STL framework is proposed in [91] via automata-based planning to activate CBFs, and the user is notified when a conflict occurs. Furthermore, the CBF constraints can be relaxed (e.g., [87,89]), which may yield missing the deadline for one or more relaxed tasks in conflict, and hence the violation of the original specification. In [92], disjunction operators in STL specifications in the context of CBFs are investigated by involving nonsmooth functions, which can also not accommodate time-conflicting requirements and periodic tasks.

To schedule STL tasks with explicit deadlines, one needs to consider the task deadlines and the reachability of task sets by the robot. Task scheduling in the presence of deadlines is broadly investigated for real-time systems via optimal scheduling algorithms such as Earliest-Deadline-First [93] and Least-Laxity-First [94]. A detailed survey on the scheduling of temporally constrained tasks can be found in [95]. Moreover, a Vehicle Routing Problem (VRP) can also be formulated with a mixed-integer program to schedule STL tasks (e.g., [34]).

To the best of our knowledge, none of the existing planning approaches can handle STL specifications changing with time, especially when these changes are unknown a priori. For example, multi-agent settings are considered in some aforementioned works; however, all agents are cooperative and contribute to the satisfaction of coupled specifications [35], or the target dynamics depend on the 'chaser' agent and can be controlled

indirectly [96]. On the other hand, when some of the agents/targets are noncooperative and move independently, these works cannot guarantee continuous satisfaction of the CBF constraints [97]. Hence, the satisfaction of STL specifications defined over targets that move independently and do not communicate their trajectories remains a challenge.

Multi-agent planning with noncooperative targets is typically studied as pursuit-evasion games. This setting generally includes the assumption of targets with the worst-case motion or adopts probabilistic search [98]. Temporal logic can also be used to express such games [99]. Although finite-time capture can be ensured, this generally requires a certain number of pursuer and evader agents [100, 101]. Moreover, pursuit-evasion games consider the reachability-type specifications, which enforce the agent only to reach a target. This is only a special—and relatively simple—case of temporal logic specifications. In this chapter, we aim to utilize the richness of STL specifications and expand the literature on scalable planning with dynamic targets.

6.2 A Family of STL Specifications for Real-Time Control

We specify desired robot behaviors with the following expressive STL syntax:

$$\begin{aligned} \Phi &::= \Phi_1 \wedge \Phi_2 \mid \Phi_1 \vee \Phi_2 \mid F_{[a,b]}\phi \mid G_{[a,b]}\phi \mid \varphi_1 U_{[a,b]}\varphi_2, \\ \phi &::= \varphi \mid \neg\phi \mid F_{[c,d]}\varphi \mid G_{[c,d]}\varphi, \\ \varphi &::= \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2, \end{aligned} \tag{6.1}$$

where Φ, ϕ, φ are STL specifications. Note that the fragment in (6.1) is slightly different than the conventional syntax in (2.1). While it allows for nested temporal operators, temporal operators cannot be in conjunction/disjunction with bare predicates. That is, expressions such as $F_{[a,b]}(\mu_1 \wedge G_{[c,d]}\mu_2)$ are not allowed. Still, the fragment is more expressive than that of many STL CBF literature (e.g., [35], [89]).

The mission horizon H is assumed to be equal to the STL horizon as $H = hrz(\Phi)$.

Remark 6.2.1. *We consider an overall STL specification Φ that expresses a set of tasks Φ_i in conjunction as follows:*

$$\Phi := \Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_k. \tag{6.2}$$

Each task Φ_i consists of temporal operator(s) and a spatial inner task/specification φ_i that may include Boolean connectives in accordance with the syntax in (6.1)*, e.g., $\Phi_i =$*

$F_{[a,b]}\varphi_i$ and $\Phi_j = G_{[c,d]}\big(\varphi_{j,1} \wedge \varphi_{j,2}\big)$ where $\varphi_j = \varphi_{j,1} \wedge \varphi_{j,2}$ is the inner task/specification of Φ_j.

Definition 6.2.1 (Disjunctive Normal Form (DNF) [1])**.** *An inner task φ is said to be in DNF if it has the form of $\bigvee_i \bigwedge_j \mu_{i,j}$ as the disjunction of conjunctions of predicates.*

Without loss of generality, we consider STL tasks Φ_i with inner tasks φ_i in DNF, e.g., $\Phi = \Phi_1 \wedge \Phi_2$ with $\Phi_1 = G_{[a,b]}F_{[c,d]}\varphi_1$ and $\Phi_2 = F_{[0,H]}\big(\varphi_{2,1} \vee (\varphi_{2,2} \wedge \varphi_{2,3})\big)$.

Remark 6.2.2. *Any STL specification can be rewritten in negation-free form [36]. Furthermore, we can rewrite any inner task φ_i in DNF [1].*

For example, consider the STL task $\Phi_i = \neg G_{[0,H]}\big(x \leq 0 \vee (y \leq 0 \wedge z \leq 0)\big)$, requiring the specification of "either x has to be nonpositive or both y and z have to be nonpositive at the same time at all instants within $[0, H]$" not to hold. We denote its inner part as $\varphi_i = x \leq 0 \vee (y \leq 0 \wedge z \leq 0)$, i.e., $\Phi_i = \neg G_{[0,H]}\varphi_i$. This task can be rewritten negation-free and in DNF as $\Phi_i = F_{[0,H]}\big((x \geq 0 \wedge y \geq 0) \vee (x \geq 0 \wedge z \geq 0)\big)$.

6.3 Problem Formulation: STL Control Synthesis with Time-Varying Specifications

We use a family of dynamical systems such that $f(\mathbf{x}) = \mathbf{0_n}$ and $g(\mathbf{x}) = \boldsymbol{I}_n$ where $\mathbf{0_n}$ and $\boldsymbol{I}_n$ denote $n \times 1$ vector of zeros and identity matrix of $n \times n$, respectively. That is, assume holonomic robots with equal controllable and total degrees of freedom, i.e., $n = m$. This renders the system

$$\dot{\mathbf{x}} = \mathbf{u}. \tag{6.3}$$

We also consider the admissible input set as $\mathcal{U} := \{\mathbf{u} \in \mathbb{R}^n \mid \|\mathbf{u}\| \leq u_{\max}\}$, where $u_{\max} \in \mathbb{R}_{\geq 0}$ is the maximum input.The dynamical model in (6.3) is frequently used in the literature for the motion planning of multi-agent systems, and the resulting plans are followed via velocity-based tracking controllers under more complex dynamics (e.g., [91, 102]).

Throughout this chapter, we assume targets with compact geometric shapes. Any compact target shape can be under-approximated via an inner-fit circle (or sphere in 3D). For simplicity, 2D cases are considered hereafter. However, extension to 3D is

straightforward. We denote each target by the couple $\left(\mathbf{x}_i^{target}, rad_i^{target}\right)$ representing the center and radius of the inner-fit circle, respectively, e.g., $\Phi_i = F_{[0,H]}\|\mathbf{x}-\mathbf{x}_i^{target}\| < rad_i^{target}$. We consider the following optimization problem:

Problem 6.3.1 (STL Control Synthesis)**.** *Given a dynamical system* (6.3)*, find an optimal control law,* $\mathbf{u}^*$*, such that the resulting system trajectory* $\mathbf{x}$ *satisfies the given STL specification* Φ *with the minimum control effort, as long as such satisfaction is feasible with the initial state* $\mathbf{x}_0$ *and the control bounds* $\mathbf{u} \in \mathcal{U}$*.*

We now assume that each target i is assumed to move according to $\dot{\mathbf{x}}_i^{target} = \mathbf{u}_i^{target}$ and to obey $\|\mathbf{u}_i^{target}\| \leq u_{i,\max}^{target}$.

Assumption 6.3.1 (Limited Target Information)**.** *While current target positions and velocities, i.e.,* $\mathbf{x}_i^{target}(t)$ *and* $\dot{\mathbf{x}}_i^{target}(t)$*,* $\forall i$*, are available to the robot, the future trajectories of dynamic targets are unknown.*

In other words, we assume that the robot has only the knowledge of instantaneous target positions and velocities. This assumption is not restrictive as the real-time position and velocity estimations of the tracked targets are available via methods such as LiDAR-based sensing [103].

We then formulate another optimal control problem subject to an STL specification as in (6.2) with the syntax in (6.1) defined over dynamic targets. We consider a cost for the system (6.3) as $\mathcal{J}(\mathbf{u}) = \int_0^H \mathbf{u}^T \mathbf{u}\, dt$ yielding to the following problem:

Problem 6.3.2 (STL Control Synthesis with Dynamic Targets)**.** *Consider an environment with no obstacles and some dynamic target areas. Let* Φ *be an STL specification that is defined over the target areas defined by dynamic compact sets with bounded velocities under Assumption 6.3.1. Given a dynamical system as in* (6.3) *and its initial state* $\mathbf{x}_0$*, find an optimal control law,* $\mathbf{u}^*$*, such that the resulting system trajectory* $(\mathbf{x}, 0)$ *satisfies* Φ *with the minimum cost* $\mathcal{J}(\mathbf{u})$*, i.e.,*

$$\begin{aligned} \mathbf{u}^* := \arg\min_{\mathbf{u}\in\mathcal{U}} \ & \mathcal{J}(\mathbf{u}) \\ s.t. \quad & \dot{\mathbf{x}} = \mathbf{u},\ \mathbf{x}(0) = \mathbf{x}_0,\ (\mathbf{x}, 0) \models \Phi, \\ & \dot{\mathbf{x}}_i^{target} = \mathbf{u}_i^{target}, \|\mathbf{u}_i^{target}\| \leq u_{i,\max}^{target},\ \forall i. \end{aligned}$$

Note that $\mathcal{J}(\mathbf{u})$ is quadratic, and CBF constraints in (2.6) are linear in control for a constant state. Therefore, if we could represent the satisfaction of STL specification Φ in the form of (6.2) as CBF constraints, then the Problem 6.3.2 can approximately be represented as multiple quadratic programs (QP) to be solved sequentially over the discrete time.

6.4 Solution Approach: STL Control Synthesis via Control Barrier Functions

We focus on the Problem 6.3.2 as it captures Problem 6.3.1 as well. However, solutions to Problem 6.3.1 will also be considered later on for comparison with existing work.

We propose an approximate solution to Problem 6.3.2, which includes the definition of the worst-case distances to/between target areas moving under Assumption 6.3.1. With this information, a feasible sequence of STL tasks is generated, and a sequential CBF is defined over this sequence to satisfy them. Let us start with the worst-case distance definition between the robot and a dynamic target:

Definition 6.4.1 (Signed Distance [104])**.** *Let $\mathbf{x} \in \mathcal{X}$ be a point in metric space $\mathcal{X}$ and $\mathcal{P} \subseteq \mathcal{X}$ be a compact set in which the inner specification φ defined in* (6.1) *is satisfied, i.e., $\mathcal{P} := \{\mathbf{x} \in \mathcal{X} \mid \mathbf{x} \models \varphi\}$. Then, we define the*

- *Distance from $\mathbf{x}$ to $\mathcal{P}$ as $dist(\mathbf{x}, \mathcal{P}) := \inf_{\mathbf{x}^* \in \mathcal{P}} \|\mathbf{x} - \mathbf{x}^*\|$,*
- *Depth of $\mathbf{x}$ in $\mathcal{P}$ as $depth(\mathbf{x}, \mathcal{P}) := dist(\mathbf{x}, \mathcal{X} \setminus \mathcal{P})$,*
- *Signed Distance from $\mathbf{x}$ to $\mathcal{P}$ to be*

$$Dist(\mathbf{x}, \mathcal{P}) := \begin{cases} -depth(\mathbf{x}, \mathcal{P}), & \text{if } \mathbf{x} \in \mathcal{P}, \\ dist(\mathbf{x}, \mathcal{P}), & \text{otherwise.} \end{cases}$$

In other words, the distance is negative inside $\mathcal{P}$, zero on the boundary, and positive outside. As $\mathcal{P}$ may not be stationary, the worst-case future distance after $r \in \mathbb{R}_{\geq 0}$ time units is also defined considering that $\mathcal{P}$ can move with a maximum velocity of $u_{\mathcal{P},\max}$ for at most

$$Dist_{wc}(\mathbf{x}, \mathcal{P}, r) := Dist(\mathbf{x}, \mathcal{P}) + u_{\mathcal{P},\max} \cdot r, \tag{6.4}$$

in the opposite direction of where the robot is positioned.

It is worth noting that the distance definitions in Def. 6.4.1 do not consider obstacles between the current and target states, as we consider an environment with no obstacles.

Definition 6.4.2 (Remaining Time)**.** *An STL task Φ_i in conjunction with others according to* (6.2)*, is assigned with a remaining time $r_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ such that $r_i(t) := r_i(0) - t$. The robot has to achieve the spatial requirements φ_i associated with Φ_i at or before t' where $r_i(t') = 0$.*

When there is only one task to achieve, i.e., $\Phi = \Phi_i$, the aim is to reach the target area associated with Φ_i, say $\mathcal{P}_i$, before the deadline. As $\mathcal{P}_i$ can move with a maximum velocity of, say $u_{\mathcal{P}_i,\max}$, the robot is required to stay close to $\mathcal{P}_i$ such that it can be reached within the remaining time r_i even if it evades with the maximum velocity during that time. We formulate this requirement by a barrier function[1]

$$\mathfrak{h}_i(\mathbf{x}, t) := r_i - \frac{Dist_{wc}(\mathbf{x}, \mathcal{P}_i, r_i)}{u_{\max}} \geq 0. \tag{6.5}$$

Note that r_i strictly decreases with time. Moreover, as $r_i \to 0$, we want $Dist_{wc}(\mathbf{x}, \mathcal{P}_i, r_i) \to 0$, and hence $Dist(\mathbf{x}, \mathcal{P}_i) \to 0$. In this regard, we can define a time-varying set (shrinking as it gets closer to the deadline), the system must always stay inside by $\mathcal{C}_i := \{\mathbf{x} \in \mathbb{R}^n \mid \mathfrak{h}_i(\mathbf{x}, t) \geq 0\}$.

In the presence of $k > 1$ tasks in conjunction as in (6.2), the system should lie in the intersection of the superlevel sets of barrier functions as $\mathbf{x} \in \mathcal{C}$ where $\mathcal{C} := \{\mathbf{x} \in \mathbb{R}^n \mid \mathfrak{h}_i(\mathbf{x}, t) \geq 0, \forall i \in \{1, \ldots, k\}\}$, i.e., $\mathcal{C} = \bigcap_i \mathcal{C}_i$ as emphasized in relevant literature (e.g., [89, 91]). However, since each feasible set $\mathcal{C}_i$ shrinks individually with time, tasks with conflicting requirements within the overlapping time windows may cause an empty intersection of the feasible sets, and hence infeasibility.

For example, consider the system designated by cross mark in Fig. 6.1 under the STL specification $\Phi = F_{[0,H]}\|\mathbf{x} - \mathbf{x}_{red}\| < rad \wedge F_{[0,H]}\|\mathbf{x} - \mathbf{x}_{blue}\| < rad$, requiring the visit of both red and blue stationary circles within H time units. Two barrier functions associated with each region can be constructed such that ensuring $\mathfrak{h}_{red}(\mathbf{x}, t) \geq 0$ and $\mathfrak{h}_{blue}(\mathbf{x}, t) \geq 0$, $\forall t \geq 0$ implies $t \to H \implies \|\mathbf{x} - \mathbf{x}_{red}\| < rad$ and $\|\mathbf{x} - \mathbf{x}_{blue}\| < rad$. This

[1]The distance function in Def. 6.4.1 may not be differentiable at some point inside the target set $\mathcal{P}$. However, the task removal procedure discussed in the next sections will prevent the system from reaching such a point similar to the switch mechanism in [35].

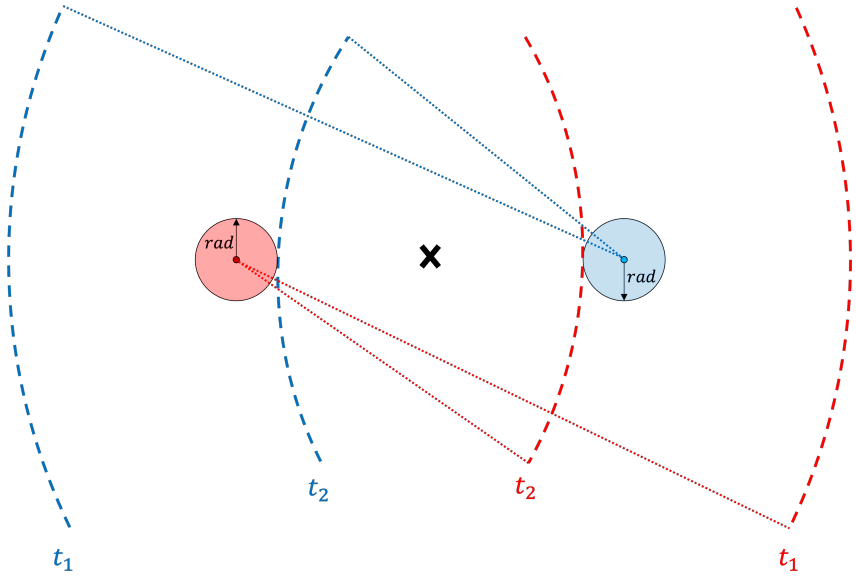


Figure 6.1: An example of conflicting STL tasks with temporally overlapping requirements, $\Phi = F_{[0,H]}\|\mathbf{x} - \mathbf{x}_{red}\| < rad \wedge F_{[0,H]}\|\mathbf{x} - \mathbf{x}_{blue}\| < rad$. The system represented by the cross mark has to stay inside the feasible time-varying sets associated with each target shown by red and blue.

implication is enforced by the feasible time-varying sets depicted for two different time instants, $t_2 > t_1$, via dashed circular arcs in Fig. 6.1, which shrink in time toward the respective targets. However, it is common practice to steer the system only when the border of this shrinking set pushes the system, i.e., waiting until the system absolutely has to move. This may yield the following scenario: At $t = t_1$, it is possible to visit either the red region first and then the blue one or vice versa. On the other hand, at $t = t_1$, none of the borders push the system toward a target. Consequently, at $t = t_2$, when separately concerned, both targets are still reachable. However, the satisfaction of both tasks on time now becomes infeasible as each target is now excluded from the feasible set associated with the other target, i.e., the red (blue) target is now out of the blue (red) arc. On top of this, if the targets are moving, the mission might get infeasible faster. For instance, if the targets move in opposite directions, then the mission would become infeasible even before t_2.

A common way to avoid infeasibilities is the relaxation of CBF constraints which may cause the violation of specifications, particularly the relaxed ones. On the other hand, we tackle the infeasibility problem in the presence of conflicting tasks by enforcing

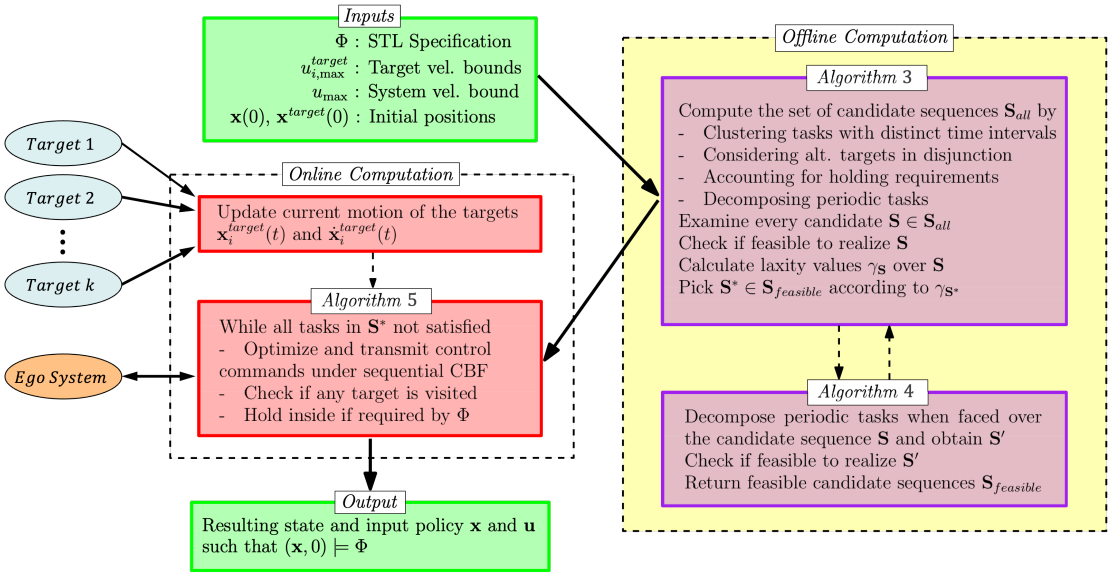


Figure 6.2: Outline of the proposed framework.

stricter feasible sets. In particular, we propose a sequential CBF formulation to ensure that any Φ_i is achieved by taking into account the shrinkage of feasible sets associated with the remaining tasks $\Phi_j, \forall j \in \{1, \ldots, k\} \backslash \{i\}$. Therefore, even if there is more time to achieve it, the system may go for Φ_i immediately in accordance with the remaining task deadlines. In this CBF formulation, we utilize a sequence of STL tasks that is feasible to achieve in a particular order (although it can be disobeyed during execution). A term in the sequence is removed once the associated task is satisfied. The general framework of how to generate a feasible sequence and online control of the system is highlighted in Fig. 6.2. Next, we elaborate more on the construction of a feasible sequence of dynamic STL tasks.

6.4.1 Scheduling STL Tasks

In this section, we will obtain task sequences where the desired STL specification can be achieved by following that particular order of the tasks. The feasibility of such a sequence depends on the maximum velocities of both the controlled robot and the targets, and the task deadlines defined over these dynamic targets. Let us start with the pairwise worst-case distance definitions between two targets:

Definition 6.4.3 (Ordered Set Distance[2])**.** *We define the ordered distance between two compact reachable sets $\mathcal{P}_i$, $\mathcal{P}_j$ as*

$$Dist(\mathcal{P}_i, \mathcal{P}_j) := \sup_{\mathbf{x}\in\mathcal{P}_i} \; dist(\mathbf{x}, \mathcal{P}_j).$$

Similar to (6.4)*, the worst-case pairwise distance considering target motions in r_i and r_j time units is obtained below, assuming both targets move in opposite directions.*

$$Dist_{wc}(\mathcal{P}_i, \mathcal{P}_j, r_i, r_j) := Dist(\mathcal{P}_i, \mathcal{P}_j) + u_{i,\max}^{\mathcal{P}_i} \cdot r_i + u_{j,\max}^{\mathcal{P}_j} \cdot r_j. \tag{6.6}$$

Intuitively, the worst-case scenario is considered in the definition of the set distance such that the system starts at the farthermost point in $\mathcal{P}_i$ as if it moves for r_i time units, and travels to the farthermost point in $\mathcal{P}_j$ after its motion of r_j time units. This can be interpreted as reaching $\mathcal{P}_i$ as if it travels with its maximum velocity along the direction opposite to $\mathcal{P}_j$, which itself moves with its maximum velocity.

For notational simplicity, by using the Defs. 6.4.1 and 6.4.3 together with the input bound $u_{\max}$ and with a slight abuse of notation, we define two variables, namely

$$\mathfrak{d}_i := \frac{Dist_{wc}(\mathbf{x}, \mathcal{P}_i, r_i)}{u_{\max}} \text{ and } \mathfrak{d}_{i,j} := \frac{Dist_{wc}(\mathcal{P}_i, \mathcal{P}_j, r_i, r_j)}{u_{\max}}, \tag{6.7}$$

as the required time for the system to reach the i^{th} dynamic target under the maximum input and as the worst-case minimum travel time between i^{th} and j^{th} targets, respectively.

Definition 6.4.4 (Task Sequence and Its Subsequences)**.** *Given an STL specification with k tasks as in* (6.2)*, the task sequence $\mathbf{S}$ is an order of the task indices $\{1, \ldots, k\}$. For example, consider $\Phi := \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge \Phi_4$, then $\mathbf{S} = (3, 1, 2, 4)$ is a task sequence for Φ. Each term in the sequence $\mathbf{S}(i)$ has a corresponding compact and dynamic target set $\mathcal{P}_{\mathbf{S}(i)}$ in which the inner part $\varphi_{\mathbf{S}(i)}$ of the task $\Phi_{\mathbf{S}(i)}$ is satisfied. A subsequence $\mathbf{S}'$ is obtained by removing one or more terms of $\mathbf{S}$ after the associated target set(s) are visited where $1 \leq |\mathbf{S}'| \leq |\mathbf{S}|$, and $|\mathbf{S}'| = |\mathbf{S}|$ implies $\mathbf{S}' = \mathbf{S}$. For example, $(1, 2, 4)$, $(3, 1, 2)$, and $(3, 2)$ are some subsequences of $\mathbf{S}$.*

Note that the mapping between the tasks $\{1, \ldots, k\}$ and their indices in a sequence $\mathbf{S}$ is onto but not injective as the same periodic task may appear more than once in the sequence, i.e., $|\mathbf{S}| \geq k$.

[2]This metric is a form of Hausdorff distance with a particular order.

Definition 6.4.5 (Feasibility of an STL Task and a Sequence)**.** *The i^{th} STL task $\Phi_{\mathbf{S}(i)}$ among a task sequence $\mathbf{S}$ with a deadline $r_{\mathbf{S}(i)}$ is said to be feasible with respect to the dynamics in* (6.3) *if $r_{\mathbf{S}(i)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}$ for $i \geq 2$ and $r_{\mathbf{S}(1)} \geq \mathfrak{d}_{\mathbf{S}(1)}$ for the first task in the sequence. Moreover, if a sequence is comprised of only feasible tasks, then it is a feasible sequence.*

Let us have a feasible sequence of tasks $\mathbf{S}$ according to Def. 6.4.5. We next show that the accomplishment of STL tasks in the given order in $\mathbf{S}$ (i.e., satisfying the first task in the order next) preserves the feasibility of the remaining tasks. Moreover, we also show that even if the order in $\mathbf{S}$ is violated during the execution (e.g., satisfying a later task in the sequence first and removing it), the resulting subsequence will still be feasible.

Lemma 6.4.1 (Conservation of Sequence Feasibility)**.** *Consider an STL specification $\Phi := \Phi_1 \wedge \ldots \wedge \Phi_k$ and a sequence of its tasks $\mathbf{S}$ as in Def. 6.4.4 together with the duration definitions in* (6.7)*. If $r_{\mathbf{S}(i)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)},\ \forall i \in \mathbf{S}$, then $r_{\mathbf{S}'(j)} \geq \mathfrak{d}_{\mathbf{S}'(1)} + \sum_{l=2}^{j} \mathfrak{d}_{\mathbf{S}'(l-1),\mathbf{S}'(l)},\ \forall j \in \mathbf{S}'$, for any subsequence $\mathbf{S}'$.*

Proof. To prove the continuous feasibility of the sequence $\mathbf{S}$, we investigate different cases with regard to the locations of the achieved and removed task(s) among the sequence:

Case 1: (Only the first term in the sequence, $\mathfrak{d}_{\mathbf{S}(1)}$, is removed, i.e., $\mathbf{S}' = \mathbf{S} \setminus \{\mathbf{S}(1)\}$.) The term $\mathfrak{d}_{\mathbf{S}(1)}$ is removed when the set of states associated with it, $\mathcal{P}_{\mathbf{S}(1)}$, is visited. We want to show that after the removal of $\mathbf{S}(1)$, $r_{\mathbf{S}(i)} \geq \mathfrak{d}_{\mathbf{S}(2)} + \sum_{l=3}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}, \forall i \in \mathbf{S} \setminus \{\mathbf{S}(1)\}$ holds. It is given in the premise that $r_{\mathbf{S}(i)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}, \forall i \in \mathbf{S}$, then we have $r_{\mathbf{S}(i)} \geq \sum_{l=2}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)} = \mathfrak{d}_{\mathbf{S}(1),\mathbf{S}(2)} + \sum_{l=3}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}, \forall i \in \mathbf{S}$. Moreover, by the Defs. 6.4.1 and 6.4.3, we know $\mathfrak{d}_{\mathbf{S}(1),\mathbf{S}(2)} \geq \mathfrak{d}_{\mathbf{S}(2)} = Dist_{wc}(\mathbf{x}, \mathcal{P}_{\mathbf{S}(2)}, r_{\mathbf{S}(2)})/u_{\max}, \forall \mathbf{x} \in \mathcal{P}_{\mathbf{S}(1)}$. Hence, the condition in the premise implies $r_{\mathbf{S}(i)} \geq \mathfrak{d}_{\mathbf{S}(2)} + \sum_{l=3}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}, \forall i \in \mathbf{S} \setminus \{\mathbf{S}(1)\} = \mathbf{S}'$.

Case 2: (Later tasks in the sequence are removed.) Let $\mathbf{S}'_{-1}$ be any subsequence of $\mathbf{S} \setminus \{\mathbf{S}(1)\}$ and $\mathbf{S}''_{-1}$ be any subsequence of $\mathbf{S}'_{-1}$. Then it follows from the triangle inequality that $\sum_{l=2}^{|\mathbf{S}'_{-1}|} \mathfrak{d}_{\mathbf{S}'_{-1}(l-1),\mathbf{S}'_{-1}(l)} \geq \sum_{l=2}^{|\mathbf{S}''_{-1}|} \mathfrak{d}_{\mathbf{S}''_{-1}(l-1),\mathbf{S}''_{-1}(l)}$, which implies $\mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{|\mathbf{S}'_{-1}|} \mathfrak{d}_{\mathbf{S}'_{-1}(l-1),\mathbf{S}'_{-1}(l)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{|\mathbf{S}''_{-1}|} \mathfrak{d}_{\mathbf{S}''_{-1}(l-1),\mathbf{S}''_{-1}(l)}$. Since $\mathbf{S}'_{-1}$ and $\mathbf{S}''_{-1}$ are arbitrary, the conditional statement in the premise holds.

Case 3: (The first and later task(s) are removed together.) It follows from Cases 1 and 2 that any subsequence obtained by removing $\mathbf{S}(1)$ and one or more later tasks in $\mathbf{S}$ together preserves the inequality in the premise. □

We first create all possible orders of tasks $\mathbf{S}_{all}$ and obtain the feasible sequences $\mathbf{S}_{feasible}$ offline by filtering according to i) the possible scheduling conflicts (e.g., time interval inconsistencies) of STL tasks and ii) the feasibility of the candidate sequences according to the ability to travel the worst-case distances before deadlines. When STL tasks are in disjunction (i.e., defined over redundant targets), we obtain sequences for each alternative scenario. Moreover, when periodic tasks are present, the sequence generation procedure also involves the decomposition of a periodic task to multiple fixed-interval ones.

Removing Time Interval Inconsistencies

Let two tasks be given as Φ_i and Φ_j with the time intervals bounded by $I_i = [a_i, b_i]$ and $I_j = [a_j, b_j]$, respectively. Then we apply the following filter when creating the set of all candidate but not necessarily feasible sequences $\mathbf{S}_{all}$:

$$b_j < a_i \implies \mathbf{S}_{i,j} \notin \mathbf{S}_{all}, \tag{6.8}$$

where $\mathbf{S}_{i,j} := \{\mathbf{S} \mid \mathbf{S}(x) = i,\ \mathbf{S}(y) = j,\ x < y\}$. This rule is used to cluster the tasks with distinct time intervals so that the complexity of candidate sequence generation can be reduced. For instance, let a specification be given as $\Phi = \Phi_1 \wedge \Phi_2$ with $\Phi_1 = F_{[20,40]}\varphi_1$ and $\Phi_2 = F_{[0,10]}\varphi_2$. It is obvious that the sequence $(2, 1)$ is the only possible one given the intervals of the first and second tasks.

Handling Disjunctions During Sequence Generation

Let us denote the single STL task in (6.2) that comprises k_d other tasks in disjunction by Φ_d where $d \in \{1, \ldots, k\}$. Suppose that each STL task Φ_j in (6.2) with $j \in \{1, \ldots, k\} \backslash \{d\}$ is defined over q_j number of alternative targets. Then we consider all $\prod_{i \in \{1,\ldots,k\}} q_i$ possible STL tasks when determining candidate sequences $\mathbf{S}_{all}$ where $q_d = \sum_{l=1}^{k_d} q_{d_l}$ with q_{d_l} representing the number of redundant target areas for each STL task l in disjunction with other STL tasks.

For example, consider $\Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3$ with $\Phi_1 = F_{[a,b]}(\mu_1 \wedge \mu_2)$, $\Phi_2 = G_{[c,d]}\mu_3$, and $\Phi_3 = \Phi_{3,1} \vee \Phi_{3,2}$ where $\Phi_{3,1} = F_{[e,f]}(\mu_4 \vee \mu_5)$ and $\Phi_{3,2} = G_{[g,h]}\mu_6$. We have $d = 3$ and $\prod_i q_i = 2 \times 1 \times (2+1) = 6$ alternative sets of STL tasks to start with where $q_1 = 2$, $q_2 = 1$, and $q_3 = q_{3,1} + q_{3,2} = 2 + 1 = 3$.

Reflecting the Hold Requirements in Sequence Calculations

The STL fragment given in (6.1) accommodates STL tasks that require to be continuously satisfied for a specific length of time once started, e.g., $\Phi_{hold\,1} = G_{[a,b]}\varphi$ and $\Phi_{hold\,2} = F_{[a,b]}G_{[c,d]}\varphi$. For this reason, when checking the feasibility of a sequence, it is required to consider the amount of time the robot has to spend while achieving such a task in addition to the time required to reach the associated target area. Furthermore, as the targets are dynamic, the robot may need to move attached to them during the holding. Since this additional motion can be in the opposite direction of where the robot needs to be heading next (if any task remains), we have to reflect this possible additional time to get into the course in the worst-case travel duration calculations (i.e., (6.7)) from targets associated with such tasks.

Let Φ_j be any task in the sequence and Φ_{hold} be a task requiring a hold. If we denote the specified duration of the hold by I_{hold}, e.g., $I_{hold\,1} = b - a$ and $I_{hold\,2} = d - c$, then we update the worst-case pairwise travel duration from the target associated with Φ_{hold} to the one that of Φ_j as:

$$\begin{aligned} Dist_{wc}(\mathcal{P}_{hold}, \mathcal{P}_j, r_{hold}, r_j) := & Dist(\mathcal{P}_{hold}, \mathcal{P}_j) \\ & + u_{hold,\max}^{\mathcal{P}_{hold}} \cdot r_{hold} + u_{j,\max}^{\mathcal{P}_j} \cdot r_j \\ & + u_{hold,\max}^{\mathcal{P}_{hold}} \cdot I_{hold} + u_{\max} \cdot I_{hold}, \end{aligned} \tag{6.9}$$

where the first three terms stand for the conventional worst-case distance between the current and the next target areas as in (6.6); the last two terms account for the worst-case distance traveled attached to the target while holding, and the time needs to be spent while holding, respectively.

Decomposition of Periodic Tasks

Any periodic STL task, e.g., $\Phi_{periodic} = G_{[a,b]}F_{[c,d]}\varphi$, can be decomposed as:

$$\begin{aligned} \Phi_{periodic} = G_{[a,b]}F_{[c,d]}\varphi = \bigwedge_{i=0}^{k_\tau - 1} F_{[t_i, t_i+\tau]}\,\varphi \\ s.t. \quad t_0 \in [a+c, a+d] \\ t_i \in [t_{i-1}, \max(t_{i-1}+\tau, b+d)], \forall i \in [1, k_\tau], \end{aligned} \tag{6.10}$$

where t_i denotes each time instant at which φ is satisfied with $t_{k_\tau} \in [b+c, b+d]$; $\tau = d-c$ is the time period of the task $\Phi_{periodic}$. The number of decomposed *finally* tasks, k_τ in (6.10) can be finitely many depending on the realized trajectory and the sequence of the tasks achieved.

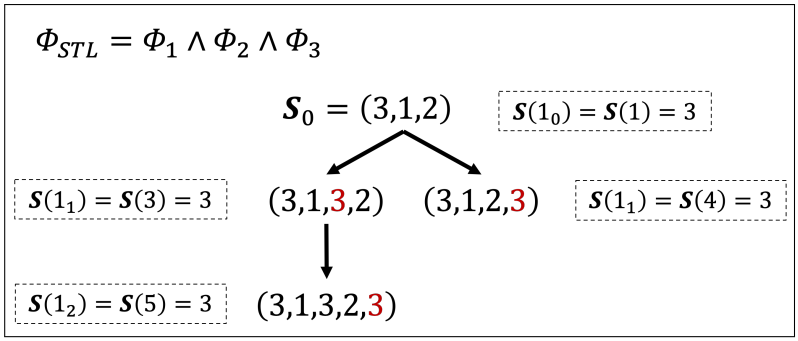


Figure 6.3: Derivation of new sequences from a candidate, $\mathbf{S}_0$, for the specification $\Phi = \Phi_1 \wedge \Phi_2 \wedge \Phi_3$ by decomposing the periodic task Φ_3 recursively.

When inserting the decomposed finally tasks among a candidate sequence $\mathbf{S}$, the representation in (6.10) may be redundant as multiple same finally tasks could appear between two other tasks in the sequence. To prevent the additional complexity this brings, satisfaction times t_i in the above decomposition can be bounded by considering the previous and next tasks for each possible insertion of the periodic task in a given candidate sequence. In other words, instead of a blind decomposition, we consider the travel times to/from the periodic task areas within a given sequence while inserting successive periodic tasks. In this regard, a task will appear only once between two other tasks in the resulting sequence, which are feasible to travel to/from based on the worst-case relative distances and their deadlines. We obtain such representation by

decomposing the periodic tasks as:

$$\Phi = G_{[a,b]}F_{[c,d]}\varphi = \bigwedge_i F_{[e_i,l_i]}\ \varphi, \tag{6.11}$$

where $e_i, l_i \in [a+c, b+d], \forall i$ are the earliest and latest possible satisfaction instants depending on the tasks in the sequence (if any) before and next to the periodic task, respectively. In order to calculate the earliest and latest possible satisfaction instants, we do the following: We denote the order of a task in the sequence by s. In accordance with (6.11), let us use $i \geq 0$ to number the appearances of the same periodic task $\mathbf{S}(s_i)$ in the sequence with $s_0 = s$. For instance, let $\mathbf{S} = (3, 1, 3, 2)$ with Φ_3 being a periodic task, then we define $\mathbf{S}(1_0) = \mathbf{S}(1) = 3$ and $\mathbf{S}(1_1) = \mathbf{S}(3) = 3$ as depicted in the left branch rooting from $\mathbf{S}_0$ in Fig. 6.3. We determine the bounds on the satisfaction instants as follows:

$$\begin{aligned} e_i &:= t_{prev} + \sum_{j=s_{i-1}}^{s_i-1} \mathfrak{d}_{\mathbf{S}(j),\mathbf{S}(j+1)}, && \textit{for } i > 0, \\ l_i &:= \min_{z\in\{s_i+1,\ldots,|\mathbf{S}|-1\}} r_{\mathbf{S}(z)} - \sum_{j=s_i}^{z-1} \mathfrak{d}_{\mathbf{S}(j),\mathbf{S}(j+1)}, && \textit{for } i \geq 0, \end{aligned} \tag{6.12}$$

where $t_{prev} \in \{e_{i-1}, l_{i-1}\}$ is the last time the periodic task is achieved with $e_0 := \max\left(a+c, \mathfrak{d}_{S(1)} + \sum_{j=1}^{s_0-1} \mathfrak{d}_{\mathbf{S}(j),\mathbf{S}(j+1)}\right)$.

Note that different s_i alternatives exist depending on where s_0 is placed and how many other tasks are completed between two consecutive visits to the periodic target. Hence, we have to examine all such possibilities. However, we need the following assumption to keep the possibilities finite.

Assumption 6.4.1 (Distinct Periodic Targets)**.** *Any two target areas $\mathcal{P}_i$ and $\mathcal{P}_j$ associated with any two periodic tasks Φ_i and Φ_j have empty intersection, i.e., $\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$.*

This assumption prevents the algorithm from generating infinitely many possible sequences as some time has to pass while traveling between two target areas with no intersection, and we have finite time bounds.

Task Deadlines among a Sequence

We consider the satisfaction of STL tasks over dynamic targets in a particular order, i.e., in a sequence. However, assuming that the targets will be reached exactly at their

associated deadlines after moving with their maximum velocities until then may be unrealistic and utterly conservative. This is because the future tasks in the sequence can have earlier deadlines since we consider all possible task orders in the candidate sequences. In other words, a given sequence may enforce the achievement of a task way before its deadline; therefore, in the worst-case distance calculations in (6.7), considering only the associated deadline may be conservative and render a set of specifications infeasible that would otherwise be feasible. For instance, suppose $\Phi = \Phi_1 \wedge \Phi_2$ with $r_1 = 20$ and $r_2 = 10$, if the candidate sequence is $\mathbf{S} = \{1, 2\}$, then it should not be possible for the target associated with Φ_1 to move 20 time units before its achievement, since Φ_2 should be satisfied within 10 time units after Φ_1 is accomplished. To account for such cases, we incorporate the remaining times of future tasks in the sequence besides the original deadlines.

Definition 6.4.6 (Sequential Remaining Time Bounds)**.** *For a feasible STL task sequence* $\mathbf{S}$*, the actual remaining time to achieve an STL task* $\Phi_{\mathbf{S}(i)}$ *in the sequence is bounded by the deadlines of the remaining tasks in the sequence as follows:*

$$\overline{r}_{\mathbf{S}(i)} := \min_{j \in [i, \ldots, |\mathbf{S}|]} r_{\mathbf{S}(j)}. \tag{6.13}$$

We next show that using the remaining time bounds based on (6.13) instead of actual task deadlines does not affect the feasibility of the sequence.

Proposition 6.4.1 (Feasibility of Sequential Remaining Time Bounds)**.** *Let a sequence of* k *STL tasks defined according to Def. 6.4.4 be denoted as* $\mathbf{S}$*. If* $\mathbf{S}$ *is feasible per Def. 6.4.5, then replacing the remaining times* $r_{\mathbf{S}(i)}$ *(Def. 6.4.2), with the sequential remaining time bounds* $\overline{r}_{\mathbf{S}(i)}$ *in Def. 6.4.6 for each task* $i \in \{1, \ldots, k\}$ *preserves the sequence feasibility.*

Proof. Per Def. 6.4.5, the feasibility of i^{th} task in the sequence implies $r_{\mathbf{S}(i)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}$. For another task in the j^{th} order such that $i \leq j \leq |\mathbf{S}|$, we also have $r_{\mathbf{S}(j)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)} + \sum_{l=i+1}^{j} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}$ as $\mathbf{S}$ is a feasible sequence. Therefore, we obtain $r_{\mathbf{S}(j)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}$. As j is arbitrary, replacing $r_{\mathbf{S}(i)}$ with any of the $r_{\mathbf{S}(j)}$ for $j \in [i, \ldots, |\mathbf{S}|]$, particularly with the result of (6.13) leads $\overline{r}_{\mathbf{S}(i)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{i} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}$. Thus the proposition follows since i is arbitrary. □

For example, consider a specification $\Phi = F_{[0,30]}\varphi_1 \wedge F_{[0,40]}\varphi_2$ with the committed sequence of $(2, 1)$. In this case, as the second task φ_2 needs to be satisfied first, it becomes redundant to assume that the target associated with it may move for 40 time units, even for the worst case. This is because, the satisfaction of φ_2 should occur before the deadline of the φ_1, $r_1 = 30$, as well as before its own deadline, $r_2 = 40$. Therefore, if the sequence of $(2, 1)$ is committed to, it is safe to continue calculations according to $\Phi = F_{[0,30]}\varphi_1 \wedge F_{[0,30]}\varphi_2$ via (6.13). Actually, the satisfaction of φ_2 should take place much earlier so that there is enough time to satisfy φ_1 before $t = 30$. This requirement will be enforced by sequential CBFs proposed in Sec. 6.4.3.

6.4.2 Offline Task Sequence Generation

We start with a candidate sequence $\mathbf{S}_0 \in \mathbf{S}_{all}$ where any periodic task is placed according to its initial satisfaction requirement, i.e., only s_0 elements are placed for each periodic task. For instance, $G_{[a,b]}F_{[c,d]}\varphi$ is initially inserted in $\mathbf{S}_0$ as $F_{[a+c,a+d]}\varphi$. We implement a periodic task decomposition procedure for each candidate sequence in Algs. 3 and 4. This procedure evaluates possible spots to insert periodic tasks with the earliest (or latest) possible times of satisfaction within the interval associated with that spot, e_i (or l_i).

For a given STL specification Φ, Alg. 3 examines each candidate sequence by checking the feasibility of reaching the dynamic targets in that particular order before the associated deadlines. The initial deadline for each task is determined in line 1. The STL tasks with distinct time intervals are clustered according to (6.8), and possible permutations of candidate task sequences considering all possible disjunction scenarios (see Sec. 6.4.1) are found in line 2.

Feasibility check is performed recursively throughout each candidate sequence in lines 4-22. The check is conducted based on the sequence-specific deadlines (see Sec. 6.4.1) and laxity values defined for each task within the sequence.

Definition 6.4.7 (Laxity Value of an STL Task)**.** *The difference between the remaining time to achieve an STL task placed in the sequence* $\mathbf{S}$ *and the worst-case duration to reach the dynamic target associated with the task from* $\mathbf{x}(0)$ *is the laxity value of that*

Algorithm 3: Feasible Sequence Generator and Selector with Periodic Tasks

input : STL specification $\Phi = \Phi_1 \wedge \ldots \wedge \Phi_k$;
Velocity bounds, initial positions, and target radii

Determine deadlines r_i according to Φ for $i \in \{1,\ldots,k\}$;
Cluster the STL tasks with distinct intervals using (6.8) and find the set of all possible sequences $\mathbf{S}_{all}$ by incorporating all disjunction scenarios as in Sec. 6.4.1;
$\mathbf{S}_{feasible} = \emptyset$, $\gamma_{feasible} = \emptyset$;
for *Each candidate sequence* $\mathbf{S}_0 \in \mathbf{S}_{all}$ **do**
    $\mathbf{S} = \mathbf{S}_0$, $s = 0$;
    Determine realistic deadlines $\overline{r}_i$ (Def. 6.4.6) and worst-case travel times over $\mathbf{S}$ via (6.7);
    while $|\mathbf{S}| > s$ **do**
        $s = s + 1$;
        if $\gamma_{\mathbf{S}(s)} < 0$ **then** break, switch sequences;
        if $\mathbf{S}(s)$ *is a periodic task with* $s < |\mathbf{S}|$ **then**
            Determine e and l according to (6.12);
            if $e \leq l$ *and* $e + \tau_{\mathbf{S}(s)} \leq hrz(\Phi)$ **then**
                $\overline{\mathbf{S}} = BranchOut(\mathbf{S}, e, l)$;
                $\mathbf{S}_{feasible} = \mathbf{S}_{feasible} \cup \overline{\mathbf{S}}$;
                Break, switch sequences;
            else if $e > l$ **then**
                Break, switch sequences;
            end
        end
    end
    $\mathbf{S}_{feasible} = \mathbf{S}_{feasible} \cup \{\mathbf{S}\}$;
end
Pick the best $\mathbf{S}^* \in \mathbf{S}_{feasible}$ according to $\gamma_{\mathbf{S}^*}$;

return : Feasible sequence $\mathbf{S}^*$

task with respect to the system (6.3)*, i.e.,*

$$\gamma_{\mathbf{S}(s)} := \overline{r}_{\mathbf{S}(s)} - \mathfrak{d}_{\mathbf{S}(1)} - \sum_{i=1}^{s-1} \mathfrak{d}_{\mathbf{S}(i),\mathbf{S}(i+1)}. \tag{6.14}$$

In other words, laxity value measures how early each task s in the sequence $\mathbf{S}$ can be satisfied before its deadline. For a sequence to be feasible according to Def. 6.4.5, every task in the sequence must have a nonnegative laxity value.

Algorithm 4: *BranchOut*: Decomposing a Periodic Task among the Candidate Sequence

input : Candidate Sequence $\mathbf{S}$, Earliest and Latest satisfaction times e, l for the periodic task $\mathbf{S}(s)$

1 $\#Branch = 1$, $j = s$, $\mathbf{S}_{feasible} = \emptyset$, $\gamma_{feasible} = \emptyset$;
2 **for** t_{prev}=$\{e,l\}$ **do**
3 **while** $\sum_{i=s}^{j} \mathfrak{d}_{\mathbf{S}(i),\mathbf{S}(i+1)} + \mathfrak{d}_{\mathbf{S}(j+1),\mathbf{S}(s)} \leq \tau_{\mathbf{S}(s)}$ **do**
4 $\mathbf{S}^{\#Branch} = InsertBetween(\mathbf{S}(j+1), \mathbf{S}(j+2))$;
5 $r_{\mathbf{S}^{\#Branch}(j+2)} = t_{prev} + \tau_{\mathbf{S}(s)}$
6 $\#Branch = \#Branch + 1$, $j = j + 1$;
7 Increase $\mathfrak{d}_{\mathbf{S}(j+1),\mathbf{S}(s)}$ and $\mathfrak{d}_{\mathbf{S}(s),\mathbf{S}(j+2)}$ by $\mathfrak{d}_\delta$ (6.15);
8 **end**
9 **for** *all* $\mathbf{S}' \in \mathbf{S}^{\#Branch}$ **do**
10 $s' = s$;
11 **while** $|\mathbf{S}'| > s'$ **do**
12 $s' = s' + 1$;
13 $\gamma_{\mathbf{S}'(s')} = \overline{r}_{\mathbf{S}'(s')} - t_{prev} - \sum_{i=s}^{s'-1} \mathfrak{d}_{\mathbf{S}'(i),\mathbf{S}'(i+1)}$;
14 Repeat the lines 9-19 of Alg. 3 for $\mathbf{S}'(s')$;
15 **end**
16 $\mathbf{S}_{feasible} = \mathbf{S}_{feasible} \cup \{\mathbf{S}'\}$;
17 **end**
18 **end**

return : $\mathbf{S}_{feasible}$

Whenever the next task is periodic during the feasibility check, the decomposition process takes place starting line 10 to schedule possible future visits to the triggering periodic task. First, the earliest and latest time instants to leave the periodic task area considering the adjacent tasks in the sequence are found in line 11 according to (6.12). As several potential sequences can be derived from $\mathbf{S}_0$, Alg. 4 is called in line 13, which branches out these sequences from $\mathbf{S}_0$. The feasibility checking process continues within the new possible branches. The Alg. 4 will continue to call itself with an expanding tree logic via depth-first search whenever the feasibility check reaches new periodic tasks. When the expansion is concluded, the recursively found set of feasible sequences $\overline{\mathbf{S}}$ will be returned to Alg. 3 in line 13. The newly generated feasible sequences are added to the set of all feasible sequences $\mathbf{S}_{feasible}$ in line 14. The operator $\cup$ is used to denote such a union of sets of sequences with a slight abuse of notation. On the other hand, if there is no periodic task in Φ, Alg. 4 will never be triggered, and a choice will be made in line 23 among the feasible candidate sequences from line 21, based on laxity values.

When triggered by a periodic task, Alg. 4 creates an implicit tree rooting from a previous sequence (initially $\mathbf{S}_0$) with possible decomposition spots for that task via depth-first search. According to the period of the triggering task, it derives new sequences at which the next satisfaction of the same periodic task is inserted at different possible locations in lines 3-8, see the first two branches in Fig. 6.3 for example.

Consider a periodic task $\mathbf{S}(j)$ in the j^{th} order of the sequence. When examining possible spots for a successive decomposition of $\mathbf{S}(j)$, one must consider the additional worst-case distance to/from $\mathbf{S}(j)$ from/to the dynamic targets associated with the tasks that the decomposition is made between. For instance, if the decomposition is made within the first couple of next tasks, i.e., $\mathbf{S}(j+1)$ and $\mathbf{S}(j+2)$, an additional worst-case travel time from/to these tasks, which are immediately before and after the decomposition spot, has to be considered. This addition is due to the motion of the target area associated with $\mathbf{S}(j)$ during the length of the period. After the successive insertion of $\mathbf{S}(j)$ between $\mathbf{S}(j+1)$ and $\mathbf{S}(j+2)$, the next decomposition of $\mathbf{S}(j)$ will be the $(j+2)^{th}$ task in the updated sequence, and the original $\mathbf{S}(j+2)$ will shift to $(j+3)^{th}$ spot. The additional worst-case travel duration to/from the new decomposition of the periodic

task $\mathbf{S}(s)$ from/to the new adjacent tasks in the sequence is given by

$$\mathfrak{d}_\delta := \frac{u^{target}_{\mathbf{S}(s),\max} \cdot \tau_{\mathbf{S}(s)}}{u_{\max}}, \tag{6.15}$$

which accounts for the additional distance traveled by the periodic target, i.e., $u^{target}_{\mathbf{S}(s),\max} \cdot \tau_{\mathbf{S}(s)}$, and how long it would take to compensate for it. Between lines 9-17, the feasibility of the possible insertion scenarios is checked similarly to Alg. 3, considering the worst-case distances updated via (6.15) in line 7 and based on the earliest and latest possible times to satisfy the triggering periodic task. During this check, whenever the next task is periodic, the branching out mechanism is triggered again, and new sequences are generated by the decomposition of the new triggering periodic task recursively.

After completing the decomposition process, a sequence is picked among the feasible alternatives in line 23 of the Alg. 3. Next, we define the following criteria to choose the sequence with the greatest least laxity among its tasks:

$$\mathbf{S}^* := \underset{\mathbf{S}\in\mathbf{S}_{feasible}}{\arg\max} \min_{i\in\mathbf{S}} \gamma_{\mathbf{S}(i)}. \tag{6.16}$$

This selection can also be made based on some other user-defined criteria with different performance metric definitions. For instance, it is also possible to choose the sequence that cumulatively has the highest laxity among the tasks in the sequence instead of the criteria in (6.16). We next show that decomposing the periodic task by inserting it into different spots along the sequence does not affect the outcome of the critical (least) laxity associated with that sequence.

Proposition 6.4.2 (Invariance of Critical Laxity)**.** *Suppose a feasible sequence* $\mathbf{S}$ *of at least two STL tasks as defined in Defs. 6.4.4 and 6.4.5. For any periodic task* $\Phi_i \in \mathbf{S}$*,* $\exists \Phi_j \in \mathbf{S}$ *such that* $\gamma_j \leq \gamma_{i^*}$ *where* γ_j *and* γ_{i^*} *are the laxity values of* Φ_j *and any successive insertion of* Φ_i *along* $\mathbf{S}$*, respectively, defined according to* (6.14)*.*

Proof. Without loss of generality, assume that the system will satisfy one task, Φ_{int}, between two consecutive satisfactions of Φ_i as shown in Fig. 6.4. Let the period of $\Phi_i = G_{[a,b]}F_{[c,d]}$ be denoted as $\tau = d - c$, then we have $\tau \geq \mathfrak{d}_{i,int} + \mathfrak{d}_{int,i}$ as $\mathbf{S}$ is feasible (Def. 6.4.5). Suppose another task Φ_{later} positioned in the sequence to be achieved later than the two different satisfactions of Φ_i. If we denote the laxity values of two

consecutive satisfactions of Φ_i by γ_{i_0} and γ_{i_1}, i.e., $\gamma_{i^*} = \gamma_{i_1}$, there are two possible cases: **Case 1:** $(\gamma_{i_0} \geq \gamma_{later})$ As $\gamma_{i_0} = \overline{r}_i - \mathfrak{d}_{\mathbf{S}(1)} - \sum_{j=1}^{s_{i_0}-1} \mathfrak{d}_{\mathbf{S}(j),\mathbf{S}(j+1)}$ and $\gamma_{later} = \overline{r}_{later} - \mathfrak{d}_{\mathbf{S}(1)} - \sum_{j=1}^{s_{later}-1} \mathfrak{d}_{\mathbf{S}(j),\mathbf{S}(j+1)}$, this case implies $\overline{r}_i \geq \overline{r}_{later} - \mathfrak{d}_{i,int} - \mathfrak{d}_{int,i} - \sum_{j=s_{i_1}}^{s_{later}-1} \mathfrak{d}_{\mathbf{S}(j),\mathbf{S}(j+1)}$, and it follows $\overline{r}_i \geq \overline{r}_{later} - \tau - \sum_{j=s_{i_1}}^{s_{later}-1} \mathfrak{d}_{\mathbf{S}(j),\mathbf{S}(j+1)}$. Now assume that $\gamma_{i_1} < \gamma_{later}$ holds. This assumption requires $\overline{r}_i + \tau < \overline{r}_{later} - \sum_{j=s_{i_1}}^{s_{later}-1} \mathfrak{d}_{\mathbf{S}(j),\mathbf{S}(j+1)}$ to be true which conflicts with what is shown in the previous step. Hence, any subsequent satisfaction cannot have the least laxity among $\mathbf{S}$ in this case.

Case 2: $(\gamma_{i_0} < \gamma_{later})$ In this case, proving $\gamma_{i_0} \leq \gamma_{i_1}$ is sufficient. Assume that the reverse, i.e., $\gamma_{i_0} > \gamma_{i_1}$, is true. By following a similar procedure in the first case, we obtain $\overline{r}_i > \overline{r}_i + \tau - \mathfrak{d}_{i,int} - \mathfrak{d}_{int,i}$ which is a conflict. □

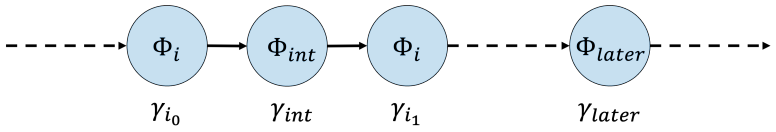


Figure 6.4: Tasks in a sequence with the associated laxity values, including a periodic task Φ_i. While solid arrows represent the immediate connection, dashed arrows denote the possibility of having other task(s) in between.

Proposition 6.4.2 will be useful in deciding which portion of the sequence is more time-critical without knowing the exact laxity values of the successive periodic task decompositions. While this time-critical portion governs the system, subsequent satisfaction requirements and deadlines for periodic tasks can be determined in real time, based on the previous satisfaction instants and knowing that they will not be time-critical.

6.4.3 Control Barrier Functions with Sequential Tasks

We construct sequential CBF candidates over a feasible sequence $\mathbf{S}$ of k tasks in (6.2) obtained via the Alg. 3 as:

$$\begin{aligned} \mathfrak{b}_i(\mathbf{x}, t) := \overline{r}_{\mathbf{S}(i)} - \frac{Dist_{wc}\big(\mathbf{x}, \mathcal{P}_{\mathbf{S}(1)}, \overline{r}_{\mathbf{S}(1)}\big)}{u_{\max}} \\ - \frac{\sum_{l=2}^{i} Dist_{wc}\big(\mathcal{P}_{\mathbf{S}(l-1)}, \mathcal{P}_{\mathbf{S}(l)}, \overline{r}_{\mathbf{S}(l-1)}, \overline{r}_{\mathbf{S}(l)}\big)}{u_{\max}}, \ \forall i \in \mathbf{S}, \end{aligned} \tag{6.17}$$

where $r_{\mathbf{S}(i)}$ is the remaining time to achieve i^{th} task in the sequence $\mathbf{S}$, and $\overline{r}_{\mathbf{S}(i)} := \min_{j\in[i,\ldots,|\mathbf{S}|]} r_{\mathbf{S}(j)}$ is the actual time bound to reach the i^{th} target according to the deadlines of the next tasks in the sequence as defined in Def. 6.4.6. Nonnegativeness of $\mathfrak{b}_i(\mathbf{x},t)$ for all i simply implies that the system should achieve the first task in the order, $\Phi_{\mathbf{S}(1)}$, such that the first i tasks in the sequence can still be achieved on time. Therefore, we want our system to always be in $\mathcal{B} = \bigcap_i \mathcal{B}_i$ for collective achievement, where $\mathcal{B}_i := \{\mathbf{x} \in \mathbb{R}^n \mid \mathfrak{b}_i(\mathbf{x},t) \geq 0\}$.

After determining the sequence of tasks $\mathbf{S}$ and the candidate CBFs, we apply the sequential CBF constraint based on the most time-critical candidate in (6.17) found via,

$$\mathfrak{b}(\mathbf{x},t) := \min_{i\in\mathbf{S}} \ \mathfrak{b}_i(\mathbf{x},t). \tag{6.18}$$

Remark 6.4.1. *Nonnegativeness of the most time-critical sequential CBF* $\mathfrak{b}(\mathbf{x},t) \geq 0$ *implies* $\mathfrak{b}_i(\mathbf{x},t) \geq 0,\ \forall i \in \mathbf{S}$. *Moreover, as discussed in Proposition 6.4.2, successive decompositions of a periodic task cannot have the most time-critical CBF via* (6.18).

Note that removing terms from a sequence of STL tasks yields feasible subsequences as shown in Lemma 6.4.1. We now present the following corollary to show that the sequential CBF stays nonnegative after the satisfaction of STL task(s) and the removal of the term(s) associated with them.

Corollary 6.4.1. *Consider an STL specification* $\Phi := \Phi_1 \wedge \ldots \wedge \Phi_k$ *and a sequence of its tasks* $\mathbf{S}$ *as in Def. 6.4.4 together with the duration definitions in* (6.7). *If a sequential control barrier function defined over* $\mathbf{S}$ *via* (6.17) *and* (6.18) *is nonnegative, then a new control barrier function similarly defined over any subsequence of* $\mathbf{S}$ *is also nonnegative.*

Proof. Note that the sequential control barrier function is constructed in (6.17) and (6.18) by the duration definitions in (6.7) and the sequential remaining time bounds in (6.13). Thus the corollary follows from Lemma 6.4.1, indicating the feasibility of the resulting sequence after a task removal, and Proposition 6.4.1 showing the viability of using sequential time bounds instead of the actual remaining times. □

Since we consider a quadratic running cost (as in Sec 6.3) and the CBF constraints in (2.6) are linear in control for a constant state, Problem 6.3.2 reduces to a quadratic

program (QP) with explicit input and STL-related CBF constraints to be solved sequentially over the discrete time:

$$\begin{aligned} &\min_{\mathbf{u}\in\mathcal{U}} \ \frac{1}{2}\mathbf{u}^T\mathbf{u} \\ &s.t. \ \frac{\partial \mathfrak{b}}{\partial \mathbf{x}}^T \mathbf{u} + \frac{\partial \mathfrak{b}}{\partial \mathbf{x}^{target}}^T \mathbf{u}^{target} + \frac{\partial \mathfrak{b}}{\partial t} + \alpha(\mathfrak{b}) \geq 0, \end{aligned} \tag{6.19}$$

where $\mathbf{x}^{target} \in \mathbb{R}^{nk}$ and $\mathbf{u}^{target} \in \mathbb{R}^{mk}$ are the stacked vector of current target positions and velocities, respectively. Let $\boldsymbol{r}(t) := [r_1, \ \ldots, \ r_k]^T$ be the stack vector of task remaining times and $\mathbf{1}_n \in \mathbb{R}^n$ be the vector of ones, then we have $\partial\mathfrak{b}/\partial t = -\partial\mathfrak{b}/\partial\boldsymbol{r}^T\mathbf{1}_k$ since $\partial r_i/\partial t = -1, \ \forall i$.

6.4.4 STL Control Synthesis under CBFs with Sequential Tasks

After a feasible sequence of STL tasks $\mathbf{S}$ is determined, a sequential CBF is constructed over it via (6.17) and (6.18). The QP problem equipped with CBF constraints (6.19) is solved at each time step over time in Alg. 5. When a task is achieved or at least a target area associated with it is reached, the task is removed from the sequence, and the sequential CBF constraints are updated accordingly. In this regard, when there are no tasks left in the sequence, we deduce that the STL specification Φ in (6.2) is satisfied. However, the removal of a task associated with the visited set does not necessarily mean the task is completely forgotten, as some STL tasks may require holding the target area for a while. Depending on the temporal operator of the newly achieved STL task, an additional CBF constraint that aims to hold the robot inside the target area may be initiated. Details of the sequential CBF implementation can be summarized based on the temporal operator(s) of each STL task Φ_i as follows:

- ***Finally*** **($\Phi_i = F_{[a,b]}\varphi_i$):** The STL task is inserted in $\mathbf{S}$ with a remaining time of $r_i = b$. We enforce its achievement with the sequential CBF until the inner task φ_i is satisfied within the given time window $[a, b]$. When it does, line 10 in Alg. 5 is not triggered and the task is just removed from the sequence in line 18.

- ***Globally*** **($\Phi_i = G_{[a,b]}\varphi_i$) or *Finally-Globally* ($\Phi_i = F_{[a,b]}G_{[c,d]}\varphi_i$):** Once the task satisfaction begins, the associated term inside the sequence is removed. Since these tasks require a hold for a predefined period of time (e.g., $b-a$ for $G_{[a,b]}\varphi_i$),

we enforce this hold with an individual CBF as in (6.5) until the last time step φ_i needs to hold by keeping the remaining time fixed to $r_i = 0$. For this aim, we

Algorithm 5: Main STL Sequential CBF Algorithm

input : STL specification $\Phi = \phi_1 \wedge \ldots \wedge \phi_k$ and feasible sequence $\mathbf{S}$; Vel. bounds $u_{\max}, u_{i,\max}^{target}$; Initial positions $\mathbf{x}(0), \mathbf{x}_i(0)^{target}$; Target radii rad_i^{target} Class $\mathcal{K}$ functions $\alpha_i(\cdot)$ for $i \in \{1, \ldots, k\}$

$t = 0$, $c_{hold} = 0$;
while $\mathbf{S} \neq \emptyset$ **do**
  Get target pos. $\mathbf{x}_i^{target}(t)$ and vel. $\mathbf{u}_i^{target}(t)$ $\forall i \in \mathbf{S}$;
  Calculate pairwise worst-case distances within $\mathbf{S}$;
  Formulate the CBF constraints via (6.17) and (6.18) by adding the hold-requirement if $c_{hold} > 0$ via (6.5);
  Solve (6.19) to find the input to the system;
  Apply the input $\mathbf{u}$ to the system; get new state $\mathbf{x}^+$;
  Increase t, decrease c_{hold} by one step;
  if *Any* $\mathbf{S}(s) \in \mathbf{S}$ *is hit* **then**
    if $\Phi_{\mathbf{S}(s)}$ *is not finally* **then**
      if $\Phi_{\mathbf{S}(s)}$ *is a periodic task* **then**
        Hold until forced to leave the current target;
      else if $\Phi_{\mathbf{S}(s)}$ *is globally or finally-globally* **then**
        Define $c_{hold} = I_{hold}$ via STL task $\Phi_{\mathbf{S}(s)}$;
        Enforce holding until $c_{hold} = 0$;
      end
    end
    $\mathbf{S} = \mathbf{S} \setminus \{\mathbf{S}(s)\}$;
    if $\mathbf{S}(1)$ *has alternative targets* **then**
      Get alternative target positions and velocities;
      Obtain potential worst-case distances;
      Compare alternative CBF values;
      Update $\mathbf{S}$ if any alternative yields higher CBF;
    end
  end
end

return : State trajectory $(\mathbf{x}, 0)$ and control policy $(\mathbf{u}, 0)$

introduce the c_{hold} parameter in Alg. 5 to count the required holding time within lines 14-15. Moreover, this additional holding requirement is already incorporated in the worst-case pairwise travel duration calculations from targets associated with such tasks in (6.9). In other words, we account for the time the robot must spend to hold while achieving such dynamic tasks in the sequence calculation.

- ***Globally-Finally*** **($\Phi_i = G_{[a,b]}F_{[c,d]}\varphi_i$):** We treat a periodic task like multiple decomposed *finally* tasks as explained in Sec. 6.4.1. When the task is satisfied within the given time interval, the robot is asked to stay inside the target as long as it can via an individual CBF as in (6.5) (line 12 in Alg. 5). When the remaining task deadlines in the sequence start to become urgent, the sequential CBF automatically makes the robot end the holding by forcing it out of the target area. As soon as this happens, the successive decomposition is assigned with the period of Φ_i as the remaining time.

- ***Until*** **($\Phi_i = \varphi_{i,1}U_{[a,b]}\varphi_{i,2}$):** We use equivalent tasks, $\Phi_i = G_{[0,t']}\varphi_{i,1} \wedge F_{[a,b]}\varphi_{i,2}$ where $\varphi_{i,2}$ is satisfied at $t' \in [a, b]$.

Details of the implementation can be found in Table 6.1. Moreover, when the sequence $\mathbf{S}$ is modified according to the given instructions, we check if the new $\mathbf{S}(1)$ has an alternative, i.e., involves disjunction. If it does, we calculate all possible sequential CBF values replacing the $\mathbf{S}(1)$ with its alternatives but keeping the rest of the sequence fixed. This also includes the update of the worst-case distances for each alternative scenario. In case an alternative target results in a higher CBF value over the sequence, we modify the sequence by replacing $\mathbf{S}(1)$ with a superior alternative (lines 19-24, Alg. 5).

Theorem 6.4.1 (Soundness)**.** *Given an STL specification $\Phi := \Phi_1 \wedge \Phi_2 \wedge \ldots \wedge \Phi_k$ with the syntax in* (6.1) *and a sequence $\mathbf{S}$ of k STL tasks with dynamic target areas as defined in Def. 6.4.4 such that*

$$r_{\mathbf{S}(j)} \geq \mathfrak{d}_{\mathbf{S}(1)} + \sum_{l=2}^{j} \mathfrak{d}_{\mathbf{S}(l-1),\mathbf{S}(l)}, \ \forall j \in \mathbf{S},$$

where r_i is the remaining time to reach the target associated with Φ_i; $\mathfrak{d}_i$ and $\mathfrak{d}_{i,j}$ are defined in (6.7)*, the control law obtained by sequentially solving* (6.19) *via Alg. 5 yields a trajectory for the system* (6.3) *that satisfies Φ.*

Table 6.1: How to implement the CBF constraints in accordance with the temporal operator of respective STL task and the associated time interval(s).

Temporal Operators of Φ_i	Initial Remaining Time, r_i	Handling of STL Tasks in the Sequence	Holding Mechanism
$F_{[a,b]}\varphi_i$	$r_i = b$	Once achieved within $[a, b]$, remove.	No additional holding is required.
$G_{[a,b]}\varphi_i$	$r_i = a$	Remove at $t = a$.	Once started, fix $r_i = 0$, hold until $t = b$.
$F_{[a,b]}G_{[c,d]}\varphi_i$	$r_i = b + c$	Remove if started within $t' \in [a + c, b + c]$.	Once started at t', fix $r_i = 0$, hold until $t = t' + d - c$.
$G_{[a,b]}F_{[c,d]}\varphi_i$	$r_i = a + d$	Whenever achieved within $[a + c, b + d]$, remove the current decomposition. Once left the target, reset $r_i = d - c$ for the next decomposition (if any).	Each time a periodic task is achieved, fix $r_i = 0$, hold until forced to leave by the sequential CBF.

Proof. Consider a time-varying CBF that is nonnegative at t', e.g., $\mathfrak{b}(\mathbf{x}(t'), t') \geq 0$ in (6.18). In light of Theorem 1 in [35], the control law $\mathbf{u}$ satisfying the constraints in (6.19) (i.e., $\mathbf{u}$ being inside the set $S_{\mathbf{u}}$ in [35]) implies $\mathfrak{b}(\mathbf{x}(t), t) \geq 0$, $\forall t \geq t'$. Let us now show that it is feasible to keep $\mathfrak{b}(\mathbf{x}(t), t)$ nonnegative. Let the target set associated with the task Φ_i be $\mathcal{P}$ and $\mathbf{x}^* \in \partial\mathcal{P}$, then the initially nonnegative sequential CBF implies $r(t') \cdot u_{\max} \geq \|\mathbf{x}^* - \mathbf{x}(t')\| + u_{\mathcal{P},\max} \cdot r(t')$ (it is $>$ if there are successive tasks) where t' is the first time the target set is switched to $\mathcal{P}$. Note that the system (6.3) is capable of moving according to $\|\mathbf{x}(t) - \mathbf{x}(t')\| = u_{\max}\big(r(t') - r(t)\big)$, $\forall t \geq t'$. Then we obtain $\|\mathbf{x}(t) - \mathbf{x}(t')\| + r(t) \cdot u_{\max} \geq \|\mathbf{x}^* - \mathbf{x}(t')\| + u_{\mathcal{P},\max} \cdot r(t')$, i.e., the system can reach $\mathbf{x}^*$ under $u_{\max}$ within $t \in [t', t' + r(t')]$ even if the target evades with the velocity of $u_{\mathcal{P},\max}$, and it is feasible to sustain the nonnegativeness of the sequential CBF. The sequential CBF $\mathfrak{b}(\mathbf{x}, t)$ is selected to be the most time-critical one among the candidates via (6.18). Therefore, if it is nonnegative, then all the candidates in (6.17) will be nonnegative as well, as discussed in the Remark 6.4.1, meaning each target in the sequence can be

reached before its deadline. Now let us consider the possible temporal operators:

Case 1: (Only $F_{[a,b]}\varphi$ among the tasks). By Lemma (6.4.1), the new sequence $\mathbf{S}' = \mathbf{S} \setminus \{\mathbf{S}(j)\}$ obtained after a task $\Phi_{\mathbf{S}(j)}$ is satisfied and removed is also feasible, and per Corollary 6.4.1 the sequential CBF remains nonnegative. Moreover, since r_i decreases for all i and $\mathfrak{b}(\mathbf{x}, t)$ stays nonnegative (as well as the other $\mathfrak{b}_i(\mathbf{x}, t)$'s as discussed in Remark 6.4.1), each task Φ_i is achieved and removed from $\mathbf{S}$ in at most r_i time units. This process continues until no task is left in the sequence $\mathbf{S}$.

Case 2: ($G_{[a,b]}\varphi$ or $F_{[a,b]}G_{[c,d]}\varphi$ among the tasks). The remaining time for the *globally* operator is defined to be the initial time of its interval, e.g., $r_G = a$ for $G_{[a,b]}\varphi$. Once the satisfaction starts, an individual CBF in (6.5) is initiated with the remaining time fixed as $r_G = 0$ until the end of *globally* time interval. This ensures the system stays inside the target set of the task. Moreover, the absence of *globally* task in the sequential CBF while it is still active does not undermine the satisfaction since the time required to hold φ is already considered in the sequence construction even when the target moves.

Case 3: ($G_{[a,b]}F_{[c,d]}\varphi$ among the tasks). The task is initially treated as $F_{[a+c,a+d]}\ \varphi$ and by resetting the remaining time by the period of $\tau = d - c$ for the subsequent satisfactions (i.e., $F_{[0,\tau]}\varphi$) once φ is satisfied and associated target area is left as explained in Table 6.1. The guarantees given in Case 1 continue until the satisfaction of φ that needs to be achieved repetitively. At each satisfaction, a holding mechanism is applied via the individual CBF in (6.5) as if no time left to get to the target. This delays the leave from the target assigned to the periodic task, and hence the deadline for the subsequent satisfaction of it. As the latest time the system can leave the target after a periodic satisfaction is accounted for during the sequence generation and the sequential CBF remains active which makes the system leave the periodic task satisfaction, the completion of the sequence is not undermined. □

6.4.5 Complexity Analysis

We analyze the computational complexity of the proposed framework in Fig 6.2. The bulk of the computation lies in generating candidate sequences and decomposition of periodic tasks which are both performed offline before the execution.

For the online part (Alg. 5), the computational complexity is influenced by the horizon of the STL specification, and major factors are generating the CBF constraints relying on the total number of tasks in conjunction $\big($as in (6.2)$\big)$, solution of the QP problem in (6.19), and the selection procedure which is triggered when a spatial requirement of the next task is redundant due to disjunction. In this regard, the worst-case complexity for the online iterative algorithm can be obtained as $O\big(hrz(\Phi)\times(k+n)+\sum_{i\in\{1,\dots,k\}} q_i\big)$ where $hrz(\Phi)$ is the STL horizon, k is the number of STL tasks in conjunction, n is the number of states of the system, and q_i is the number of alternative target areas associated with the task Φ_i (for details on the handling of disjunction, see Sec. 6.4.1). The right-hand side of the summation is dropped when there is no disjunction.

The offline part includes the sequence generation with periodic task decomposition and the determination of the best sequence with the greatest least-laxity. If there is no disjunction or periodic task, the number of operations in sequence construction (Alg. 3) scales as $O\big(k^2k!\big)$. When we have disjunction(s), it becomes $O\big(k^2k!\times\prod_{i\in\{1,\dots,k\}} q_i\big)$. In the presence of periodic tasks, the candidate sequences in Alg. 3 have to branch out by calling Alg. 4 and increasing the complexity to $O\big(k^2k!\times\prod_{i\in\{1,\dots,k\}} q_i\big)+O(BranchOut)$. Note that the complexity of Alg. 4 $\big($i.e., $O(BranchOut)\big)$ depends heavily on the number of periodic tasks and how many decompositions of periodic tasks can be inserted throughout the mission horizon. In particular, the depth-first search is implemented via feasibility checks yielding an arbitrary depth depending on how the targets are initially placed and the feasibility of candidate sequences. The worst-case depth length would be the total number of time instances a periodic task can be decomposed into, i.e., $hrz(\Phi)/\Delta t$. At each such instant, a periodic task triggers the Alg. 4 recursively, and in the worst case when all tasks are periodic, $k-1$ periodic tasks remain besides the task that triggers Alg. 4 in the first place. Hence, the worst-case breadth size could be $k-1$. As a result, Alg. 4 grows at worst as $O\big((k-1)^{hrz(\Phi)/\Delta t}\big)$.

6.4.6 Extension of the Approach

The system (6.3) with a scalar input bound $u_{\max}$ guarantees that if the proposed CBF (6.17),(6.18) is initially nonnegative, then the system can reach the target set on time. Note that the bound $u_{\max}$ does not change with time. That is, ensuring $\mathfrak{h}(\mathbf{x},t)\geq 0$ within $t\in[0,r]$, forces the system to stay in the proximity of the target set such that in

the worst case the set can be reached in at most $r - t$ time units (and is reached when $t = r$) with the maximal input policies. On the other hand, for more complex systems than (6.3), e.g., (2.5), we could have nonnegative CBFs over time, if we had an overall bound on the derivative of each state, e.g., velocity, and a system property making this bound nondecreasing with time (at least under maximal inputs). Then the proposed CBF could preserve its nonnegativeness. For example, monotone and/or cooperative control systems [105, 106], can satisfy this under mild assumptions.

6.4.7 Conservativeness of the Approach

The sequence of subtasks (Def. 6.4.4) is constructed via the ordered set distances (Def. 6.4.3) which assume that a target set of states traveled with a maximum speed and is reached through its farthest point with respect to the next target set. This may result in not having a feasible sequence $\mathbf{S}$ whereas the STL specification is actually attainable. The user may circumvent this by relaxing the proposed CBF constraint in (6.19), and penalizing the relaxation variable in the objective function, similar to [87,89,107]. After noting these, it is worth reiterating that here we assume STL specifications that have a feasible sequence $\mathbf{S}$, and CBF functions that are initially nonnegative (under $u_{\max}$) so that we could pursue forward invariance over their zero superlevel sets.

6.5 Case Studies: Dynamic Targets

We develop a control synthesis tool that constructs a feasible sequence of STL tasks (Algs. 3 and 4) prior to deployment and generates online trajectories (Alg. 5) satisfying the given STL specification (6.2). We validate the efficacy of the proposed approach via simulations and experiments. The code is created in Python by utilizing CasADi [108] for symbolic encoding of the problem (6.19) and we obtain the sequential solutions using `qpoases` solver. While the proposed method is agnostic to the actual robotic system to be deployed, we tested our algorithms for the real-time planning of Crazyflies 2.0 drones. The proposed method is verified experimentally in real time for various scenarios. The targets are considered as circles assigned to individual planar drones with given radii. The experiments are conducted in a $4.6m \times 3.5m \times 3m$ indoor environment as shown in Fig. 6.5, using a `VICON` motion capture system with 8 cameras. We fly the ego drone

(in red) at a higher altitude than the targets to avoid collisions. In the experiments, we use the Crazyswarm [109] package to perform the low-level controls, communication, and interface of the `VICON` system with the Crazyflies.

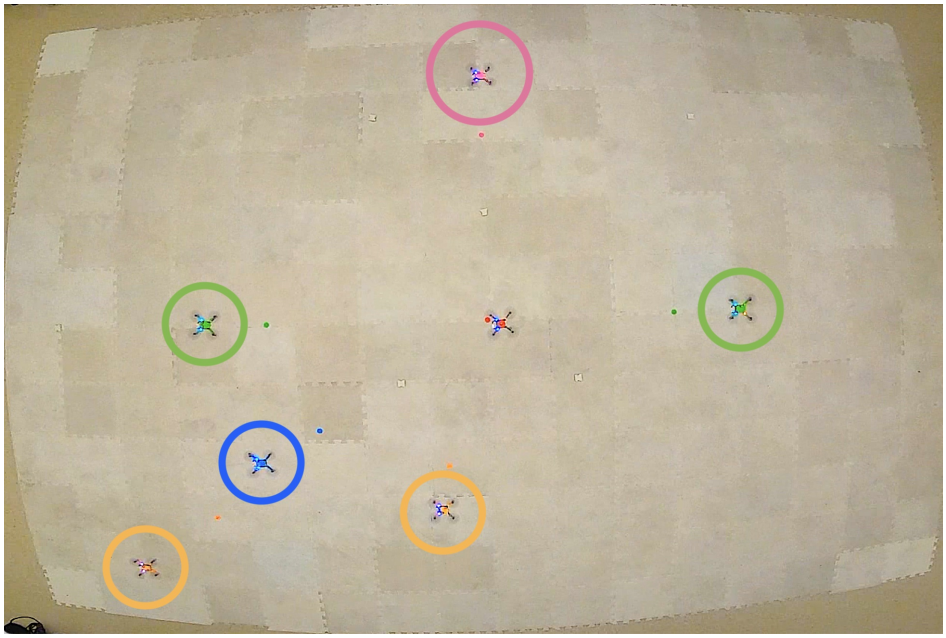


Figure 6.5: Physical environment for case 1 (scenarios 1 and 2). The Crazyflies are at their initial positions. While the red drone in the middle is controlled (ego), the remaining drones represent the targets with different colors according to the temporal operators of the STL tasks assigned to them.

We consider five scenarios under different STL specifications: i) the main scenario with a complex STL specification $\Phi_{case\,1}$ defined over dynamic tasks moving on predefined trajectories with maximum velocity, ii) the same scenario with randomly moving targets, iii) another specification $\Phi_{case\,2}$ comprised of only periodic tasks, iv) a specification $\Phi_{case\,3}$ defined over adversarial targets which are repelled by the controlled robot, and v) the same adversarial scenario with much faster targets. In addition to these scenarios, in another case, we considered the increasing number of targets, each assigned with the same reach and hold type of STL specification, to demonstrate the scalability of our approach. Details of the scenario settings are shown in Table 6.2.

Note that in none of the scenarios shown here, the robot is aware of the future

Table 6.2: Mission parameters for different scenarios with the STL specifications in (6.20), (6.21), (6.22), and (6.24).

Case	STL Spec.	Vel. bounds $[m/s]$, $\left(u_{\max}, u_{\max}^{target}\right)$	Target Trajectory Profiles
Scenario 1	$\Phi_{case\,1}$	$\left(0.7, 0.06\right)$	Predefined w/maximal velocities
Scenario 2	$\Phi_{case\,1}$	$\left(0.7, 0.06\right)$	Random velocities
Scenario 3	$\Phi_{case\,2}$	$\left(0.7, 0.05\right)$	Predefined w/maximal velocities
Scenario 4	$\Phi_{case\,3}$	$\left(0.6, 0.08\right)$	Adversarial obeying to (6.23)
Scenario 5	$\Phi_{case\,3}$	$\left(0.6, 0.2\right)$	Adversarial obeying to (6.23)
Scalability Analysis	$\Phi_{case\,4}$	$\left(0.7, 0.1\right)$	Predefined w/maximal velocities (spiral)

trajectories of the targets. In other words, the targets either follow predefined trajectories (scenarios 1 and 3) or calculate their own (scenarios 2, 4, and 5), and all these cases are unknown to the robot. The only available information regarding the targets is their instantaneous position and velocity, of which real-time estimations are possible via methods such as LiDAR-based sensing [103].

6.5.1 Scenarios 1 and 2: Main Scenario with Different Target Trajectories

In these scenarios, we consider six different targets assigned with various temporal operators. Moreover, two of these targets (4 and 6) are interchangeable via a disjunction operator. The STL specification that expresses spatial and temporal requirements over the dynamic targets is given as follows:

$$\begin{aligned}\Phi_{case\,1} := & F_{[0,15]} Target_{1,1} \wedge F_{[0,15]} Target_{1,2} \\ & \wedge G_{[33,35]} Target_{1,3} \\ & \wedge F_{[0,27]} G_{[0,3]} \left(Target_{1,4} \vee Target_{1,6}\right) \\ & \wedge G_{[0,20]} F_{[0,10]} Target_{1,5},\end{aligned} \tag{6.20}$$

where $Target_{i,j}$ is the expression implying the visiting of j^{th} circular target in the specification $\Phi_{case\,i}$ at t as an STL predicate in the form of $\left(x(t) - x_{i,j}(t)\right)^2 + \left(y(t) - y_{i,j}(t)\right)^2 \leq rad_{i,j}^2$ with the robot position vector $\mathbf{x} = [x,\ y]^T$. The specification in (6.20)

requires the robot to visit the first two targets within the first 15 seconds. Moreover, the third target has to be continuously visited within the interval of $[33, 35]$. The controlled robot also needs to reach either the fourth or sixth target within the first 30 seconds and stay inside for at least three seconds. While conducting all these tasks, it has to keep visiting the fifth target with a period of 10 seconds.

Note that all the tasks except the one defined on the Target 3 have conflicting requirements with overlapping time windows $\big($e.g., eventually visit Targets 1 and 2 within the same interval $[0, 15]\big)$. We show some scenarios in Sec. 6.6, which showcase the potential failure of the existing STL control synthesis approaches using CBFs for such tasks.

In the first scenario, the targets are assigned with some trajectories in the form of circular arcs requiring them to move outwards from the environment with maximum velocities. In the second scenario, the targets are assigned random velocities picked uniformly, obeying the velocity bounds. The trajectories of the ego drone and targets can be seen in Fig. 6.6 as snapshots from the experiment accompanied by simulation visuals.

Table 6.3: Scheduled sequence of STL tasks for each respective specification (output of the Alg. 3) and the actual sequence realized by the controlled robot (output of the Alg. 5).

	$\Phi_{case\,1}$	$\Phi_{case\,2}$	$\Phi_{case\,3}$
Initial task sequence	$\{1, 5, 2, 5, 4, 3\}$	$\{1, 2, 3, 1, 2\}$	$\{1, 2\}$
Realized task sequence	$\{1, 5, 2, 5, 6, 3\}$	$\{1, 2, 3, 1, 2\}$	$\{1, 2\}$

The Targets 4 and 6 are interchangeable as expressed in (6.20). Although the initial task sequence generated by Alg. 5 includes Target 4 (see Table 6.3), when the target previous to 4 is reached, the robot decides to go for Target 6 next, as it is closer at the moment. This is because the sequence comprising the Target 6 constitutes a higher CBF value.

The second scenario also conducts the same experiment with random target velocity vectors. Out of fifty randomized trials, the execution times of the simulations are presented in Table 6.4. Only one of these random cases is conducted as an experiment with drones for verification. The sequence generation process is independent of target

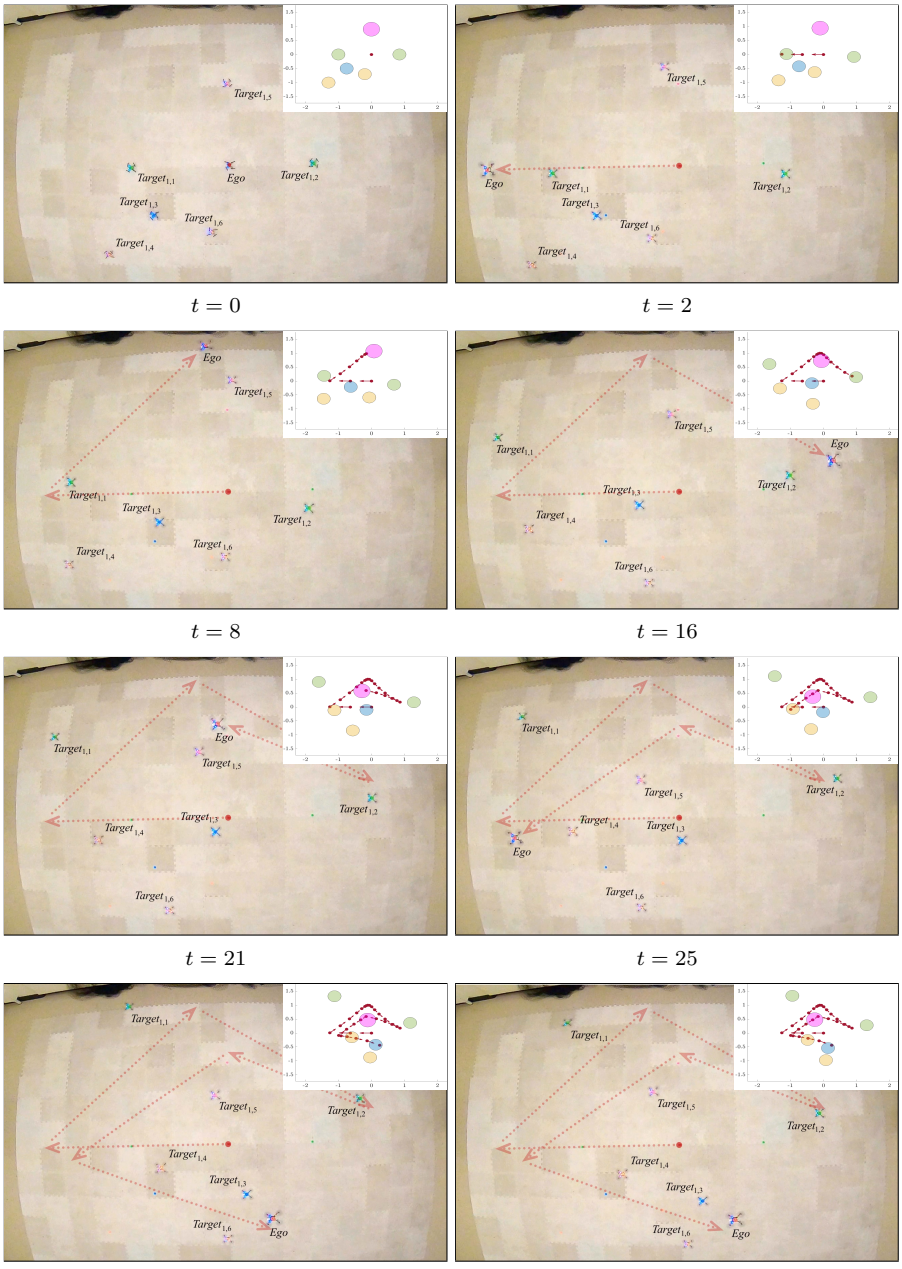


Figure 6.6: Time history of the ego and target drones yielding the satisfaction of the specification $\Phi_{case\,1}$ in (6.20). While the green drones belong to the *finally* tasks, the blue and magenta targets correspond to *globally* and *globally-finally* (periodic) tasks, respectively. Two target areas in disjunction inside the *globally-finally* task are associated with the yellow drones. The abstract motion of the ego drone between the task areas is represented by dashed arrows.

trajectories which are not available as a priori. Moreover, the execution times mostly depend on the number of targets and the mission horizon due to the online nature of the approach. In this regard, the simulation results for scenarios 1 and 2 are presented jointly under $\Phi_{case\,1}$ in Table 6.4, for the same specification.

Table 6.4: Simulation execution times required for generating STL-satisfying trajectories subject to the specifications in (6.20), (6.21), and (6.22).

STL Specification	Algs. 3 & 4 — Offline	Algorithm 5 — Online		
	Sequence Generation Time $[ms]$	Average QP Solution Time $[ms]$	Average Online Runtime $[ms]$	Total Online Runtime $[ms]$
$\Phi_{case\,1}$ $(n=50)$	40.6	0.3 ± 0.006	1.0 ± 0.016	36.2 ± 0.501
$\Phi_{case\,2}$	11.6	0.3	1.0	30.8
$\Phi_{case\,3}$	1.9	0.3	1.3	19.7

6.5.2 Scenario 3: All Periodic Tasks

In this scenario, three targets are placed in the environment, each of which has to be repetitively reached with a period of 15 seconds. The STL specification expressing this requirement can be formulated as follows:

$$\begin{aligned}\Phi_{case\,2} := & G_{[0,15]}F_{[0,15]}Target_{2,1} \wedge G_{[0,15]}F_{[0,15]}Target_{2,2} \\ & \wedge G_{[0,15]}F_{[0,15]}Target_{2,3}.\end{aligned} \tag{6.21}$$

The targets move according to some predefined circular arc trajectories. The ego drone keeps visiting each region such that the gap between two consecutive visits is not greater than 15 seconds. The trajectories of the ego drone and the targets are depicted in Fig. 6.7 with simulation results in Table 6.4. As shown in Table 6.3, the robot realizes the initially proposed sequence with the decomposed periodic tasks.

6.5.3 Scenarios 4 and 5: Pursuit-Evasion under Different Target Velocities

In this scenario, we consider two different targets repelled by the ego drone. The robot has to reach and stay with the targets for 2 seconds within the first 15 seconds. The

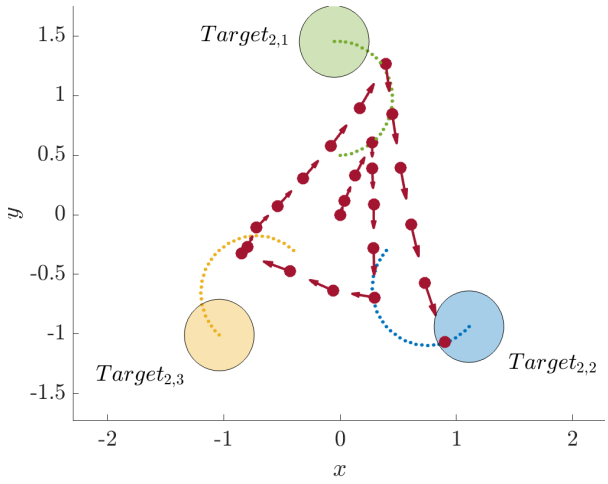


Figure 6.7: Satisfactory robot trajectory for the specification $\Phi_{case\,2}$ in (6.21) with three periodic tasks defined over the targets moving on circular arcs with their maximum velocities. The target trajectories are illustrated with dashed lines. Duration of the mission is $hrz(\Phi_{case\,2}) = 30$ seconds.

specification defined over these dynamic targets is as follows:

$$\Phi_{case\,3} := F_{[0,13]}G_{[0,2]}Target_{3,1} \wedge F_{[0,13]}G_{[0,2]}Target_{3,2}. \tag{6.22}$$

For the adversarial targets, we consider "indirect herding under STL specifications" setting and utilized a target motion in the form of [96]:

$$\mathbf{u}_i^{target} := \begin{cases} [0 \;\; -u_{i,\max}^{target}]^T, & \|\mathbf{x}_i^{target} - \mathbf{x}\| \geq 2, \\ \dfrac{(\mathbf{x}_i^{target} - \mathbf{x}) \cdot u_{i,\max}^{target}}{\|\mathbf{x}_i^{target} - \mathbf{x}\|} \cdot \left(1 - e^{-\|\mathbf{x}_i^{target} - \mathbf{x}\|}\right), & \|\mathbf{x}_i^{target} - \mathbf{x}\| < 2, \end{cases} \tag{6.23}$$

which requires the targets to follow a predefined trajectory in *unchased* mode first, and when the controlled robot is close enough, it starts to repel the targets.

While we need relatively slow targets to start with a nonnegative CBF (Fig. 6.8a), we also tried faster targets that yield negative CBFs. In practice, the controlled robot satisfied the STL specification in (6.22) even for faster targets as depicted in Fig. 6.8b.

As the execution times depend heavily on the number of targets and the mission horizon due to the online nature of our approach, execution times for the simulations of scenarios 4 and 5 are presented jointly under $\Phi_{case\,3}$ in Table 6.4.

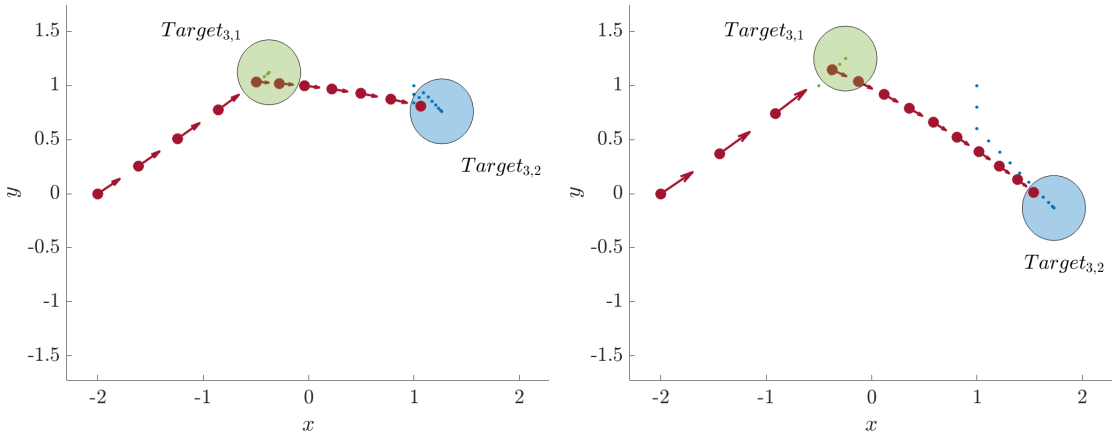


Figure 6.8: Satisfactory robot trajectories for specification $\Phi_{case\,3}$ in (6.22) with adversarial targets in (a) scenario 4 with the targets slow enough to start with a nonnegative CBF function and (b) scenario 5 with the targets having higher velocity bounds. Mission horizon is $hrz(\Phi_{case\,3}) = 15$ seconds.

6.5.4 Scalability Analysis

As shown in Table 6.4, the number of STL tasks placed in the sequence, including all repetitions, is the dominating factor in the sequence generation process. On the other hand, due to the nature of our approach, which enforces multiple tasks within a single CBF, the online part of the algorithm is not drastically affected by the increasing number of tasks.

To confirm these findings systematically, we perform a scalability analysis in which we examine multiple scenarios with varying numbers of targets, each assigned with a similar STL specification. We consider $k \in [1, 20]$ targets that need to be reached and held onto for 2 seconds within the first $7k$ seconds. The mission horizon is kept variable depending on the number of targets since a mission with more targets takes longer to complete. The overall specification can be constructed as:

$$\Phi_{case\,4} := \bigwedge_{i=1}^{k} F_{[0,7k-2]} G_{[0,2]} Target_{4,i}. \tag{6.24}$$

Depending on the total number k, each target is placed at equal distances on a circle centered at the origin with the radius of 1 m. Each target is assigned spiral trajectories on which they move away with their maximum velocities. Some sample trajectories

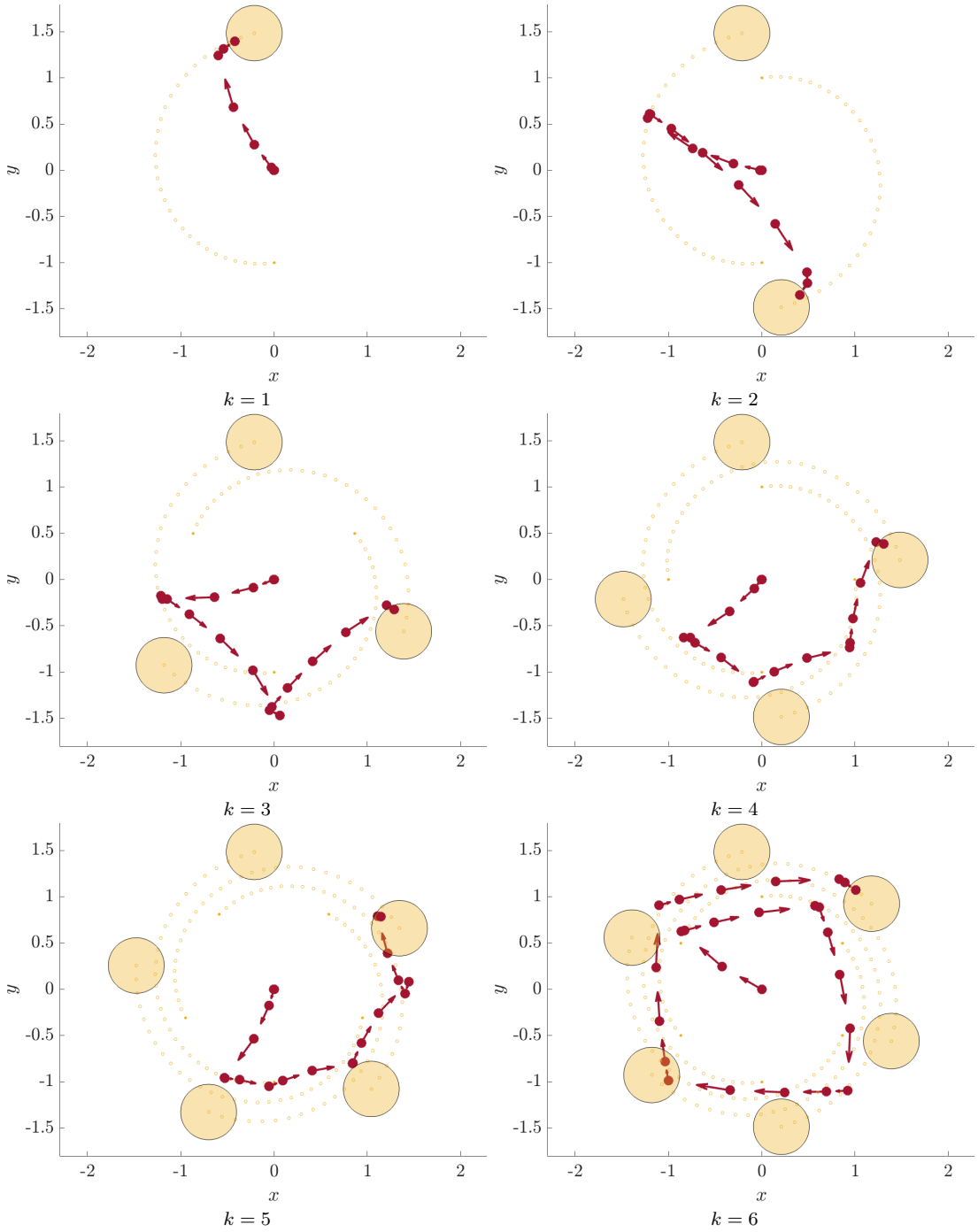
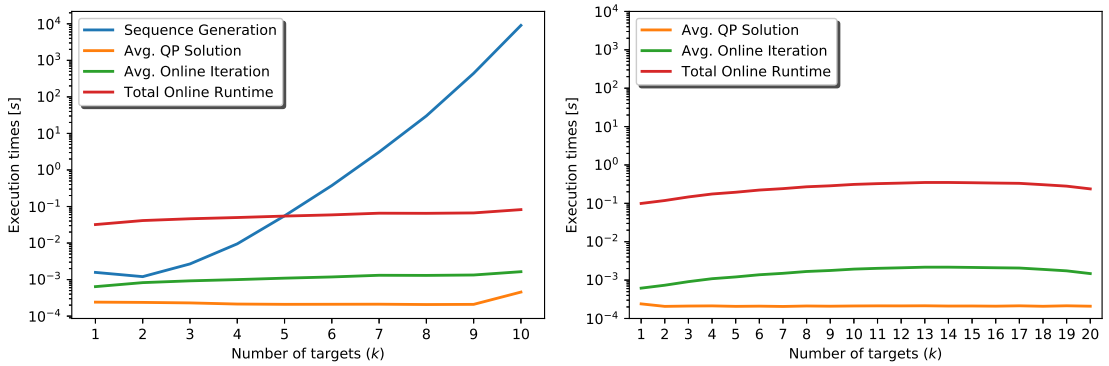


Figure 6.9: Robot trajectories satisfying the specification of $\Phi_{case\,4'} := \bigwedge_{i=1}^{k} F_{[0,28]} G_{[0,2]} Target_{4,i}$ for the varying number of dynamic targets. Targets follow circular trajectories that expand outwards for the duration of $hrz(\Phi_{case\,4'}) = 30$ seconds.



(a) (b)

Figure 6.10: How execution time scales with the length of the STL specification depending on the total number of dynamic targets (a) with the proposed sequence generation framework and (b) with a feasible sequence of STL tasks known as a priori. Beware of the logarithmic scale in the vertical axis.

with up to six targets with a fixed mission horizon are illustrated in Fig. 6.9.

As expected, the only major impact of the increasing number of targets was in the sequence generation time. Due to the proposed offline sequence generation framework that exhaustively evaluates all potential sequence candidates, specifications with 10 or fewer tasks in conjunction appear to be practical. The results for up to 10 targets are presented in Fig. 6.10a. Since this sequence generation process is handled offline prior to the deployment, the only restricting factor for the real-time implementation would be the execution times of each iteration. With the online nature of our approach involving solutions to simple optimization problems (e.g. quadratic programs), iteration times seem favorable and not affected much by the increasing number of targets. In this regard, if a feasible sequence is known a priori, the number of targets has only a minor impact on the online execution times and the online part of the method scales very well with the increasing number of targets as depicted in Fig. 6.10b.

6.6 Application to Stationary Targets

We now investigate the application of our approach to dynamical systems under STL specifications with fixed target regions. This enables us to compare our results with the existing work on STL control synthesis which consider stationary targets.

The theoretical results presented in this chapter carry forward to this setting as well. In particular, we build barrier functions using $Dist(\mathbf{x}, \mathcal{P})$ from Def. 6.4.1 instead of $Dist_{wc}(\mathbf{x}, \mathcal{P})$ defined in (6.4). This is a trivial transformation of the existing worst-case distance definitions by setting $u_{i,\max}^{\mathcal{P}_i} = 0,\ \forall i$.

6.6.1 Benchmark Analysis

The inclusion of actuation limits in the formulation of CBFs and considering the sequential satisfaction of STL subtasks have multiple advantages. We will show these benefits compared to the existing methods. Note that for each specification Φ in this subsection, we use Φ' in our method, which is obtained by bounding the predicates in Φ for the sake of compact target sets as our method requires. For example, in (6.25), we apply $x \geq 10 \wedge x \leq 11$ instead of $x \geq 10$, and $x \leq 5 \wedge x \geq 4$ instead of $x \leq 5$. This neither undermines nor facilitates the accomplishment of the subtask (since $5 \leq x_0 \leq 10$). The original specifications are used for the other methods in the benchmark analysis as required.

First of all, [35] mandates the achievement of fixed waypoints between the initial state and the target by using time-varying CBFs. Such an approach demands continuous progress toward the target and results in infeasibility when multiple targets are active in different directions. In [89], such conflicting subtasks are tried to be handled via relaxations with finite-time convergent CBFs (causing delays in the satisfaction and potentially violation of the original specification). For example, consider an STL specification and its bounded equivalent under the initial condition $x_0 = 8$,

$$\begin{aligned} \Phi &= \Phi_1 \wedge \Phi_2 = F_{[0,5]} x \geq 10 \wedge F_{[1,6]} x \leq 5, \\ \Phi' &= F_{[0,5]}(x \geq 10 \wedge x \leq 11) \wedge F_{[1,6]}(x \leq 5 \wedge x \geq 4). \end{aligned} \tag{6.25}$$

In [89], the CBF associated with $\Phi_2 = F_{[1,6]} x \geq 5$ is relaxed since its time window is later than $\Phi_1 = F_{[0,5]} x \geq 10$. However, this relaxation yields the violation of Φ_2 as Fig. 6.11 depicts. The proposed CBF, on the other hand, achieves both specifications on time by checking if it is feasible to go to first $x \leq 5$, then $x \geq 10$, and vice versa within the allowed time.

The incapability to define and achieve conflicting specifications as in the above case results in failure for the periodic subtasks as well. Consider the STL specification and

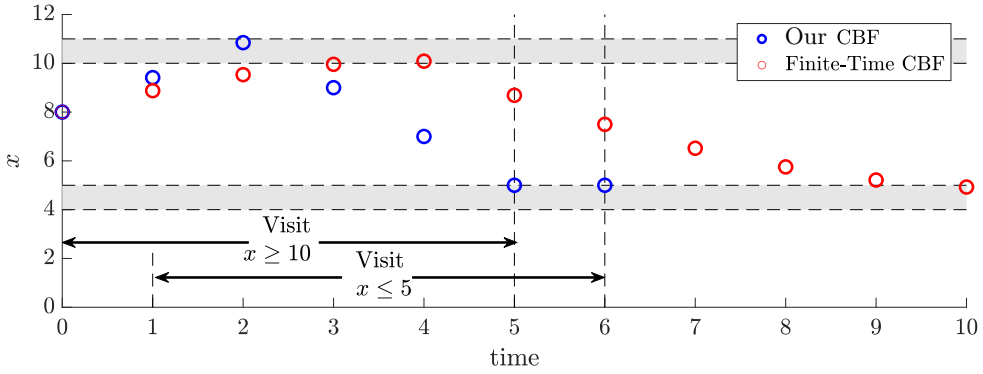


Figure 6.11: Comparison of the two CBF approaches for the STL specifications Φ and Φ' in (6.25).

its bounded equivalent with $x_0 = 7$ given below,

$$\begin{aligned}\Phi &= G_{[0,20]}F_{[0,10]}x \geq 10 \wedge F_{[0,15]}x \leq 5 \wedge F_{[20,30]}x \leq 3,\\ \Phi' &= G_{[0,20]}F_{[0,10]}(x \geq 10 \wedge x \leq 11) \wedge F_{[0,15]}(x \leq 5 \wedge x \geq 4)\\ &\qquad \wedge F_{[20,30]}(x \leq 3 \wedge x \geq 2).\end{aligned} \tag{6.26}$$

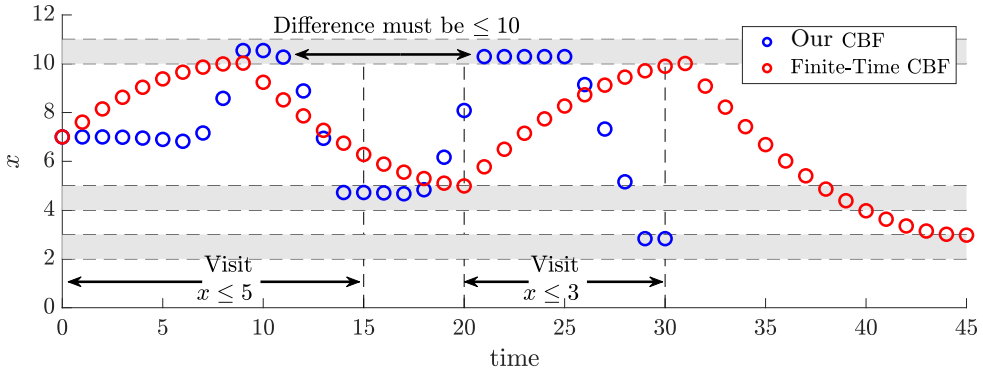


Figure 6.12: Comparison of the two CBF approaches for the STL specifications Φ and Φ' in (6.26).

In general, recursive task definitions are missing in the STL CBF literature. Still, by giving the order of satisfaction for the predicates as a priori in an ad-hoc way, the approach in [89] may generate a trajectory that visits desired regions repetitively. But it fails to do so on time again even when the relaxation is applied (and the order of relaxations is predetermined). Figure 6.12 depicts this case while the proposed CBF approach generates a trajectory that repetitively visits $x = 10$ while achieving other

subtasks on time. Another advantage of the proposed CBF approach is better handling of the disjunction operator. To illustrate, consider the STL specification below with its equivalent under $x_0 = 5$,

$$\begin{aligned}\Phi &= \Phi_1 \wedge \Phi_2 = F_{[10,15]} x \geq 9 \wedge \big(F_{[0,5]} x \leq 3 \vee F_{[0,5]} x \geq 7.5\big),\\ \Phi' &= F_{[10,15]}(x \geq 9 \wedge x \leq 10) \wedge \big(F_{[0,5]}(x \leq 3 \wedge x \geq 2)\\ &\qquad \vee F_{[0,5]}(x \geq 7.5 \wedge x \leq 8.5)\big).\end{aligned} \tag{6.27}$$

As Fig. 6.13 represents, for [35][3] and [89], applying the disjunction yields to achieve the closer alternative. However, this may cause an unnecessary delay in accomplishing other subtasks (e.g., Φ_1 in (6.27)) and an additional input cost.

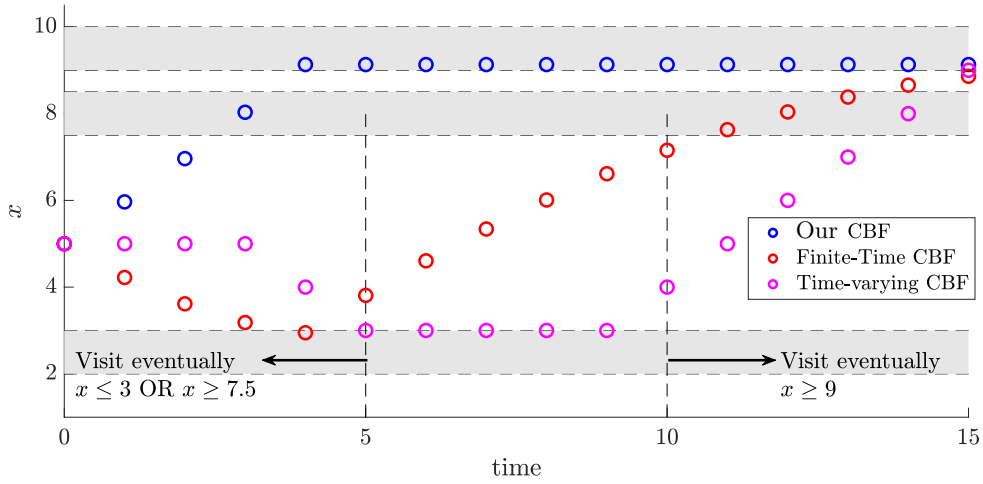


Figure 6.13: Comparison of the three CBF approaches for the STL specifications Φ and Φ' in (6.27).

6.6.2 Case Study

We also consider a more complex scenario to illustrate the capabilities of the proposed approach, which cannot be defined using other STL CBF approaches (e.g. [35,89]), with $\mathbf{x} = [x\ y]^T$, $\mathbf{u} = [u_x\ u_y]^T$, and $\|\mathbf{u}\| \leq 1$. The STL specification the system is required

[3]The work in [35] normally does not consider disjunction operator and preserves soundness. Hence, we added a sound disjunction operator (by using a smooth approximation of max to enrich the comparison.

to achieve is

$$\begin{aligned}\Phi &= \Phi_1 \wedge \Phi_2 \wedge \Phi_3 \wedge \Phi_4 \wedge \Phi_5 \\ &= G_{[0,15]}F_{[0,10]}\ Region_1 \wedge F_{[0,15]}\ Region_2 \wedge F_{[0,15]}\ Region_3 \\ &\wedge F_{[0,40]}G_{[0,10]}\ Region_4 \wedge G_{[38,45]}\ \big(Region_5 \vee Region_6\big),\end{aligned} \tag{6.28}$$

where $Region_i = \{x, y \in \mathbb{R} \mid (x - \boldsymbol{x'}(i))^2 + (y - \boldsymbol{y'}(i))^2 \leq \boldsymbol{rad}(i)^2\}$, and we use $\boldsymbol{rad} = [1.5, 0.5, 1, 1, 0.75, 0.75]$, $\boldsymbol{x'} = [7, 2, 12, 7, 11, 3]$, and $\boldsymbol{y'} = [6, 5, 5, 2, 2, 2]$ for $i = 1, \ldots, 6$. Note that the tasks defined on the regions $1 - 4$ have conflicts, and the last two tasks may have overlapping requirements as well (depending on when region 4 is visited).

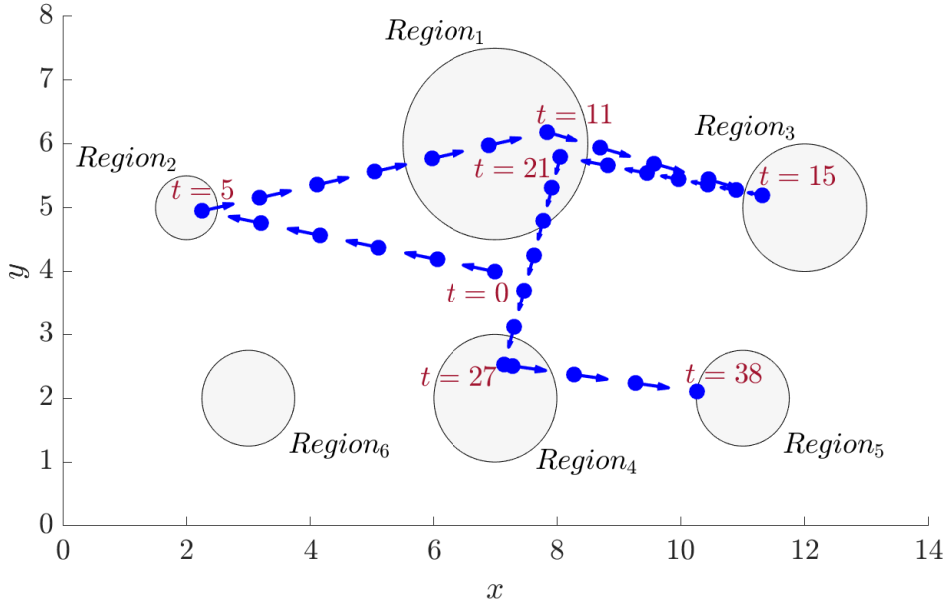


Figure 6.14: Trajectory of the system satisfying the STL specification in (6.28).

A new sequence is calculated only once at $t = 11$. Moreover, while region 6 is listed in the new sequence as the last one to be visited, the system visits region 5 instead as Fig. 6.14 depicts. This is because of the redundancy provided by the disjunction operator. The proposed (secondary) and task-specific (primary) CBF values that are required to be nonnegative throughout the mission are shown in Fig. 6.15. Note that while the task-specific CBFs are switched as the tasks are accomplished, the proposed CBF only loses term as the regions are visited.

Finally, we compare our results for the STL specification in (6.28) with the common

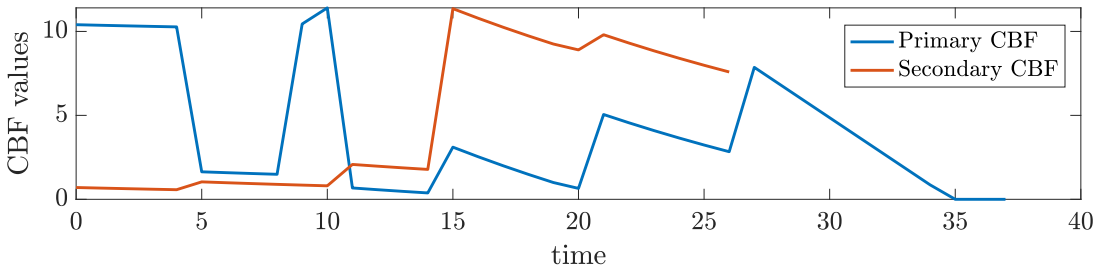


Figure 6.15: The change in the values of individual (primary) (6.5) and sequential (proposed) (6.18) control barrier functions which are required to be always nonnegative.

STL control synthesis methods that include the solution of a mixed-integer quadratic program[4] (MIQP) as in [14] and a nonlinear program (NLP) formulated with the smooth STL robustness metrics [27]. Table 6.5 shows that our approach provides a time-efficient solution for a reasonable cost of additional input. Furthermore, compared to the MIQP and NLP approaches which are defined through a horizon, the sequential implementation of the QP in (6.19) facilitates the real-time applicability with the stepwise solutions in order of milliseconds.

Table 6.5: Comparison of the results with popular control synthesis approaches for the STL specification in (6.28).

Solution Method	Proposed CBFs	Smooth Rob. Metrics (NLP)	MIQP Encoding
Total Cost $\sum_t \mathcal{J}^*$ [–]	20.29	14.65	17.03
Solution Time [s]	0.48 (Avg. 0.013 ms)	2.29	9.24

6.7 Discussion

This chapter addresses the problem of planning trajectories for robotic systems that aim to satisfy Signal Temporal Logic specifications that contain dynamic targets (i.e., time-varying predicates). The main challenge of the problem is making decisions in a way

[4]Linearity requirement in the constraints of the MIQP problem is met by specifying the subtask regions as inner-fitted squares to the circular subtask areas in the original specification (6.28).

that the STL specification over the dynamic noncooperative targets is satisfied while the robot does not have knowledge of how targets are moving. We tackle this challenge by proposing a methodology that creates a feasible sequence of STL tasks and time-varying control barrier functions associated with this sequence. The STL tasks we deal with can include temporally overlapping, redundant, and periodic requirements. We construct Sequential CBFs to account for the worst-case bounds for target motions and achieve logical, temporal, and spatial requirements associated with these targets. The proposed CBF formulations include the actuation limits and the satisfiability of multiple tasks in the mission. We guarantee the satisfaction of STL specifications defined over the compact target sets if there is a feasible sequence over the convex under-approximations of these sets. The sequence feasibility implies nonnegative barrier functions requiring targets with bounded velocities. Under these assumptions, the target motion, e.g., random, predefined, or adversarial, has no major effect on the success. Moreover, such changes in the mission setting, along with the high number of targets, do not undermine the real-time applicability as all of the tasks are incorporated into a single CBF, and the bulk of the computation lies in the generating sequence, which is performed offline before the execution. That said, even if it is executed offline, the proposed task scheduling framework currently performs efficiently for less than 10 tasks in conjunction. The future direction of this work can include extending the proposed framework with more efficient scheduling, possibly by using an STL tree (e.g., [110]), to richer specifications and a broader family of dynamical systems.

Chapter 7

Conclusion and Future Directions

7.1 Conclusion

To conclude the dissertation, we refer to the research questions presented in Sec. 1.2 once again. Regarding Research Question 1, the proposed integral and derivative predicates for Signal Temporal Logic in Chapter 3 prove useful in defining specifications regarding the cumulative effects and the rate of change in a signal, thereby preemptable tasks. Multi-agent settings with heterogeneous dynamics and abilities can significantly benefit from these new capabilities as investigated in Chapter 4. In particular, we can now define flexible and redundant mission tasks with temporal logic specifications. This is done optimally by formulating mixed-integer programs that are almost agnostic to the increasing number of agents, proving scalable. However, one needs to remember that such programs are shown to be NP-hard, and various parameters such as mission length can significantly increase the computation time.

Although we can compute the delays in the mission via post-processing as shown in Chapter 4, we formally define a metric to quantify temporal relaxation in Chapter 5 to address Research Question 2. This metric is used directly as an optimization variable in motion planning. We also propose a reactive planning algorithm that generates robot trajectories resilient to temporal violations caused by unforeseen events, which may render the original specifications infeasible. The system makes on-the-fly decisions when confronted with unforeseen events—such as potential collisions or environmental changes necessitating *task removal* or *partial satisfaction*—by initiating local corrections

or replanning trajectories to minimize temporal violations. The optimization process involves a trade-off between parts of the mission as per the user preferences.

The methods discussed so far mainly involve mixed-integer programs (e.g., MILP), which are computationally expensive given the intricacy of STL specifications. Chapter 6 investigates real-time applicability in STL specifications, addressing the Research Question 3. An online method is particularly needed to make decisions such that the STL specification over dynamic, noncooperative targets is satisfied, even when the robot is unaware of target movements. To address this, we construct Sequential CBFs to account for the worst-case bounds in target motion, achieving logical, temporal, and spatial requirements. The target motion—whether random, predefined, or adversarial—has minimal effect on mission success. Additionally, such changes in the mission setting and an increasing number of targets do not undermine real-time applicability. However, the proposed task scheduling framework currently performs efficiently with fewer than 10 concurrent tasks. Moreover, although the fragment of STL used in this real-time setting is more expressive than in the existing literature, it remains slightly more limited than the conventional syntax, as opposed to what is stated in Research Question 3. In practice, however, the benefits of the approach and the achieved level of expressiveness outweigh these limitations.

7.2 Future Work

We now discuss some future directions on the motion planning and control subject to temporal logic specifications. We also discuss our preliminary work on these subjects.

Interaction between High-Level Planners and Low-Level Controllers

The focus of this dissertation has primarily been on high-level motion planning for autonomous systems, with an emphasis on generating feasible and optimal trajectories without delving into the specifics of low-level control design. Although called "STL control synthesis," the methods presented in this dissertation tackle the motion planning and control problems simultaneously, finding the state trajectory that satisfies the STL specification and the control inputs that realize this trajectory. However, an important direction for future work is to explore how considering the dynamics of low-level controllers (e.g., LQR, MPC) could impact and potentially improve planning outcomes.

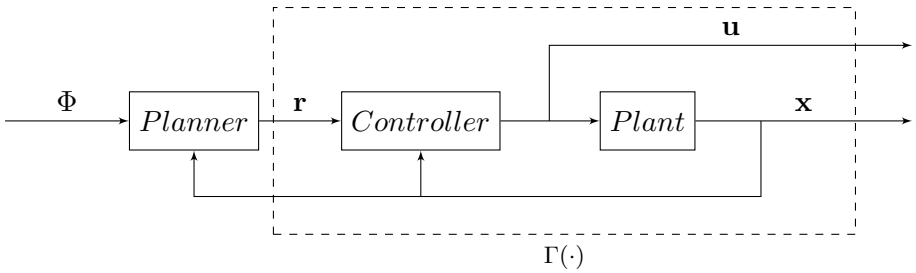


Figure 7.1: A block diagram of the planner and tracker. The parametric block $\Gamma(\cdot)$ incorporates both the low-level controller and the system dynamics [116].

The traditional separation between high-level planning and low-level control (e.g., [111–114]) often leads to challenges, particularly when the mission specifications are highly aggressive or the system is pushed close to its physical limits. In these scenarios, the planner may generate trajectories that are theoretically optimal but practically infeasible for the low-level controller to track accurately. Even if the high-level planner is aware of the poor tracker performance, in the traditional hierarchical approach, it has no means to adjust itself to, e.g., give the low-level tracker 'more room'. Such lack of adaptation in response to unmet specifications or even faults in the traditional architecture due to the unidirectional flow of information from the planner to the tracker has been recognized as a major open problem in the field [115].

In related work, the authors of [111] propose a planner and tracker co-design method of which success hinges on the ability to generate satisfactory plans via the Fly-by-Logic tool [15]— a high-level planner— under some additional constraints of the tracking error bounds. Our recent study [116] also presents a promising approach where the high-level planner and the low-level controller are designed in an iterative, unified process, by explicitly considering the closed-loop performance of the low-level controller. This is done by incorporating a parametric closed-loop model—the parametric block $\Gamma(\cdot)$ in Fig. 7.1—during the planning phase. In this framework, the high-level planner and the low-level controller communicate iteratively, adjusting the reference trajectories based on the controller's performance. This approach shows promise in bridging the gap between planning and control, allowing for more robust trajectory generation that accounts for real-world limitations such as actuator constraints and tracking errors.

In general, incorporating tracking control dynamics into high-level motion planning represents a significant opportunity for future research. Techniques bridging the gap between those have the potential to enhance the robustness and feasibility of generated trajectories, especially in challenging scenarios. Future work could explore sampling rate inconsistencies between planners and controllers and account for more realistic system dynamics for low-level control compared to planning.

Perception- and Estimation-Aware Motion Planning

Another important area for future exploration is the consideration of perception and state estimation performance during the high-level planning process. Traditional high-level planning approaches generally assume perfect state information, which rarely reflects real-world scenarios. In reality, the accuracy of state estimates is affected by various factors, including sensor quality, environmental conditions, and the nature of the trajectory itself. For example, aggressive maneuvers may improve the observability of certain system states, while more benign trajectories might lead to poorer estimation quality.

In motion planning literature with temporal logic constraints, process and measurement uncertainties are considered to improve the robustness of the trajectories (e.g., [117]). However, the effect of the generated trajectories on the estimation quality is not investigated in detail. In our recent study [118], we introduced a high-level planning strategy that explicitly accounts for the conditional observability of the state estimation (sensor fusion) algorithms as well as the mission specification encoded as a temporal logic formula. As state estimates are determined by blending sensor measurements with knowledge of the system's dynamics and improved by a realization of the intended trajectory [119], the proposed approach aims to generate trajectories in a way that they contribute to measurement quality and guarantee the satisfaction of the desired temporal logic specification.

Overall, incorporating state estimation performance into high-level motion planning represents an important opportunity for future research. Explicit consideration of the dynamics of state estimation in an estimation-aware planning framework can potentially enhance the applicability of the planned trajectories. That said, [118] uses a practical noise model that is dependent on the changes in the states, as depicted in Fig. 7.2. This is consistent with the sensing models in the real world [120,121], [122, Chapter 7]. However,

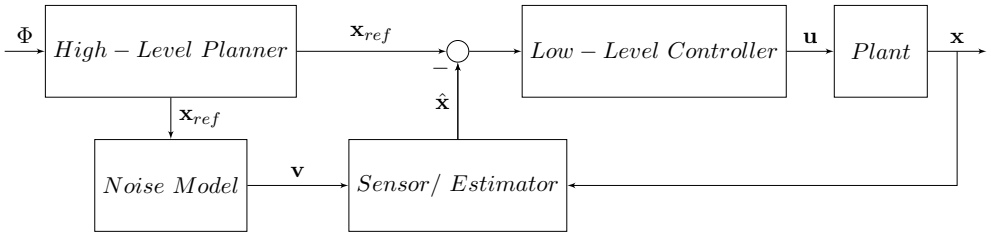


Figure 7.2: A block diagram illustrating the layers of the approach in [118] that is different from existing methods by including a module that incorporates trajectory-dependent estimation performance in the closed-loop.

the use of more realistic models that can be directly incorporated into optimization routines of high-level planning can be investigated further. Moreover, although agile maneuvers contribute to the state-estimation performance, they might be harder to track in practice with an additional cost of control effort. Therefore, investigating the trade-off between the control effort and estimation quality which may require the design of a new metric that comprises both sensing performance and efficiency can be another future direction.

References

[1] C. Baier and J. Katoen, *Principles of model checking*. MIT, 2008.

[2] A. Pnueli, "The temporal logic of programs," in *Symposium on Foundations of Computer Science*, pp. 46–57, IEEE, 1977.

[3] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[4] S. Karaman, R. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Conference Decision Control*, pp. 2117–2122, 2008.

[5] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[6] D. Aksaray, K. Leahy, and C. Belta, "Distributed multi-agent persistent surveillance under temporal logic constraints," *IFAC-PapersOnLine*, vol. 48, no. 22, pp. 174–179, 2015.

[7] L. Lindemann, J. Nowak, L. Schönbächler, M. Guo, J. Tumova, and D. Dimarogonas, "Coupled Multi-Robot Systems Under Linear Temporal Logic and Signal Temporal Logic Tasks," *IEEE Transactions on Control Systems Technology*, 2019.

[8] Y. Chen, X. Ding, and C. Belta, "Synthesis of distributed control and communication schemes from global LTL specifications," in *2011 50th IEEE Conference*

on Decision and Control and European Control Conference, pp. 2718–2723, IEEE, 2011.

[9] M. Guo, J. Tumova, and D. Dimarogonas, "Communication-free multi-agent control under local temporal tasks and relative-distance constraints," *IEEE Transactions on Automatic Control*, vol. 61, no. 12, pp. 3948–3962, 2016.

[10] K. Leahy, Z. Serlin, C. I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, "Scalable and robust algorithms for task-based coordination from high-level specifications (scratches)," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2516–2535, 2021.

[11] B. Lacerda and P. U. Lima, "Petri net based multi-robot task coordination from temporal logic specifications," *Robotics and Autonomous Systems*, vol. 122, p. 103289, 2019.

[12] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2020–2025, IEEE, 2005.

[13] Y. Zhou, D. Maity, and J. S. Baras, "Timed automata approach for motion planning using metric interval temporal logic," in *2016 European Control Conference (ECC)*, pp. 690–695, IEEE, 2016.

[14] V. Raman, A. Donzé, M. Maasoumy, R. Murray, A. Sangiovanni-Vincentelli, and S. Seshia, "Model predictive control with signal temporal logic spec.," in *Conference on Decision and Control*, pp. 81–87, 2014.

[15] Y. V. Pant, H. Abbas, R. A. Quaye, and R. Mangharam, "Fly-by-logic: control of multi-drone fleets with temporal logic objectives," in *International Conference on Cyber-Physical Systems*, pp. 186–197, IEEE, 2018.

[16] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Control synthesis using signal temporal logic specifications with integral and derivative predicates," in *American Control Conference (ACC)*, pp. 4873–4878, IEEE, 2021.

[17] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Distributed Planning of Multi-Agent Systems with Coupled Temporal Logic Specifications," in *AIAA Scitech*, p. 1123, 2021.

[18] R. Peterson, A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoglu, "Decentralized safe reactive planning under twtl specifications," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6599–6604, IEEE, 2020.

[19] R. Peterson, A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Distributed safe planning for satisfying minimal temporal relaxations of twtl specifications," *Robotics and Autonomous Systems*, vol. 142, p. 103801, 2021.

[20] A. Pantazides, D. Aksaray, and D. Gebre-Egziabher, "Satellite mission planning with signal temporal logic specifications," in *AIAA SCITECH 2022 Forum*, p. 1091, 2022.

[21] D. Aksaray, "Resilient satisfaction of persistent and safety specifications by autonomous systems," in *AIAA Scitech Forum*, p. 1124, 2021.

[22] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Control barrier functions with actuation constraints under signal temporal logic specifications," in *2022 European Control Conference (ECC)*, pp. 162–168, IEEE, 2022.

[23] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-time Systems*, vol. 2, no. 4, pp. 255–299, 1990.

[24] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *Journal of the ACM (JACM)*, vol. 43, no. 1, pp. 116–146, 1996.

[25] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Proc. Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pp. 152–166, 2004.

[26] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 92–106, 2010.

[27] Y. V. Pant, H. Abbas, and R. Mangharam, "Smooth operator: Control using the smooth robustness of temporal logic," in *Conference on Control Tech. and Applications*, pp. 1235–1240, 2017.

[28] V. Kurtz and H. Lin, "Mixed-integer programming for signal temporal logic with fewer binary variables," *IEEE Control Systems Letters*, vol. 6, pp. 2635–2640, 2022.

[29] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Planning of heterogeneous multi-agent systems under signal temporal logic specifications with integral predicates," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1375–1382, 2021.

[30] N. Mehdipour, C. I. Vasile, and C. Belta, "Specifying user preferences using weighted signal temporal logic," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2006–2011, 2020.

[31] D. Aksaray, A. Jones, Z. Kong, M. Schwager, and C. Belta, "Q-learning for robust satisfaction of signal temporal logic specifications," in *Conference on Decision and Control (CDC)*, pp. 6565–6570, IEEE, 2016.

[32] G. Silano, T. Baca, R. Penicka, D. Liuzza, and M. Saska, "Power line inspection tasks with multi-aerial robot systems via signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4169–4176, 2021.

[33] S. Uzun, P. Elango, P.-L. Garoche, and B. Acikmese, "Optimization with temporal and logical specifications via generalized mean-based smooth robustness measures," *arXiv preprint arXiv:2405.10996*, 2024.

[34] A. Caballero and G. Silano, "A stl motion planner for bird diverter installation tasks with multi-robot aerial systems," *IEEE Access*, 2023.

[35] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE control systems letters*, vol. 3, no. 1, pp. 96–101, 2018.

[36] J. Ouaknine and J. Worrell, "Some recent results in mtl," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 1–13, 2008.

[37] A. Dokhanchi, B. Hoxha, and G. Fainekos, "On-line monitoring for temporal logic robustness," in *International Conference on Runtime Verification*, pp. 231–246, Springer, 2014.

[38] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 772–779, IEEE, 2015.

[39] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2016.

[40] X. Xu, "Constrained control of input–output linearizable systems using control sharing barrier func.," *Automatica*, vol. 87, pp. 195–201, 2018.

[41] T. Akazaki and I. Hasuo, "Time robustness in MTL and expressivity in hybrid system falsification," in *International Conference on Computer Aided Verification*, pp. 356–374, 2015.

[42] A. Rodionova, E. Bartocci, D. Nickovic, and R. Grosu, "Temporal logic as filtering," in *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pp. 11–20, 2016.

[43] S. Silvetti, L. Nenzi, E. Bartocci, and L. Bortolussi, "Signal convolution logic," in *International Symposium on Automated Tech. for Verification and Analysis*, pp. 267–283, Springer, 2018.

[44] Z. Xu and A. A. Julius, "Census signal temporal logic inference for multiagent group behavior analysis," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 264–277, 2016.

[45] D. Sadigh and A. Kapoor, "Safe control under uncertainty with probabilistic signal temporal logic," in *Robotics: Science and Systems*, 2016.

[46] L. Brim, P. Dluhoš, D. Šafránek, and T. Vejpustek, "STL∗: Extending signal temporal logic with signal-value freezing operator," *Information and Computation*, vol. 236, pp. 52–67, 2014.

[47] I. Haghighi, N. Mehdipour, E. Bartocci, and C. Belta, "Control from stl specifications with smooth cumulative quantitative semantics," in *Conference on Decision and Control*, pp. 4361–4366, 2019.

[48] N. Mehdipour, C. I. Vasile, and C. Belta, "Arithmetic-geometric mean robustness for control from signal temporal logic specifications," in *American Control Conference*, pp. 1690–1695, 2019.

[49] L. Lindemann and D. V. Dimarogonas, "Robust control for signal temporal logic specifications using discrete average space robustness," *Automatica*, vol. 101, pp. 377–387, 2019.

[50] A. Donzé, O. Maler, E. Bartocci, D. Nickovic, R. Grosu, and S. Smolka, "On temporal logic and signal processing," in *International Symposium on Automated Technology for Verification and Analysis*, pp. 92–106, Springer, 2012.

[51] P. Varnai and D. V. Dimarogonas, "On robustness metrics for learning stl tasks," *arXiv preprint arXiv:2003.06041*, 2020.

[52] C. Belta and S. Sadraddini, "Formal methods for control synthesis: An optimization perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 115–140, 2019.

[53] H. Abbas, A. Winn, G. Fainekos, and A. A. Julius, "Functional gradient descent method for metric temporal logic specifications," in *2014 American Control Conference*, pp. 2312–2317, IEEE, 2014.

[54] Gurobi-Optimization, "Gurobi Optimizer Ref. Manual," 2024.

[55] I. Haghighi, S. Sadraddini, and C. Belta, "Robotic swarm control from spatio-temporal specifications," in *IEEE Conference on Decision and Control (CDC)*, pp. 5708–5713, 2016.

[56] Y. Sahin, P. Nilsson, and N. Ozay, "Provably-correct coordination of large collections of agents with counting temporal logic constraints," in *Proceedings of the 8th International Conference on Cyber-Physical Systems*, pp. 249–258, ACM, 2017.

[57] Y. E. Sahin, P. Nilsson, and N. Ozay, "Synchronous and asynchronous multi-agent coordination with cltl+ constraints," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 335–342, IEEE, 2017.

[58] M. Cai, K. Leahy, Z. Serlin, and C. I. Vasile, "Probabilistic coordination of heterogeneous teams from capability temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1190–1197, 2022.

[59] A. M. Jones, K. Leahy, C. Vasile, S. Sadraddini, Z. Serlin, R. Tron, and C. Belta, "Scratchs: Scalable and robust algorithms for task-based coordination from high-level specifications," in *International Symposium of Robotics Research*, 2019.

[60] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Energy-aware planning of heterogeneous multi-agent systems for serving cooperative tasks with temporal logic specifications," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8659–8665, IEEE, 2023.

[61] C. Kurtz and H. Abbas, "Fairfly: A fair motion planner for fleets of autonomous uavs in urban airspace," in *23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2020.

[62] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[63] B. Bethke, J. P. How, and J. Vian, "Group health management of uav teams with applications to persistent surveillance," in *2008 American Control Conference*, pp. 3145–3150, IEEE, 2008.

[64] G. Shi, N. Karapetyan, A. B. Asghar, J.-P. Reddinger, J. Dotterweich, J. Humann, and P. Tokekar, "Risk-aware uav-ugv rendezvous with chance-constrained markov decision process," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 180–187, 2022.

[65] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997–3004, IEEE, 2014.

[66] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt, "The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems," *IEEE Control Systems Magazine*, vol. 40, no. 1, pp. 26–44, 2020.

[67] Z. Lin and J. S. Baras, "Optimization-based motion planning and runtime monitoring for robotic agent with space and time tolerances," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1874–1879, 2020.

[68] A. Rodionova, L. Lindemann, M. Morari, and G. J. Pappas, "Time-robust control for stl specifications," in *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 572–579, 2021.

[69] L. Lindemann, A. Rodionova, and G. J. Pappas, "Temporal robustness of stochastic signals," *arXiv preprint arXiv:2202.02583*, 2022.

[70] A. Rodionova, L. Lindemann, M. Morari, and G. J. Pappas, "Combined left and right temporal robustness for control under stl specifications," *Control Systems Letters*, vol. 7, pp. 619–624, 2022.

[71] C. I. Vasile, D. Aksaray, and C. Belta, "Time window temporal logic," *Theoretical Computer Science*, vol. 691, pp. 27–54, 2017.

[72] S. Ghosh, D. Sadigh, P. Nuzzo, V. Raman, A. Donzé, A. L. Sangiovanni-Vincentelli, S. S. Sastry, and S. A. Seshia, "Diagnosis and repair for synthesis from stl specifications," in *Proceedings of the 19th International Conference on Hybrid Systems: Comp. and Control*, pp. 31–40, 2016.

[73] A. T. Buyukkocak and D. Aksaray, "Temporal relaxation of signal temporal logic specifications for resilient control synthesis," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 2890–2896, IEEE, 2022.

[74] G. A. Cardona and C. I. Vasile, "Partial satisfaction of stl specifications for coordination of multi-robot systems," in *International Workshop on the Algorithmic Foundations of Robotics*, pp. 223–238, Springer, 2022.

[75] G. A. Cardona and C. I. Vasile, "Preferences on partial satisfaction using weighted stl specifications," in *European Control Conference (ECC)*, 2023.

[76] B. Hoxha and G. Fainekos, "Planning in dynamic environments through temporal logic monitoring," in *Workshops at the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[77] S. Saha and A. A. Julius, "An milp approach for real-time optimal controller synthesis with metric temporal logic specifications," in *2016 American Control Conference (ACC)*, pp. 1105–1110, IEEE, 2016.

[78] M. Ryll, J. Ware, J. Carter, and N. Roy, "Efficient trajectory planning for high speed flight in unknown environments," in *International Conference on Robotics and Automation (ICRA)*, pp. 732–738, IEEE, 2019.

[79] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 1002–1028, 2020.

[80] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *International Conference on Robotics and Automation (ICRA)*, pp. 12618–12624, 2021.

[81] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in matlab," in *In Proceedings of the CACSD Conference*, (Taipei, Taiwan), 2004.

[82] S. Wilson, P. Glotfelter, S. Mayya, G. Notomista, Y. Emam, X. Cai, and M. Egerstedt, "The robotarium: Automation of a remotely accessible, multi-robot testbed," *Robotics and Automation Letters*, vol. 6, no. 2, pp. 2922–2929, 2021.

[83] F. Ferraguti, C. T. Landi, S. Costi, M. Bonfè, S. Farsoni, C. Secchi, and C. Fantuzzi, "Safety barrier functions and multi-camera tracking for human–robot shared environment," *Robotics and Autonomous Systems*, vol. 124, p. 103388, 2020.

[84] A. D. Ames, G. Notomista, Y. Wardi, and M. Egerstedt, "Integral control barrier functions for dynamically defined control laws," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 887–892, 2020.

[85] Y. Huang, S. Z. Yong, and Y. Chen, "Guaranteed vehicle safety control using control-dependent barrier functions," in *American Control Conference (ACC)*, pp. 983–988, 2019.

[86] A. Li, L. Wang, P. Pierpaoli, and M. Egerstedt, "Formally correct composition of coordinated behaviors using control barrier certificates," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3723–3729, 2018.

[87] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 363–374, 2020.

[88] L. Niu and A. Clark, "Control barrier functions for abstraction-free control synthesis under temporal logic constraints," in *Conference on Decision and Control (CDC)*, pp. 816–823, IEEE, 2020.

[89] W. Xiao, C. A. Belta, and C. G. Cassandras, "High order control lyapunov-barrier functions for temporal logic specifications," in *2021 American Control Conference (ACC)*, pp. 4886–4891, IEEE, 2021.

[90] K. Garg and D. Panagou, "Control-lyapunov and control-barrier functions based quadratic program for spatio-temporal specifications," in *Conference on Decision and Control*, pp. 1422–1429, IEEE, 2019.

[91] D. Gundana and H. Kress-Gazit, "Event-based signal temporal logic synthesis for single and multi-robot tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3687–3694, 2021.

[92] A. Zehfroosh and H. G. Tanner, "Non-smooth control barrier navigation functions for stl motion planning," *Frontiers in Robotics and AI*, vol. 9, 2022.

[93] K. Ramamritham and J. A. Stankovic, "Dynamic task scheduling in hard real-time distributed systems," *IEEE software*, vol. 1, no. 3, p. 65, 1984.

[94] M. L. Dertouzos and A. K. Mok, "Multiprocessor online scheduling of hard-real-time tasks," *IEEE Transactions on software engineering*, vol. 15, no. 12, pp. 1497–1506, 1989.

[95] E. Nunes, M. Manner, H. Mitiche, and M. Gini, "A taxonomy for task allocation problems with temporal and ordering constraints," *Robotics and Autonomous Systems*, vol. 90, pp. 55–70, 2017.

[96] D. M. Le, X. Luo, L. J. Bridgeman, M. M. Zavlanos, and W. E. Dixon, "Single-agent indirect herding of multiple targets using metric temporal logic switching," in *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 1398–1403, IEEE, 2020.

[97] A. T. Buyukkocak, D. Aksaray, and Y. Yazıcıoğlu, "Sequential control barrier functions for mobile robots with dynamic temporal logic specifications," *Robotics and Autonomous Systems*, vol. 176, p. 104681, 2024.

[98] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics: A survey," *Autonomous Robots*, vol. 31, pp. 299–316, 2011.

[99] S. Bharadwaj, R. Dimitrova, J. Quattrociocchi, and U. Topcu, "Synthesis of strategies for autonomous surveillance on adversarial targets," *Robotics and Autonomous Systems*, vol. 153, p. 104084, 2022.

[100] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *IEEE transactions on robotics and automation*, vol. 18, no. 5, pp. 662–669, 2002.

[101] V. G. Lopez, F. L. Lewis, Y. Wan, E. N. Sanchez, and L. Fan, "Solutions for multiagent pursuit-evasion games on communication graphs: Finite-time capture and asymptotic behaviors," *IEEE Transactions on Automatic Control*, vol. 65, no. 5, pp. 1911–1923, 2019.

[102] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8129–8136, 2021.

[103] K. Granstrom, M. Baum, and S. Reuter, "Extended object tracking: Introduction, overview and applications," *Journal of Advances in Information Fusion*, vol. 12, no. 2, pp. 139–174, 2016.

[104] L. Vandenberghe and S. Boyd, *Convex optimization*, vol. 1. Cambridge University Press Cambridge, 2004.

[105] D. Angeli and E. D. Sontag, "Monotone control systems," *IEEE Transactions on automatic control*, vol. 48, no. 10, pp. 1684–1698, 2003.

[106] S. Sadraddini and C. Belta, "Formal synthesis of control strategies for positive monotone systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 2, pp. 480–495, 2018.

[107] M. Charitidou and D. V. Dimarogonas, "Barrier function-based model predictive control under signal temporal logic specifications," in *European Control Conference*, 2021.

[108] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear opt. and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[109] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3299–3304, IEEE, 2017.

[110] P. Yu, X. Tan, and D. V. Dimarogonas, "Continuous-time control synthesis under nested signal temporal logic specifications," *IEEE Transactions on Robotics*, 2024.

[111] Y. V. Pant, H. Yin, M. Arcak, and S. A. Seshia, "Co-design of control and planning for multi-rotor uavs with signal temporal logic specifications," in *2021 American Control Conference (ACC)*, pp. 4209–4216, IEEE, 2021.

[112] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, 2016.

[113] S. Singh, M. Chen, S. L. Herbert, C. J. Tomlin, and M. Pavone, "Robust tracking with model mismatch for fast and safe planning: an sos optimization approach," *arXiv preprint arXiv:1808.00649*, 2018.

[114] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 1517–1522, IEEE, 2017.

[115] P. Varaiya, "A question about hierarchical systems," in *System Theory*, pp. 313–324, Springer, 2000.

[116] A. T. Buyukkocak, P. Seiler, D. Aksaray, and V. Gupta, "Iterative planner/controller design to satisfy signal temporal logic specifications," in *2023 American Control Conference (ACC)*, pp. 3516–3522, IEEE, 2023.

[117] L. Lindemann, M. Cleaveland, Y. Kantaros, and G. J. Pappas, "Robust motion planning in the presence of estimation uncertainty," in *60th IEEE Conference on Decision and Control (CDC)*, pp. 5205–5212, 2021.

[118] A. T. Buyukkocak, Y. Hu, A. Taheri, D. Aksaray, and D. Gebre-Egziabher, "State-estimation-aware planning for autonomous systems with temporal logic specifications," in *AIAA SCITECH 2023 Forum*, p. 2665, 2023.

[119] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[120] D. Goshen-Meskin and I. Bar-Itzhack, "Observability analysis of piece-wise constant systems. i. theory. aerospace and electronic systems," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 28, no. 4, pp. 1056–1067, 1992.

[121] D. Goshen-Meskin and I. Bar-Itzhack, "Observability analysis of piece-wise constant systems. ii. application to inertial navigation in-flight alignment (military applications)," *IEEE Transactions on Aerospace and Electronic systems*, vol. 28, no. 4, pp. 1068–1075, 1992.

[122] D. Gebre-Egziabher, M. Petovello, and D. Bevly, "Integration of gnss and ins: Part 2 (case studies)," *GNSS Applications and Methods (Scot Gleason and Demoz Gebre-Egziabher, eds.), Artech House, Norwood, MA*, pp. 177–190, 2009.

Appendix A

Abbreviations and Symbols

A.1 List of Abbreviations

Table A.1: Abbreviations and Acronyms.

Abbreviations and Acronyms	**Meaning**
TL	Temporal Logic
LTL	Linear Temporal Logic
STL	Signal Temporal Logic
MTL	Metric Temporal Logic
MITL	Metric Interval Temporal Logic
TWTL	Time Window Temporal Logic
PNF	Positive Normal Form
DNF	Disjunctive Normal Form
RRT	Rapidly Exploring Random Tree
VRP	Vehicle Routing Problem
QP	Quadratic Program
MIP	Mixed-Integer Program
MILP	Mixed-Integer Linear Program
MIQP	Mixed-Integer Quadratic Program
NLP	Nonlinear Program

Continued on next page

Continued from previous page

Abbreviations and Acronyms	**Meaning**
CBF	Control Barrier Function
TV-CBF	Time-Varying Control Barrier Function
MPC	Model Predictive Control
LQR	Linear Quadratic Regulator
UAV	Unmanned Aerial Vehicle

A.2 List of Symbols

$\top$	True
$\bot$	False
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_{\geq 0}$	Set of non-negative real numbers
$\mathbb{R}^+$	Set of positive real numbers
$\mathbb{R}^n$	Set of n-dimensional real-valued vectors
$\mathbb{R}^{m\times n}$	Set of real-valued $m \times n$ matrices
$\mathbb{Z}$	Set of integer numbers
$\mathbb{Z}_{\geq 0}$	Set of non-negative integer numbers
$\mathbb{Z}^+$	Set of positive integer numbers
$\mathbb{Z}^n$	Set of n-dimensional integer-valued vectors
$\mathbb{Z}^{m\times n}$	Set of integer-valued $m \times n$ matrices
$\mathbf{1}$	Vector of appropriate size consisting of ones
$\|\cdot\|_1$	l_1 (sum) norm of a vector
$\|\cdot\|_2$	l_2 (Euclidean) norm of a vector
$\|\cdot\|$	Absolute value or set cardinality
$(\cdot)_+$	Projection operator for $\mathbb{R} \to \mathbb{R}_{\geq 0}$
$\mathcal{X} \oplus \mathcal{Y}$	Minkowski sum of two sets $\mathcal{X}$ and $\mathcal{Y}$
$\mathcal{A}$	Adjacency matrix of a graph
$\mathcal{N}(v)$	Set of nodes neighboring v

Continued on next page

Continued from previous page

α	Extended class $\mathcal{K}$ function with $\mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$
$U(a, b)$	Uniform distribution over $[a, b]$
$N(\mu, \sigma^2)$	Normal (Gaussian) distibution with mean μ and variance σ^2