Formal Methods for Multi-Robot Systems: Scalable and Robust Planning with Temporal Logic and Partial Satisfaction

Formal Methods for Multi-Robot Systems: Scalable and Robust Planning with Temporal Logic and Partial Satisfaction

by

Gustavo Andres Cardona Calderon

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Mechanical Engineering

Lehigh University

May 2025

Dissertation Signature Sheet

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Gustavo Andres Cardona Calderon

Planning for Multirobot Systems: a Formal Methods Approach

_____

Date

_____

Dissertation Director

_____

Approved Date

Committee Members:

_____

Cristian-Ioan Vasile, Committee Chair

_____

Subhrajit Bhattacharya

_____

Nader Motee

_____

David Saldaña

_____

Hadas Kress-Gazit

Acknowledgements

This thesis is a testament to the invaluable guidance and support of my advisor, Professor Cristian-Ioan Vasile (Cristi). From the first day I met Cristi, he fostered an environment that encouraged me to grow and flourish. I am deeply grateful for his mentorship, which has shaped my intellectual and personal journey and my academic development. I have cherished every moment of the countless meetings we have had over the years—our discussions were always engaging, and I have learned an immeasurable amount from them (I cannot remember a meeting where I did not learn something from you). Many great ideas and papers were born from sessions at his office whiteboard, and those are still some of my favorite moments at Lehigh. I am especially grateful for his emphasis on fundamental mathematics and for improving my coding skills, which have given me the confidence to tackle challenging research problems.

I also sincerely thank Professor David Saldaña for his mentorship, encouragement, and support throughout my academic endeavors. His guidance at every stage of this journey has been invaluable.

My heartfelt thanks go to the members of my doctoral committee,mittee, Professor Subhrajit Bhattacharya, Professor David Saldaña, Professor Nader Motee, and Professor Hadas Kress-Gazit, for their invaluable feedback and insights throughout my doctoral studies.

I am deeply grateful to my collaborators: Cristian-Ioan Vasile, Disha Kamale, David Saldaña, Kaier Liang, Kevin Leahy, Diego S. D'Antonio, and Rafael Fierro. I have learned

tremendously through our collaborations, and working with them has been an incredibly rewarding and enjoyable experience.

I also want to express my gratitude to my wonderful lab mates at the AIR Lab: Disha Kamale, Diego S. D'Antonio (Gogo), Kaier Liang, Guangyi Liu, Mingyu Cai, Xiaolong Wang, Jinda Cui, Jiawei Xu, Alp Sahin, and my dear friends at Lehigh—Sebas, Arte, Liz, Johanna, Diego[3], Juan and Mila, Aleja, Keri, and Rachel. Their friendship and camaraderie made my doctoral years not only intellectually stimulating but also immensely enjoyable.

I extend my deepest gratitude to my parents, Gustavo and Yaneth, my sister Paula, and my uncle Juan. Their unwavering love, support, and sacrifices have shaped the person I am today and allowed me to be where I am today. Their encouragement and belief in me have been my driving force, giving me the strength and determination to overcome challenges and pursue my ambitions. I am forever grateful for their endless support and unwavering faith in my endeavors.

Finally, to my wife, Dani, thank you for your endless support and encouragement, for pushing me to pursue dreams that I once thought were beyond my reach, and for being my anchor in times of uncertainty and doubt. I could not have done this without you.

Table of Contents

List of Figures

List of Tables

Abstract

This dissertation addresses the challenges posed by the increasing complexity of multi-robot systems by proposing a formal methods framework that ensures scalable, robust, and customizable mission planning. The primary contribution is the integration of high-level temporal logic specifications, specifically Signal Temporal Logic (STL) and its weighted extension (wSTL +) withh Mixed-Integer Linear Programming (MILP) formulations. This integration enables the automatic synthesis of controllers that meet various mission objectives. Novel MILP encodings have been developed to enhance the expressivity and computational efficiency of STL and wSTL, allowing their application in large-scale scenarios. The framework is also extended to model complex multiagent dynamics, including swarm behaviors, modular robot reconfigurations, heterogeneous team coordination, and resource-constrained logistics. Furthermore, this work introduces a systematic approach to handling infeasible specifications through partial satisfaction, ensuring that mission-critical objectives are prioritized even when some constraints cannot be fully met. Overall, this dissertation advances the state of the art in multi-robot planning by combining formal temporal logic reasoning with optimization-based control synthesis, providing a principled and practical solution for real-world uncertainties and specification infeasibility.

Chapter 1

Introduction and General Overview

1.1 Introduction

Multi-robot systems have become increasingly important in tackling complex real-world challenges[131, 61]. They can perform tasks that exceed the capacity of individual robots and can operate reliably in dynamic and uncertain environments. These systems are reliable when considering decision-making and actuation, enhancing efficiency, fault tolerance, and scalability across various applications [172, 43]. Some examples include disaster response, search and rescue operations, precision agriculture, warehouse automation, and urban logistics [45, 117, 23, 28]. Multi-robot systems can significantly reduce response times and improve coverage by enabling the parallel execution of tasks and facilitating coordinated actions [40]. This capability is especially crucial in time-sensitive scenarios involving the coordination of tasks that may happen sequentially or simultaneously. Additionally, integrating multi-robot systems allows for robust performance in hazardous or hard-to-reach areas, minimizing risks to human operators and enabling continuous operation under challenging conditions [156, 125].

When planning for multirobot systems, two main problems are pathfinding for individual agents and task allocation [70, 140]. In recent decades, multi-robot planning has shifted

from traditional algorithms to addressing the complexities of real-world applications. Initially, approaches relied on well-established pathfinding methods such as A* and Dijkstra's algorithm [153, 148], task allocation techniques like the Hungarian algorithm, and auction-based strategies [39, 83]. These methods have been effective in more straightforward or structured environments where the scale of problems and constraints is more manageable.

However, two significant challenges have arisen as multi-robot systems are deployed in increasingly complex dynamics, richer expressivity of requirements, and uncertain environments. First, the joint state space of multiple interacting robots grows exponentially, making centralized solutions computationally impractical. Second, many traditional methods do not naturally incorporate the temporal and dynamic constraints critical for ensuring safety and mission success in real-world scenarios. Recent research has focused on formal methods to address these challenges [56, 55, 58, 142]. By utilizing temporal logic to specify complex mission requirements and safety constraints, researchers are developing integrated frameworks that not only create efficient paths and allocate tasks but also rigorously verify that these plans meet high-level specifications. This combination of algorithmic efficiency with formal verification represents a significant advancement toward more reliable and scalable multi-robot systems, paving the way for advanced applications in dynamic and uncertain environments [126, 100].

Temporal logics, including Linear Temporal Logic (LTL) and Signal Temporal Logic (STL), provide powerful languages for capturing time-dependent behaviors and constraints critical for robotic systems [134, 112]. STL, in particular, has gained prominence due to its ability to manage continuous signals, use the explicit definition of time, and quantify robustness amid uncertainties [108, 109]. Early research on automata-based synthesis for temporal logic specifications demonstrated how high-level tasks could be translated into automata to facilitate controller synthesis [38, 149]. Guaranteeing correct by construction controllers generating inputs that satisfy specification requirements. However, these methods frequently encounter the state explosion problem, which limits their practicality for

large-scale or multi-robot systems. More recent approaches have turned to optimization-based methods as a solution. Mixed Integer Linear Programming (MILP) formulations present an attractive alternative, as they can seamlessly encode temporal logic constraints and leverage efficient solvers. For instance, studies by [133, 90, 150] have illustrated the effectiveness of MILP-based approaches in handling STL specifications. Given the computational challenges in MILP formulations for temporal logic, a significant research effort has been directed toward improving encoding efficiency. Traditional formulations often suffer from an explosion in the number of binary variables, leading to increased computational times and reduced scalability. Chapter 2 of this thesis introduces a novel MILP encoding that exploits structural properties of temporal logic specifications to reduce the number of required binary variables. This makes it more suitable for large-scale multi-robot planning.

Planning for multi-robot systems involves a complex set of challenges that extend well beyond traditional static task allocation and pathfinding [105]. The inherent complexities of these systems require mission descriptions that address more than just goal locations; they must also incorporate time-varying constraints, evolving task requirements, and the intricate interactions among individual robots. For instance, planning for a swarm of robots, where emergent collective behaviors take precedence, is fundamentally different from coordinating modular robots that can be reconfigured to exhibit varied emerging capabilities or performance, managing heterogeneous fleets with differing abilities, or orchestrating operations within the structured yet complex environment of a logistics warehouse. Each scenario presents unique demands on the planning process: mission specifications must adapt to evolving constraints and ensure successful task completion, while motion requirements—such as collision avoidance, energy limitations, and strict temporal deadlines—must be meticulously upheld. Additionally, the interaction constraints governing how robots communicate, collaborate, or allocate resources introduce another layer of complexity that challenges conventional planning algorithms. In Chapter 3 of this thesis, we ad-

dress the critical issue of capturing these multifaceted operational intricacies efficiently and comprehensively. By developing models that effectively integrate mission specifications, motion constraints, and interaction protocols, our approach provides a unified framework capable of managing the diverse and demanding scenarios typical of modern multi-robot systems.

When considering real-world multi-robot mission planning, it is common for specifications to be either infeasible or internally conflicting. Uncertainties in the environment, dynamic changes during operations, user errors, and competing priorities among different tasks can create situations where strict adherence to all constraints is impractical. Attempting to satisfy these constraints fully often results in overly conservative plans that may either fail to execute or underutilize the system's potential. It is essential to allow for relaxation or partial satisfaction of the given specifications to address these challenges. Planners can develop more adaptable and efficient strategies by prioritizing certain constraints over others or accepting a degree of deviation from ideal outcomes. This approach recognizes that achieving a "good enough" solution that satisfies as many constraints as possible or aligns with user-defined preferences can be far more valuable than striving for a perfect yet unattainable plan. In Chapter 4, we introduce a framework for the partial satisfaction of temporal logic languages. This framework is designed to systematically handle conflicting or infeasible mission specifications by maximizing the fulfillment of the most critical requirements. Our method incorporates user preferences into the planning process, allowing for a more flexible and robust response to the inherent uncertainties of real-world environments. This ultimately leads to more practical and effective deployments of multi-robot systems.

1.2 General Overview

This thesis presents a comprehensive framework for synthesizing controllers coordinating complex multi-robot systems dynamics while subject to temporal logic specifications. Central to this work is an optimization-based approach—specifically, Mixed Integer Linear Programming (MILP)—transforming high-level temporal logic specifications into concrete control strategies. Recent advancements in commercial solvers, such as Gurobi, incorporating relaxation techniques and specialized algorithms, have made it possible to address these complex MILP formulations. Thus, they enable real-time tracking and resolution of intricate planning challenges. A diagram capturing the general overview of this thesis and organization is shown in Fig. 1.2.1

This thesis explores various temporal logic languages, including Metric Temporal Logic (MTL), Signal Temporal Logic (STL), and a weighted extension known as wSTL. In Chapter 2, I formulate an MILP encoding that captures the semantics of wSTL[24], including the modulation of robustness imposed by weights that map user preferences and importance over choices. Additionally, I introduce a novel, disjunction-centric Mixed-Integer Linear Programming (MILP) [25] encoding that significantly reduces the number of variables needed to represent the semantics of these specifications. This encoding creates a more efficient formulation by leveraging the inherent structural properties of the temporal logic formulas.

This approach is implemented in our modular and versatile Python tool called PyTeLo [26], which automates the translation of temporal logic specifications into MILP formulations. PyTeLo abstracts the semantics by accepting a string representation of the temporal logic specification, along with optional system dynamics. It utilizes an ANTLR-generated parser to construct an Abstract Syntax Tree (AST). This AST is then recursively encoded into a MILP that can be solved using commercial solvers like Gurobi. This process streamlines the synthesis of controllers derived from high-level mission descriptions.

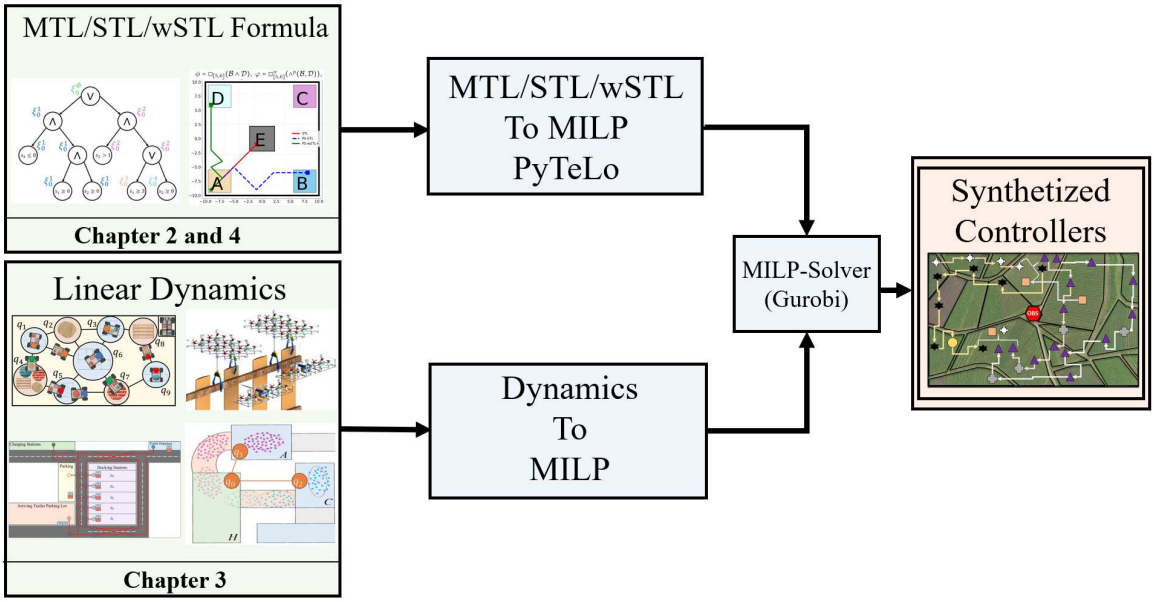


Figure 1.2.1: Thesis diagram.

Chapter 3 builds on this approach by exploring the modeling of complex multi-robot systems. In this chapter, I present scenarios involving swarms of robots [27], modular robots [29], and heterogeneous robots that are tasked with resource transportation [31] and coordinated logistics in a warehouse setting. These challenges can be framed as network flow problems. Many of the planning issues—such as pathfinding and task allocation—can initially be represented as Integer Linear Programs (ILPs) and then relaxed to Linear Programs (LPs) due to their unimodularity properties. This relaxation maintains optimal solutions while reducing computational complexity. Building upon this, we incorporate additional factors, such as inter-robot interactions, motion constraints, and task satisfaction requirements within the network flow framework. This approach offers a scalable method for managing the intricate dynamics of modern multi-robot systems.

Chapter Four addresses the critical issue of partial satisfaction in mission planning. It acknowledges that strict adherence to complete specification satisfaction can be impractical, especially when faced with conflicting or unachievable mission constraints. To tackle this, I developed a framework that maximizes satisfaction based on user-defined preferences. This approach enables the system to fulfill as many aspects of the mission as possi-

ble, ensuring robust performance even under challenging conditions.

Together, these contributions advance the state-of-the-art in multi-robot planning by integrating formal temporal logic specifications with efficient Mixed Integer Linear Programming (MILP) formulations and scalable modeling techniques. The resulting framework not only connects high-level mission planning with low-level control synthesis but also opens up opportunities for practical applications in areas such as autonomous logistics, coordinated swarm robotics, and beyond.

1.3 Summary of Contributions

The contributions of this thesis are listed bellow by chapter as follows:

Ch2: We propose an MILP formulation that captures the qualitative and quantitative semantics of wSTL developed in [111]. The proposed MILP encoding captures weighted traditional robustness to quantify satisfaction.

Ch2: We propose an efficient disjunction-centric MILP formulation for STL and wSTL specifications that results in fewer binary decision variables and constraints in the encoding.

Ch2: We extend the definition of wSTL [111] to include *Until* and *Release* operators.

Ch2: We provide the correctness proofs for all proposed encodings.

Ch2: We demonstrate the versatility of wSTL formulae and the functionality of weights to modify the solution of an equivalent STL formula. Additionally, we present case studies for control synthesis using the proposed wSTL MILP formulation and compare it against equivalent STL formulae. We perform sensitivity analysis to characterize the effect of weight modulation on the nature of synthesized trajectories.

Ch2: Finally, we present the time performance comparison between the proposed disjunction-centric MILP formulation and the standard STL encodings.

Ch3: We propose an efficient MILP approach to solve the planning problem, considering swarm splitting and merging behaviors for satisfying tasks while constraining the maximum number of simultaneous existing swarms and minimizing unnecessary splitting or traveling. Additionally, standard flow dynamics equations are modified to flow inequalities considering that node and edge equations could not be balanced due to the merging and splitting actions.

Ch3: We develop a randomized distributed method to split large swarms when communication between agents is impossible.

Ch3: We develop a method to assign splitting fractions such that the sub-swarms are balanced, given a motion plan from the MILP.

Ch3: We propose and formalize a planning problem for modular aerial robots with heterogeneous capabilities and configurations tasked with performing timed temporal logic missions. The missions involve tasks that can be performed only by a subset of robot configurations and may require robots to reconfigure during the mission.

Ch3: We propose an efficient MILP approach that minimizes the total energy consumption while satisfying the MTL mission specification, robot motion, and reconfiguration constraints.

Ch3: Extending CaTL [75] to include tasks that require both agents and resources. Defining and modeling resource transportation that can accommodate various types of resources (divisible, indivisible, packets, etc.) and considers different storage types (uniform, compartmental) and agent capacities.

Ch3: Defining Multi-Class Temporal Logic (mcSTL), an extension of STL that considers

predicates from multiple semantic classes. We also define a multi-robustness score that enables the quantification of satisfaction for each predicate class.

Ch3: Applying the multi-robustness mcSTL in the context of CaTL to independently compute the robustness of robots and resources.

Ch3: Proposing an efficient MILP-based planning approach that captures the CaTL specification, robots and resources dynamics, and transportation constraints. The MILP aims to maximize disjoint robots' and resources' robustness while minimizing spurious motion.

Ch4: An extension to wSTL [10], referred to as wSTL+, including exclusive operators such as exclusive conjunction, exclusive disjunction, exclusive always, and exclusive eventually.

Ch4: A definition of the weights, Boolean, and temporal operators in wSTL+ in the context of partial satisfaction as tie-breaking rules for conflicting subformulae, inclusive (soft), and exclusive (hard) preference constraints.

Ch4: A MILP approach that captures the semantics of wSTL+ specifications as fractions of preferred satisfaction.

Chapter 2

Expressive Mission Specification and Efficient Encoding for Planning.

2.1 Preliminaries

2.1.1 Notation

Let $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{B}$ denote integer, real, and binary numbers sets. The set of integers greater than a is $\mathbb{Z}_{\geq a}$. A vector of ones of size n is denoted by $\mathbf{1}_n$. For a set S, 2^S and $|S|$ represent its power set and cardinality. For $S \subseteq \mathbb{R}$ and $\alpha \in \mathbb{R}$, we have $\alpha + S = \{\alpha + x \mid x \in S\}$. The integer interval (range) from a to b is $[a..b]$. For a range $I = [a..b]$, we use $\underline{I} = a$ and $\bar{I} = b$. Let $x \in \mathbb{R}^d$ be a d-dimensional vector. The i-th component of x is given by x_i, $i \in [1..d]$.

2.1.2 Metric Temporal Logic (MTL)

Metric Temporal Logic (MTL), as introduced in [18], is a specification language expressing real-time properties. The syntax of MTL is

$$\phi ::= \top \mid \neg\phi \mid \pi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \Box_I \phi \mid \Diamond_I \mid \phi_1 \mathcal{U}_I \phi_2,$$

where ϕ, ϕ_1, and ϕ_2 are MTL formulae, $\top$ is the logical *True* value, $\pi \in \Pi$ is an atomic proposition. over the i-th component of signal s, $\neg$, $\vee$, and $\wedge$ are the Boolean negation, disjunction, and conjunction operators, respectively. $\Box_I$, $\Diamond_I$ $\mathcal{U}_I$ are the timed *always*, *eventually*, and *until* operator with $I = [t_1 \,..\, t_2]$ a discrete-time interval, $0 \leq t_1 \leq t_2$. The semantics of MTL formulae ϕ at time t is recursively defined over timed state sequence $w = (s, t)$ where $s = s_0, s_1 \ldots s_p$ is a sequence of atomic prepositions by [81] as

$$
\begin{aligned}
w \vDash \pi &\equiv s_o \vDash \pi, \\
w \vDash \neg\phi &\equiv w \nvDash \phi, \\
w \vDash \phi_1 \wedge \phi_2 &\equiv w \vDash \phi_1 \wedge w \vDash \phi_2, \\
w \vDash \phi_1 \vee \phi_2 &\equiv w \vDash \phi_1 \vee w \vDash \phi_2, \\
w \vDash \Diamond_I \phi &\equiv \exists t \in I, (s, t) \vDash \phi, \\
w \vDash \Box_I \phi &\equiv \forall t \in I, (s, t) \vDash \phi, \\
w \vDash \phi_1 \mathcal{U}_I \phi_2 &\equiv \exists t' \in t + I \text{ s.t. } (s, t') \vDash \phi_2 \wedge \forall t'' \in [t \,..\, t'] \, (s, t'') \vDash \phi_1,
\end{aligned}
\tag{2.1}
$$

where $\vDash$ and $\nvDash$ denote satisfaction and violation, respectively. A timed state sequence $w = (s, t)$ that satisfies ϕ is denoted by $w \vDash \phi$, and it is true if $(s, 0) \vDash \phi$.

2.1.3 Signal Temporal Logic (STL)

This section discusses the definition of Signal Temporal Logic (STL) introduced in [108] and some properties such as the STL formula's qualitative and quantitative semantics, soundness, and time-bound. STL is a tool used to monitor temporal properties of real-valued signals. A *signal* s is a function $s : \mathbb{Z}_{\geq 0} \to \mathbb{R}^n$ that maps each time point $k \in \mathbb{Z}_{\geq 0}$ to an n-dimensional vector of real values $s(k)$. The i-th component of the vector s is s_i.

Definition 2.1.1 (STL syntax)**.** *The syntax of STL in Backus-Naur form over linear predi-*

cates is

$$\phi ::= \top \mid \bot \mid \mu \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \Diamond_I \phi \mid \Box_I \phi \mid \phi_1 \mathcal{U}_I \phi_2 \mid \phi_1 \mathcal{R}_I \phi_2 \tag{2.2}$$

where $\top$ and $\bot$ are the logical *True* and *False*; μ is a linear *predicate* that takes the form $s_i \geq \pi$, where π is the threshold over the i-th component of signal s; $\neg$, $\wedge$, and $\vee$ are the Boolean *negation*, *conjunction*, and *disjunction* operators, respectively. $\Diamond$ (*eventually*), $\Box$ (*always*), $\mathcal{U}$ (*until*), and $\mathcal{R}$ (*release*) are temporal operators with time bound in the range I. $\Diamond_I \phi$ is satisfied if "the formula ϕ is true at least once during the time range I", while $\Box_I \phi$ requires ϕ to be true for the entire duration of I. The satisfaction of $\phi_1 \mathcal{U}_I \phi_2$ requires that "ϕ_1 must hold at least until ϕ_2 becomes true within time range I". Lastly, $\phi_1 \mathcal{R}_I \phi_2$ captures that "ϕ_1 releases ϕ_2, satisfied if ϕ_2 holds within time range I up until and including a time instance in I at which ϕ_1 is true". Note that if ϕ_1 does not become true, ϕ_2 must hold for the entire duration I.

The (qualitative) semantics of STL formula describes whether a signal s satisfies ϕ, and it is defined recursively for every Boolean and temporal operator as follows.

Definition 2.1.2 (Qualitative semantics)**.** *Given an STL formula ϕ and a signal s, the sat-*

isfaction of the formula at time k by signal s is defined as in [108]

$$
\begin{aligned}
(s,k) &\vDash (s_i \geq \mu) \equiv s_i(k) \geq \mu, \\
(s,k) &\vDash \neg\phi \equiv (s,k) \nvDash \phi, \\
(s,k) &\vDash \phi_1 \wedge \phi_2 \equiv \big((s,k) \vDash \phi_1\big) \wedge \big((s,k) \vDash \phi_2\big), \\
(s,k) &\vDash \phi_1 \vee \phi_2 \equiv \big((s,k) \vDash \phi_1\big) \vee \big((s,k) \vDash \phi_2\big), \\
(s,k) &\vDash \Diamond_I \phi \equiv \exists k' \in k + I \textit{ s.t. } (s,k') \vDash \phi, \\
(s,k) &\vDash \Box_I \phi \equiv \forall k' \in k + I \textit{ s.t. } (s,k') \vDash \phi, \\
(s,k) &\vDash \phi_1 \mathcal{U}_I \phi_2 \equiv \exists k' \in k + I \textit{ s.t. } \big((s,k') \vDash \phi_2 \wedge \forall k'' \in [k..k'] \textit{ s.t. } (s,k'') \vDash \phi_1\big), \\
(s,k) &\vDash \phi_1 \mathcal{R}_I \phi_2 \equiv \forall k' \in k + I \textit{ s.t. } \big((s,k') \vDash \phi_2 \vee \exists k'' \in [k..k'] \textit{ s.t. } (s,k'') \vDash \phi_1\big),
\end{aligned}
\tag{2.3}
$$

where $\vDash$ and $\nvDash$ denote satisfaction and violation, respectively. A signal s satisfying ϕ, denoted as $s \vDash \phi$, is true if $(s,0) \vDash \phi$.

In addition to Boolean semantics, the STL also includes quantitative semantics known as *robustness* that indicates the degree of satisfaction or violation of an STL formula ϕ at a given time k.

Definition 2.1.3 (Traditional STL Robustness)**.** *Given a formula ϕ and a bounded signal*

s, the robustness score $\rho(\phi, s, k)$ at time k is recursively defined as [57, 51]:

$$
\begin{aligned}
\rho(\top, s, k) &:= \rho_{\top}, \\
\rho(\mu, s, k) &:= s_i(k) - \pi, \\
\rho(\neg\phi, s, k) &:= -\rho(\phi, s, k), \\
\rho(\phi_1 \wedge \phi_2, s, k) &:= \min\left\{\rho(\phi_1, s, k),\, \rho(\phi_2, s, k)\right\}, \\
\rho(\phi_1 \vee \phi_2, s, k) &:= \max\left\{\rho(\phi_1, s, k),\, \rho(\phi_2, s, k)\right\}, \\
\rho\left(\Diamond_I \phi, s, k\right) &:= \max_{k' \in k+I} \rho(\phi, s, k'), \\
\rho\left(\Box_I \phi, s, k\right) &:= \min_{k' \in k+I} \rho(\phi, s, k'), \\
\rho\left(\phi_1 \mathcal{U}_I \phi_2, s, k\right) &:= \max_{k' \in k+I} \Big\{ \min\{\rho(\phi_2, s, k'),\, \min_{k'' \in I'} \rho(\phi_1, s, k'')\} \Big\}, \\
\rho\left(\phi_1 \mathcal{R}_I \phi_2, s, k\right) &:= \min_{k' \in k+I} \Big\{ \max\{\rho(\phi_2, s, k'),\, \max_{k'' \in I'} \rho(\phi_1, s, k'')\} \Big\},
\end{aligned}
\tag{2.4}
$$

where $\rho_{\top} = \sup_{s,\mu}\{s_i - \mu\}$ is the maximum robustness and $I' = [k..k']$.

Theorem 2.1.1 (Soundness of STL)**.** *The robustness score for an STL formula is sound, meaning that $\rho\left(\phi, s, k\right) > 0$ implies $s \vDash \phi$, capturing that signal s satisfies ϕ at time k. On the other hand, $\rho\left(\phi, s, k\right) < 0$ implies $s \nvDash \phi$, therefore, s violates ϕ at time k.*

We refer to the traditional STL robustness score, denoted by $\rho(\phi, s)$, which quantifies the extent to which a signal s satisfies the formula ϕ at time $k = 0$.

The time horizon of an STL formula determines the minimum time required to verify if a signal s satisfies or violates a given specification ϕ. Recursively, computed as in [50] as follows

$$\|\phi\| = \begin{cases} 0, & \text{if } \phi = s_i \geq \pi, \\ \|\phi_1\|, & \text{if } \phi = \neg\phi_1, \\ \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi \in \{\phi_1 \wedge \phi_2,\ \phi_1 \vee \phi_2\}, \\ \bar{I} + \max\{\|\phi_1\|, \|\phi_2\|\}, & \text{if } \phi \in \{\phi_1 \mathcal{U}_I \phi_2,\ \phi_1 \mathcal{R}_I \phi_2\}, \\ \bar{I} + \|\phi_1\|, & \text{if } \phi \in \{\Diamond_I \phi_1,\ \Box_I \phi_1\}. \end{cases} \tag{2.5}$$

2.1.4 Weighted Signal Temporal Logic (wSTL)

This section describes an extension of STL called Weighted STL (wSTL) that allows specifications to capture user preferences, priorities, and importance associated with Boolean and temporal operators, which syntax and semantics were introduced in [111].

Definition 2.1.4 (Weighted Signal Temporal Logic (wSTL) [111])**.** *The syntax of wSTL*[1] *in [24] which includes the definitions of until and release temporal operators is denoted as follows:*

$$\varphi ::= \top \mid \bot \mid \mu \mid \neg\varphi \mid \bigwedge_{i\in[1..N]}{}^{p}\varphi_i \mid \bigvee_{i\in[1..N]}{}^{p}\varphi_i \mid \Diamond_I^w \varphi \mid \Box_I^w \varphi \mid \varphi_1 \mathcal{U}_I^w \varphi_2 \mid \varphi_1 \mathcal{R}_I^w \varphi_2 \tag{2.6}$$

In this framework, the Boolean, temporal operators, true $\top$, false $\bot$, and predicate μ retain the definitions for STL. Weight functions $p : [1..N] \to \mathbb{R}_{>0}$ are used to assign positive weights to conjunction and disjunction formulae. Here, N represents the number of subformulae under the respective operator. For conjunctions, denoted as $(\wedge^p(\varphi_1, \ldots, \varphi_N))$, these weights indicate the relative significance of concurrent tasks. On the other hand, for disjunctions, denoted as $(\vee^p(\varphi_1, \ldots, \varphi_N))$, they represent priority levels among alternatives. Similarly, positive weight functions $w : I \to \mathbb{R}_{>0}$ are used to express user preferences concerning satisfaction times in the case of the eventually operator. They also underscore the importance of satisfaction times for the always operator. Weight functions associated

[1] From now on, we denote an STL specification as ϕ and a wSTL specification as φ.

with until and release operators articulate preferences regarding the timing of transitions between satisfying one subformula or the other.

The qualitative (Boolean) semantics of a wSTL formula φ is the same as the associated STL formula ϕ without the attached weight functions. Additionally, notice that when a wSTL formula φ has all unit weights, which means $p = 1$ and $w = 1$ for every sub-formula, it does not impose any preference, priorities, or importance over the Boolean or temporal operators. Therefore, STL formulae are wSTL formulae with unit weights. The time horizon of a wSTL formula $\|\varphi\|$ is the same as for STL ϕ and computed using (2.5) since weights do not affect the time duration of temporal or Boolean operators.

Example 2.1.1. *For the first 60 minutes, survey regions A, B, and C with higher importance to regions A and B. Within the next 30 minutes, report the data at either upload center U or preferably at center W as early as possible. Always avoid obstacles. Thus, this can now be expressed as* $\varphi = \wedge^{p}(\varphi_1, \varphi_2, \varphi_3)$ *where* $\varphi_1 = \Diamond^{w_1}_{[0,60]}(\wedge^{p_1}(A,\ B,\ C))$, $\varphi_2 = \Diamond^{w_2}_{[60,90]}(\vee^{p_2}(U,\ W))$, $\varphi_3 = \Box^{w_3}_{[0,90]}\neg$*obstacles.* p_1, p_2, w_1, w_2, w_3 *are weight vectors of appropriate lengths.* $p_1 = [p_{11}, p_{12}, p_{13}]$, *where* $p_{11} > p_{13}$, $p_{12} > p_{13}$ *captures higher importance for* A *and* B. *Similarly,* $p_2 = [p_{21}, p_{22}]$, *where* $p_{21} < p_{22}$ *for expressing the preference for* W. *For the temporal operator in* φ_2, *we have* $w_2 = [w_{21}, \ldots, w_{2n}]$, n *is the number of timesteps considered between* $[60..90]$ *and* $w_{2i} \geq w_{2i+1}$, $\forall i \in [1..(n-1)]$ *indicating the preference for satisfying the subtask* φ_2 *earlier during the interval. Since all three sub-tasks are equally important,* p_1 *is a vector of all ones,* $p_1 = [1, 1, 1]$. *Since there are no temporal preferences for* φ_1 *and* φ_3, w_1 *and* w_2 *are vectors of all ones of appropriate lengths.* □

wSTL has quantitative semantics, which is recursively defined, similar to the STL robustness. However, in this case, the semantics are modulated by the weights assigned to the Boolean and temporal operators, which reflect the user's satisfaction preferences. In simple terms, it is a weighted version of the traditional robustness defined in (2.4).

Definition 2.1.5 (Weighted Traditional Robustness [24]). *Given a wSTL formula φ and a bounded signal s, the weighted robustness score $\tilde{\rho}(\varphi, s, k)$[2] at time k is recursively defined as follows*

$$
\begin{aligned}
\tilde{\rho}(\mu, s, k) &\coloneqq s_i(k) - \pi, \\
\tilde{\rho}\left(\neg\varphi, s, k\right) &\coloneqq -\tilde{\rho}(\varphi, s, k), \\
\tilde{\rho}\left(\bigwedge_i^p \varphi_i, s, k\right) &\coloneqq \min_i \left\{p_i^{\wedge} \cdot \tilde{\rho}(\varphi_i, s, k)\right\}, \\
\tilde{\rho}\left(\bigvee_i^p \varphi_i, s, k\right) &\coloneqq \max_i \left\{p_i^{\vee} \cdot \tilde{\rho}(\varphi_i, s, k)\right\}, \\
\tilde{\rho}\left(\Box_I^w \varphi, s, k\right) &\coloneqq \min_{k' \in k+I} \left\{w^{\Box}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}, \\
\tilde{\rho}\left(\Diamond_I^w \varphi, s, k\right) &\coloneqq \max_{k' \in k+I} \left\{w^{\Diamond}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}, \\
\tilde{\rho}\left(\varphi_1 \mathcal{U}_I^w \varphi_2, s, k\right) &\coloneqq \max_{k' \in k+I} \left\{\min\left\{w^{\mathcal{U}}(k'-k) \cdot \tilde{\rho}(\varphi_2, s, k'), \ \min_{k'' \in I'} \left\{\tilde{\rho}(\varphi_1, s, k'')\right\}\right\}\right\}, \\
\tilde{\rho}\left(\varphi_1 \mathcal{R}_I^w \varphi_2, s, k\right) &\coloneqq \min_{k' \in k+I} \left\{\max\left\{w^{\mathcal{R}}(k'-k) \cdot \tilde{\rho}(\varphi_2, s, k'), \ \max_{k'' \in I'} \left\{\tilde{\rho}(\varphi_1, s, k'')\right\}\right\}\right\},
\end{aligned}
\tag{2.7}
$$

where $I' = [k..k']$, for Boolean operators, we have

$$
p_i^{\wedge}(r_i) = \left(\frac{1}{2} - \bar{p}_i\right)\operatorname{sign}(r_i) + \frac{1}{2}, \qquad p_i^{\vee}(r_i) = 1 - p_i^{\wedge}(r_i),
$$

where $r_i = \tilde{\rho}(\varphi_i, s, k)$ is the weighted robustness of the subformula φ_i, and $\bar{p}_i = \frac{p_i}{\mathbf{1}^T p}$ is its normalized weight. Similarly, for the unary temporal operators, we have

$$
w^{\Box}(r_{k'}) = \left(\frac{1}{2} - \bar{w}(k-k')\right)\operatorname{sign}(r_{k'}) + \frac{1}{2}, \ w^{\Diamond}(r_{k'}) = 1 - w^{\Box}(r_{k'}),
$$

where $r_{k'} = \tilde{\rho}(\varphi, s, k')$, $\bar{w}(\mathtt{t}) = \frac{w(\mathtt{t})}{W}$ and $W = \sum_{\mathtt{t}=0}^{k'-k} w(\mathtt{t})$. For the binary temporal operators

[2]From now on, we refer to the robustness of an STL formula ϕ with signal s at time k as $\rho(\phi, s, k)$ and for the robustness of a wSTL formula φ as $\tilde{\rho}(\varphi, s, k)$.

until and release, we define

$$w^{\mathcal{U}}(r_{k'}) = w^{\diamond}(r_{k'}), \qquad w^{\mathcal{R}}(r_{k'}) = w^{\Box}(r_{k'}),$$

where $r_{k'} = \tilde{\rho}(\varphi_2, s, k')$*. The weights* $p^{\wedge}$*,* $p^{\vee}$*,* $w^{\Box}$*, and* $w^{\diamond}$ *are defined as DeMorgan aggregation functions [111].*

The wSTL weighted robustness is sound iff weights are positive.

Theorem 2.1.2 (wSTL soundness [111])**.** *Let STL formula* ϕ *be the wSTL formula* φ *with all weights equal one then*

$$\begin{aligned} \tilde{\rho}(\varphi, s, k) > 0 &\iff \rho(\phi, s, k) > 0 \to s \vDash \varphi, \\ \tilde{\rho}(\varphi, s, k) < 0 &\iff \rho(\phi, s, k) < 0 \to s \nvDash \varphi. \end{aligned} \tag{2.8}$$

The soundness proof outline is provided in [111].

If all the weights defined in the wSTL formula are positive, the resulting robustness score will be a scaled value of the equivalent STL robustness. This score indicates the reliability and accuracy of the wSTL satisfaction biased with the user's preferences and importance, which is directly related to the magnitude of the weights assigned to the various components of the formula.

Remark 2.1.1. *We consider weights to be positive because having even one of the weights equal to zero creates interpretability issues by masking desired or undesired behaviors. For instance, consider the following wSTL formulae* $\varphi_{conj} = \wedge^p(\mu_1, \mu_2, \mu_3)$ *and* $\varphi_{disj} = \vee^p(\mu_1, \mu_2, \mu_3)$*, with* $p = [0.5, 0, 0.5]$*, over a signal* s*. When computing their robustness* $\tilde{\rho}(\varphi_{conj}, s, 0) = \min\{0.5 \cdot \tilde{\rho}(\mu_1, s, 0),\, 0 \cdot \tilde{\rho}(\mu_2, s, 0),\, 0.5 \cdot \tilde{\rho}(\mu_3, s, 0)\}$ *and* $\tilde{\rho}(\varphi_{disj}, s, 0) = \max\{0.5 \cdot \tilde{\rho}(\mu_1, s, 0),\, 0 \cdot \tilde{\rho}(\mu_2, s, 0),\, 0.5 \cdot \tilde{\rho}(\mu_3, s, 0)\}$*. Now consider the following cases for the sign of the robustness values of subformulae regardless of the signal:*

Table 2.1: Conjunction robustness computation cases.

$\tilde{\rho}$	$0.5 \cdot \tilde{\rho}(\mu_1, s, 0)$	$0 \cdot \tilde{\rho}(\mu_2, s, 0)$	$0.5 \cdot \tilde{\rho}(\mu_3, s, 0)$
0	+	0	+
-	-	0	+
-	+	0	-
-	-	0	-

Table 2.2: Disjunction robustness computation cases.

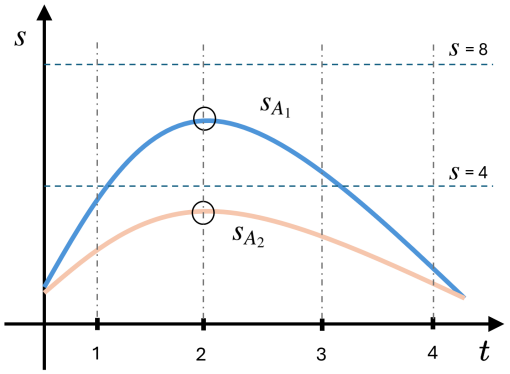
$\tilde{\rho}$	$0.5 \cdot \tilde{\rho}(\mu_1, s, 0)$	$0 \cdot \tilde{\rho}(\mu_2, s, 0)$	$0.5 \cdot \tilde{\rho}(\mu_3, s, 0)$
+	+	0	+
+	-	0	+
+	+	0	-
0	-	0	-

When given weights p, the predicate μ_2 becomes masked. As a result, regardless of whether the predicate is satisfied or violated, its computed robustness is zero. When considering conjunctive operators (e.g., conjunction and always operators), even if all predicates are satisfied, we cannot guarantee overall satisfaction based on soundness properties, since the total robustness is equal to zero. Similarly, for disjunctive operators (e.g., disjunction and eventually operators), if all subformulas violate the formula, we cannot conclude from the overall robustness that the formula has been violated. Thus, in conclusion, zero weights do not lead to the corresponding subformulae being ignored.

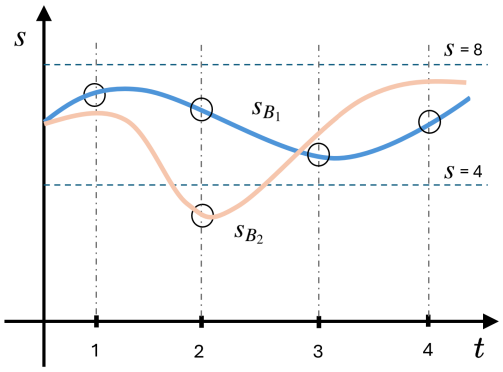
Example 2.1.2 (Weighted Traditional Robustness)**.** *Consider wSTL formula*

$$\varphi_A = \Diamond_{[1,4]}^{w_A}(\wedge^p((s \geq 4), (s \leq 8))),$$

and $\varphi_B = \Box_{[1,4]}^{w_B}(\wedge^p((s \geq 4), (s \leq 8)))$*, where* $w_A = [1, 3, 1, 1]$ *indicating that it is thrice as important to satisfy the formulae at the second time instance as compared to remaining time instances,* $w_B = [1, 1, 1, 1]$ *indicating that satisfaction at all time instances is equally important and* $p = [1, 1]$ *for both formulae, denoting that the concurrent requirements are equally important.*



(a) Signals s_{A_1} and s_{A_2} that satisfy and violate, respectively, the wSTL formula $\varphi_A = \Diamond^{w_A}_{[1,4]}(\wedge^p((s \geq 4),(s \leq 8)))$



(b) Signals s_{B_1} and s_{B_2} that satisfy and violate, respectively, the wSTL formula $\varphi_B = \Box^{w_B}_{[1,4]}(\wedge^p((s \geq 4),(s \leq 8)))$

Figure 2.1.1: Example signals for demonstrating the computation of weighted traditional robustness.

Let us consider the signals in Fig. 2.1.1 (a) and (b) as the satisfying (signals s_{A_1}, s_{B_1}) and violating (signals s_{A_2}, s_{B_2}) instances of φ_A and φ_B. From def. 2.1.5, the robustness of the satisfying signal s_{A_1} at the second time instance is given by $\tilde{\rho}(\varphi_A, s_{A_1}, 2) = \min\{(1-(\frac{1}{2}-\frac{3}{6}+\frac{1}{2}))\times 2, (1-(\frac{1}{2}-\frac{3}{6}+\frac{1}{2}))\times 2\} = \min\{\frac{3}{6}\times 2, \frac{3}{6}\times 2\} = 1$. *The robustness of the violating signal s_{A_2} is* $\tilde{\rho}(\varphi_A, s_{A_2}, 2) = \min\{(1-(\frac{1}{2}-\frac{3}{6})\times(-1)+\frac{1}{2})\times(-1), (1-(\frac{1}{2}-\frac{3}{6})\times(-1)+\frac{1}{2})\times(-1)\} = \min\{\frac{1}{2}\times(-1), \frac{1}{2}\times(-1)\} = -0.5$. *The overall robustness for both signals is given by* $\tilde{\rho}(\varphi_A, s_{A_i}) = \max_{k\in[1..4]}\{\tilde{\rho}(\varphi_A, s_{A_i}, k)\}$ *where* $i \in \{1,2\}$. *Similarly, the robustness of the signal s_{B_1} at time 2 is* $\tilde{\rho}(\varphi_B, s_{B_1}, 2) = \min\{(\frac{1}{2}-\frac{1}{4}+\frac{1}{2})\times 3, (\frac{1}{2}-\frac{1}{4}+\frac{1}{2})\times 1\} = \frac{3}{4}$. *And for signal s_{B_2},* $\tilde{\rho}(\varphi_B, s_{B_2}, 2) = \min\{((\frac{1}{2}-\frac{1}{4})\times(-1)+\frac{1}{2})\times(-1), ((\frac{1}{2}-\frac{1}{4})\times(-1)+\frac{1}{2})\times(-5)\} = -2.5$. *Thus, the overall robustness for both signals with respect to φ_B is given by* $\tilde{\rho}(\varphi_B, s_{B_i}) = \min_{k\in[1..4]}\{\tilde{\rho}(\varphi_B, s_{B_i}, k)\}$ *where* $i \in \{1,2\}$. □

Example 2.1.3. *Consider a fully actuated robot with a scalar state in the interval [-10..10] with state saturation and no control bounds for simplicity. The robot is required to 1) always between 0 to 3 minutes, be at least 5 units away from the origin, especially towards the end of the interval (importance for satisfaction times), and 2) between 4 to 6 minutes, especially towards the beginning of the interval, it should be between 2 to 4 units away from the origin (priority for satisfaction times). Additionally, it is thrice as important to*

satisfy the second requirement (importance for subformulae).

The formula expressed using wSTL is $\varphi = \wedge^{p_1}(\Box^{w_1}_{[0,3]}(s \geq 5)), \Diamond^{w_2}_{[4,6]}(\wedge^{p_2}((s \geq 2), (s \leq 4)))$. *To capture the given preferences, one possible set of weight vectors is* $p_1 = [1, 3]$, $w_1 = [1, 1, 1, 2]$, $p_2 = [0.1, 0.1]$, $w_2 = [0.9, 0.3, 0.2]$. *Note that the weights can be any positive values as long as they capture the relative importance and priorities.*

Let $\varphi_1 = \Box^{w_1}_{[0,3]}(s \geq 5)$ *and* $\varphi_2 = \Diamond^{w_2}_{[4,6]}(\wedge^{p_2}((s \geq 2), (s \leq 4)))$. *The time horizon of the formula* φ *is* $\|\varphi\| = \max(\|\varphi_1\|, \|\varphi_2\|) = \max(3, 6) = 6$. *The resulting trajectory is [10, 10, 10, 10, 3, -10, -10]. It can be observed that the robot starts at 10 at* $t = 0$ *and continues to be there until* $t = 3$ *satisfying* φ_1. *Next, it satisfies* φ_2 *at the beginning of the interval [4..6] considering the priority for satisfaction time. The overall robustness is 0.27, which can be computed as follows:* $\tilde{\rho}(\varphi, s) = min\{1 \cdot \tilde{\rho}(\varphi_1, s), 3 \cdot \tilde{\rho}(\varphi_2, s)\} = min\{min\{10, 10, 10, 20\}, 3 \cdot max\{0.9, 0.1\}\}$. *Note that the solution may not be unique.* □

2.2 STL and wSTL Control Synthesis: a Disjunction-Centric Mixed-Integer Linear Programming Approach

This section introduces an efficient optimization-based control synthesis methodology tailored for Signal Temporal Logic (STL) and its extension, weighted Signal Temporal Logic (wSTL). While STL captures Boolean and temporal operators, wSTL further allows users to express preferences and priorities over concurrent and sequential tasks denoted by weights over logical and temporal operators, along with satisfaction times. The proposed approach utilizes Mixed Integer Linear Programming (MILP) for synthesis with both STL and wSTL formulae. We introduce efficient disjunction-centric encodings for STL and wSTL that capture both qualitative and quantitative semantics. This encoding minimizes the number of variables and constraints necessary to represent STL and wSTL formulae by efficiently handling conjunction operations (e.g., conjunction, always operators) and only introducing variables when disjunction operations are used (e.g., disjunction, eventually). Multiple

case studies are conducted to demonstrate the proposed methodology's operation and computational efficiency for the control synthesis with STL and wSTL specifications. While non-linear dynamics and predicates can be considered using piecewise linear functions, this section focuses on linear predicates and dynamics. We show how cost functions involving potentially conflicting objectives expressed in terms of states, controls, and satisfaction robustness impact the solutions to the control synthesis problem for STL and wSTL. We conduct a sensitivity analysis of weights used in wSTL formulae, offering detailed insights into how these weights modulate solutions for given formulae. Finally, the time performance of the disjunction-centric encodings for both STL and wSTL is compared against state-of-the-art frameworks, comprehensively evaluating their efficiency and practical applicability.

2.2.1 Introduction

Owing to their rich expressivity, temporal logic formalisms have been increasingly useful in formally capturing high-level, complex specifications with logical and temporal modalities. Temporal logics such as Linear Temporal Logic (LTL) [6], Signal Temporal Logic (STL) [108], Metric Temporal Logic (MTL) [86], Time-window Temporal Logic (TWTL) [161], Bounded Linear Temporal Logic (BLTL) [154] succinctly capture the complex requirements, inter alia, sequentiality, liveliness, stability, guarantee, obligation, response. Here, we focus on the problem of formal control synthesis, which refers to synthesizing a control signal for a given specification that enables the system to achieve the desired behavior subject to correctness constraints.

Although LTL and its fragments have been widely used for control synthesis of dynamical systems [15, 163, 77, 141, 98, 137] given their closeness to natural language and rich expressivity, timing constraints cannot be explicitly captured using LTL. On the contrary, STL is defined over real-valued signals, has finite time bounds, and has been widely studied for verification [3], continuous monitoring [46], temporal logic inference [16, 97] as well

as multi-agent control synthesis [150, 30]. Note that STL implicitly assumes equal importance over all sub-parts of the specification. Consider the following example, wherein the importance of concurrent sub-tasks and priorities over disjunctive sub-tasks need to be expressed.

Example 2.2.1. *For the first 60 minutes, survey regions A, B, and C with higher importance to regions A and B. Within the next 30 minutes, report the data at either upload center U or preferably at center W as early as possible. Always avoid obstacles.*

Even though the subtasks "*1) surveying regions* $A,\ B$ *and* C *within 60 minutes; 2) Uploading data at one of the upload centers within next 30 minutes; and 3) avoiding obstacles*" - can be readily captured using STL, the importance for regions $A,\ B$ as well as the preference for upload center W cannot be readily expressed. This has motivated the research towards satisfying STL specifications robustly while accounting for user preferences. The existing methods employ various techniques for representing user preferences as temporal logic formulae [7, 82], regular expressions, or soft constraints [2, 127, 128]. Weighted Signal Temporal Logic (wSTL) [111] is an extension of STL that incorporates importance and priorities over sub-formulae, further enhancing STL's expressivity by incorporating preferences for satisfaction without affecting its qualitative satisfaction.

In this section, we consider the control synthesis problem for STL and wSTL formulae. STL and wSTL offer quantitative semantics referred to as *robustness of satisfaction*, which indicates the margin of satisfaction (or violation) of a specification. Owing to this quantitative semantics, the control synthesis problem can be cast as an optimization problem with the objective of maximizing the robustness of satisfaction. In literature, this optimization problem has been solved using heuristics, mixed-integer linear programming, or gradient-based methods [133, 129, 51, 173, 99]. Among these, the heuristic and gradient-based methods are scalable, *sound* (a solution returned by the solver satisfies the specification) but not *complete* (the solver may not return a solution even if the specification is feasible).

Conversely, MILP-based approaches are sound and complete and offer a globally optimal solution. Although MILPs are, in general, NP-hard, efficient off-the-shelf tools as [66] facilitate time-efficient solving. The time complexity of an MILP increases with a growing number of binary decision variables in the formulation. Most existing mixed-integer formulations for STL synthesis introduce a binary decision variable for each predicate at each time-step, thereby limiting the scalability of the approach. To alleviate this, some recent works [91, 174, 165] have introduced efficient encodings for disjunctive operators. These approaches typically employ allocating and bookkeeping binary codes for subformulae under the disjunction. However, a drawback of such approaches is their tendency to focus solely on satisfying one subformula, neglecting the requirement that -*at least* one subformula must be satisfied, and potentially one or more subformulae could be satisfied - a crucial aspect of disjunction semantics. In contrast to these methodologies, here we propose a disjunction-centric encoding that offers a novel solution by efficiently managing conjunctive operations and introducing binary decision variables only for disjunctive operations. The key idea behind this formulation is that considering the nature of conjunctive operations that impose satisfaction of all subformulae under the operator, we can ignore constraints and directly impose their satisfaction on the binary decision variables for disjunctions. Careful bookkeeping of these decision variables avoids introducing unnecessary variables and constraints and, thus, leads to smaller optimization problems and streamlines the solving process.

Here, we consider the control synthesis problem of a linear discrete-time dynamical system for STL and wSTL formulae defined over linear predicates. The main contributions of this section are as follows.

1. We propose an MILP formulation that captures the qualitative and quantitative semantics of wSTL developed in [111]. The proposed MILP encoding captures weighted traditional robustness to quantify satisfaction.

2. We propose an efficient disjunction-centric MILP formulation for STL and wSTL specifications that results in fewer binary decision variables and constraints in the encoding.

3. We extend the definition of wSTL [111] to include *Until* and *Release* operators.

4. We provide the correctness proofs for all proposed encodings.

5. We demonstrate the versatility of wSTL formulae and the functionality of weights to modify the solution of an equivalent STL formula. Additionally, we present case studies for control synthesis using the proposed wSTL MILP formulation and compare it against equivalent STL formulae. We perform sensitivity analysis to characterize the effect of weight modulation on the nature of synthesized trajectories.

6. Finally, we present the time performance comparison between the proposed disjunction-centric MILP formulation and the standard STL encodings.

2.2.2 General overview of the approach

This section outlines our methodology for tackling the control synthesis problem, where we aim to generate signals that satisfy Boolean and temporal constraints within the context of linear dynamical systems. Our approach handles specifications expressed in both Signal Temporal Logic (STL) [109], and its extension, weighted Signal Temporal Logic (wSTL) [111]. STL enables users to articulate Boolean and temporal constraints with explicit time definitions. On the other hand, wSTL extends STL by incorporating user preferences regarding Boolean conditions and temporal aspects through weights. These weights modulate the quantitative semantics of the formula, providing a flexible framework for expressing nuanced constraints. We translate the specifications and the linear dynamical systems into a Mixed Integer Linear Programming (MILP) formulation, thereby establishing an optimization-based framework capable of addressing the control synthesis problem

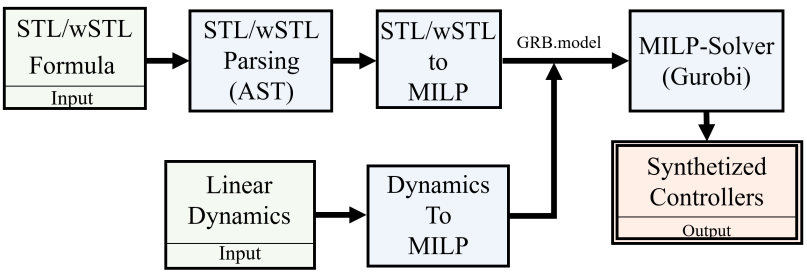


Figure 2.2.1: Schematic overview of the STL and wSTL control synthesis.

while considering both the system dynamics and temporal logic constraints.

Fig. 2.2.1, provides a schematic of our approach to the problem. Initially, the user provides an STL or wSTL specification and a linear dynamical system. The formula undergoes parsing using an Abstract Syntax Tree (AST)[3] where intermediate nodes represent logical and temporal operators, and leaves correspond to predicates [71]. The AST [71] capturing the STL or wSTL formula is constructed using an LL(*) parser [124]. We employ PyTeLo, to recursively translate the AST of the STL or wSTL formula into a MILP, effectively capturing the quantitative semantics of the formula (MILP formulation for STL and wSTL are described in Sec. 2.2.4 and Sec. 2.2.4, respectively). Our disjunction-centric encodings for STL and wSTL reduce the number of variables and constraints required to capture the formula semantics in comparison to previous encodings in [133] and [24], respectively. Subsequently, the linear dynamics are automatically formulated as a MILP, establishing connections between formula variables and dynamic variables. Finally, we employ an MILP solver, such as Gurobi [66], to solve the equivalent MILP, effectively synthesizing controllers that satisfy both the given specification and the dynamics constraints.

2.2.3 Problem Statement

This section introduces the control synthesis problem subject to temporal and logical constraints captured as wSTL formulae. Let us consider a discrete-time dynamical system

[3] A formula can have many equivalent ASTs [71] determined by the parsing methods used.

modeled as

$$s(k+1) = A\, s(k) + B\, u(k), \qquad s(0) = s_\circ, \tag{2.9}$$

where $s(k) \in \mathbf{S} \subseteq \mathbb{R}^n$ represents the state of the system at time step k, while $u(k) \in \mathbf{U} \subseteq \mathbb{R}^m$ denotes the control input at the k-th time instance, with $k \in \mathbb{Z}_{\geq 0}$. The initial state is denoted by $s_\circ \in \mathbf{S}$, and the system's dynamics are characterized by appropriately sized matrices A and B, representing the state transition and input gain matrices, respectively. Consider a trajectory of the system, denoted by $s = s(0)s(1)\ldots$, which is generated by applying a control sequence $u = u(0)u(1)\ldots$ starting from the initial state $s_\circ$. Let $J(s,u)$ represent a cost function associated with generating control signal u while the system is in state s. Some of the more typically formulated cost functions consider sums comprising weighted terms such as 1-norm, ∞-norm, and linear terms.

The problem of control synthesis requires finding an input control sequence $u^* : [0..K-1] \to \mathbf{U}$ that balances minimizing the cost J with maximizing the robustness of φ within the framework of wSTL (STL). This must be achieved while adhering to logical and temporal constraints to satisfy wSTL φ (STL ϕ). The formal definition of the control synthesis problem subject to logical and temporal constraints is as follows.

Problem 2.2.1. *Given a discrete linear dynamical system of the form* (2.9)*, the initial state $s_\circ$, a wSTL φ specification capturing user preferences as in* (2.6) *synthesize a sequence of control inputs u^* such that,*

$$\begin{aligned} u^* = \underset{u}{\operatorname{argmin}}\, & J(s,u) - \lambda\, \tilde{\rho}(\varphi, s) \\ \textit{s.t.} \quad & (2.9) \quad \textit{(linear discrete dynamics)}, \\ & s \vDash \varphi \quad \textit{(mission specification satisfaction)}, \\ & \underline{u} \leq u \leq \bar{u} \quad \textit{(control signal saturation)}, \end{aligned} \tag{2.10}$$

where $\underline{u}$ and $\bar{u}$ are lower and upper control bounds, and $\lambda \geq 0$ is a regularization param-

eter that allows tuning the trade-off between the cost function J and formula satisfaction captured via maximization of the robustness.

The planning time horizon is denoted by K such that $K \geq \|\varphi\|$.

Remark 2.2.1. *In the absence of a defined cost function $J(s, u)$ in* (2.10)*, Pb. 2.2.1 can be reduced to maximizing the robustness $\tilde{\rho}(\varphi, s)$. This approach ensures the satisfaction of φ while considering the user's preferences for satisfaction and the significance of subformulae.*

Note that if the wSTL formula has all weights equal to one, then (2.10) is equivalent to solving the STL control synthesis problem.

2.2.4 Control Synthesis

In this section, we present an optimization problem formulation for Pb. 2.2.1 and introduce a Mixed Integer Linear Program (MILP) for a disjunction-centric encoding for STL and wSTL. Additionally, we compare the disjunction-centric encoding with the wSTL encoding used in our previous work [24].

Assumption 2.2.1. *wSTL specifications are over linear predicates.*

While wSTL formulae have the flexibility to be defined using general, non-linear predicates of the form $l_\mu(s(k)) \geq \pi$, where $l : \mathbb{R}^n \to \mathbb{R}$, we opt to constrain them to more straightforward linear functions, specifically $s_i(k) \geq \pi$ (or $s_i(k) \leq \pi$, as specified below). Despite this restriction, our MILP encoding methodology remains viable. This is achieved by introducing output variables $y_\mu = l_\mu(s(k))$ for all non-linear predicates μ and employing piecewise-linear approximations to represent the output functions l_μ.

STL was initially designed to articulate the desired behavior of systems operating in the continuous-time domain. However, when using MILPs to solve synthesis problems with STL constraints, there is a need to transition to a discrete-time framework. For clarity

and brevity, we consider uniform time discretization. However, the MILP encoding can also be adapted to non-uniform discretization as long as the system dynamics are either inherently linear or can be linearized. However, in both scenarios, the resulting discrete-time control policy may not ensure the satisfaction of the STL or wSTL specification for continuous-time signals as guaranteed. To ensure continuous-time satisfaction, such as ensuring collision avoidance for robots, additional measures may need to be employed, such as employing state constraints like funnels or tubes or imposing robustness lower bound.

Assumption 2.2.2. *wSTL specifications are in positive normal form [133, 57].*

The assumption is not restrictive, as any STL formula, including its weighted extension wSTL, can be converted into a positive normal form, with negation operators only appearing in front of predicates. The transformation process entails eliminating negations completely and defining predicates using comparison operators such as $\geq$ and $\leq$ [133].

Disjunction-centric MILP encoding for STL

In this section, we formulate an MILP capturing a disjunction-centric encoding for STL formulae. This MILP encoding is designed to reduce the number of variables needed to capture the qualitative (2.3) and quantitative (2.4) semantics of an STL formula, compared to the encoding in [133].

Let us define an *augmented-AST*, a data structure we recursively use to create the disjunction-centric encoding for STL, defined as follows.

Definition 2.2.1 (Augmented-AST for STL)**.** *An augmented-AST for STL is represented as a directed tree denoted by $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \xi)$, where $\mathcal{N}$ stands for the set of nodes. Each node is characterized by a tuple $n = (\phi', k)$, where ϕ' represents a subformula, which could be a predicate, Boolean, or temporal operator, and k is the time step at which the node is considered. We associate with each node a binary decision variable $\xi(n) \in \mathbb{B}$ used for the*

MILP encoding. Nodes may be associated with the same decision variable. The edges $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ *depict the relationships between nodes. We define the sets* $\mathrm{pa}(n) = \{n' \mid (n', n) \in \mathcal{E}\}$ *and* $\mathrm{ch}(n) = \{n' \mid (n, n') \in \mathcal{E}\}$ *to represent the parent and children nodes of a given node* n*, respectively. Additionally, a partial order* $\preccurlyeq$ *is established over* $\mathcal{N}$ *such that* $n \preccurlyeq n'$ *if there exists a path from node* n *to node* n'*, indicating* n *being an ancestor of* n'*. The set of ancestors of a node* n *is denoted by* $\mathcal{A}(n) = \{n' \mid n' \preccurlyeq n\}$.

The time step k associated with a node in $\mathcal{N}$ is necessary for temporal operators. For example, the augmented-AST $\phi = \Diamond_{[0,2]} x \geq 2$ will have a root node associated with the eventually operator and three children associated with the predicate $x \geq 2$ at times $k \in [0..2]$.

Note that predicates are leaf nodes n with no children, $\mathrm{ch}(n) = \varnothing$. Similarly, if node n is the root node, then $\mathrm{pa}(n) = \varnothing$.

In Alg. 1, the propagation of MILP variables from root to leaves is computed. Initially, the method traverses the nodes in topological order. If n is the root node ($\mathrm{pa}(n) = \varnothing$), then we create MILP variable $\xi(n) \in \mathbb{B}$ with is defined to capture the satisfaction of the overall formula, lines 4-6. For the disjunction, eventually, until, and release operator, binary variables are created for each child n_{ch} of node n, and stored in $\xi(n_{ch}), \forall n_{ch} \in \mathrm{ch}(n)$, lines 7-9. Conversely, if the node n corresponds to conjunction and always operators, then each child inherits the MILP variable of node n, lines 10-12. Note that for a predicate μ where $\mathrm{pa}(\mu) \neq \varnothing$, we have $\xi(\mu, k) = \xi(\mathrm{pa}(\mu, k))$ which follows from lines 7-9 or lines 10-12 depending on the operation of its parent.

The MILP encoding operates recursively across the augmented-AST nodes corresponding to the STL formula ϕ, starting from the leaves representing the predicates. Let μ denote a *predicate* and k be a time step such that $(\mu, k) \in \mathcal{N}$. The variable $\xi(\mu, k)$ takes a value of one if the predicate μ contributes to the satisfaction of formula ϕ at time $k \in [0..K]$, and zero otherwise. Note that if $\xi(\mu, k)$ takes value one that implies that $\xi(\mathcal{A}(\mu, k)) = \{1\}$, indicating that the MILP variables of all of the ancestors of μ at k must be satisfied. We in-

Algorithm 1 Propagation of MILP variables over augmented-AST for STL.

input: $AST(\phi)$ ▷ AST of the formula ϕ.
output: $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \xi)$ ▷ Augmented-AST.
for $n = (\phi', k) \in \mathcal{N}$ **do** ▷ In topological order.
    if $\mathrm{pa}(n) = \varnothing$ **then**
        $\xi(n) \leftarrow create\_decision\_var(\mathbb{B})$ ▷ Creates binary variable for root node
    end if
    if $\phi' \in \{\vee, \Diamond, \mathcal{U}, \mathcal{R}\}$ **then**
        $\xi(n_{ch}) \leftarrow create\_decision\_var(\mathbb{B}),\ \forall n_{ch} \in \mathrm{ch}(n)$ ▷ Create new binary variables for each child of n.
    end if
    if $\phi' \in \{\wedge, \Box\}$ **then**
        $\xi(n_{ch}) \leftarrow \xi(n),\ \forall n_{ch} \in \mathrm{ch}(n)$ ▷ Inherit variables from parent to children of n.
    end if
end for

troduce a sufficiently large number M (e.g., surpassing the highest upper bound of signals in the STL formula ϕ) and the variable ρ, capturing the robustness of the predicate. The following constraints capture the satisfaction of predicates of the form $s(k) \geq \pi$.

$$\phi = \mu \Rightarrow \begin{cases} s(k) + M(1 - \xi(\mu, k)) \geq \pi + \rho \\ s(k) - M \cdot \xi(\mu, k) \leq \pi + \rho \end{cases}, \tag{2.11}$$

Note that the predicate constraints primarily assess whether the predicate contributes to the overall satisfaction without imposing constraints on the values that variables $s(k)$ and ρ can take. Thus, in case the predicate is not taken into account for satisfaction, the constraints do not make the overall problem infeasible. However, since the objective is to maximize robustness, $s(k)$ is compelled to take the value that optimizes ρ, ensuring it is as far from the threshold value π as feasible. In the scenario involving predicates of the form $s(k) \leq \pi$, we adjust the constraints as follows.

$$\phi = \mu \Rightarrow \begin{cases} s(k) - M(1 - \xi(\mu, k)) \leq \pi - \rho \\ s(k) + M \cdot \xi(\mu, k) \geq \pi - \rho \end{cases}. \tag{2.12}$$

For the *conjunction* operator, there is no need to impose constraints in the model. This is due to the nature of conjunction, which mandates that all children must hold for the operator to be satisfied. Therefore, the conjunction operator inherits its MILP variable to all its children in the augmented-AST, capturing the STL formula.

$$\phi = \bigwedge_i \phi_i \Rightarrow \text{no constraints.} \tag{2.13}$$

In case of the *disjunction* operator, we have $\xi(\phi_i, k) \in \mathbb{B},\ \forall \phi_i \in \mathrm{ch}(\phi)$ as the variable taking value one if subformula ϕ_i is satisfied at time $k \in [0..K]$ and zero otherwise. The following set of constraints captures the "max" nature of the disjunction operator

$$\phi = \bigvee_i \phi_i \Rightarrow \begin{cases} \xi(\phi, k) \leq \sum_i \xi(\phi_i, k) \\ \xi(\phi, k) \geq \xi(\phi_i, k) \end{cases}, \tag{2.14}$$

where $\xi(\phi, k)$ is the MILP variable of ϕ which takes value one if at least one child variable $\xi(\phi_i, k)$ is satisfied.

For the *eventually* operator, the set of constraints follows the logic for the disjunction operator but over time intervals as follows

$$\phi = \Diamond_I \phi_1 \Rightarrow \begin{cases} \xi(\phi, k) \leq \sum_{k'} \xi(\phi_i, k') \\ \xi(\phi_i, k) \geq \xi(\phi_i, k') \end{cases}, \ \forall k' \in k + I, \tag{2.15}$$

where $\xi(\phi, k)$ is the MILP variable of ϕ, which takes value one if at least on variable $\xi(\phi_i, k')$ is satisfied in the interval $k' \in k + I$

The *Always* operator behaves the same as the conjunction operator. There is no need to define constraints. The children inherit the node's variable according to the augmented-

AST, capturing the STL formula.

$$\varphi = \Box_I \phi_1 \Rightarrow \text{no constraints.} \tag{2.16}$$

For the *Until* operator, we use variables $\xi(\phi_1, k'')$ and $\xi(\phi_2, k')$ in which we want $\xi(\phi_1, k'')$ to take value one until $\xi(\phi_2, k')$ gets satisfied at some time step $k' \in k + I$. The following set of constraints captures the until operator

$$\phi = \phi_1 \mathcal{U}_I \phi_2 \Rightarrow \begin{cases} \xi(\phi, k) \leq \sum_{k' \in k+I} \xi(\phi_2, k') & \\ \xi(\phi, k) \geq \xi(\phi_2, k') & \forall k' \in k + I, \\ \xi(\phi_1, k'') \geq \xi(\phi_2, k') & \forall k' \in k + I,\ \forall k'' \in [\underline{I}..k'], \end{cases} \tag{2.17}$$

where the first two constraints require that the MILP variable $\xi(\phi, k)$ takes value one if at some point during the interval I, ϕ_2 is satisfied. The third constraint requires that there exists a time $k' \in k + I$ where ϕ_2 is satisfied and ϕ_1 holds at all times $k'' \in [\underline{I}..k']$.

Lastly, for the *release* operator, we have $\xi(\phi_2, k')$ and $\xi(\phi_1, k'')$. We want $\xi(\phi_2, k')$ to take value one at some time instant $k' \in k + I$, satisfying ϕ_2 such that ϕ_1 is satisfied at all times $k'' \in [k..k' - 1]$. Additionally, if ϕ_2 is never satisfied, then ϕ_1 must hold for the entire time interval $k + I$. The following set of constraints captures the release operator semantics

$$\phi = \phi_1 \mathcal{R}_I \phi_2 \Rightarrow \begin{cases} \xi(\phi, k) \leq \xi(\phi_2, k') + \xi(\phi_1, k'')\ \forall k' \in k + I,\ \forall k'' \in [k..k' - 1], \\ \xi(\phi, k) \geq 1 - |I| + \sum_{k' \in k+I} \xi(\phi_2, k') + \xi(\phi_1, k'')\ \forall k'' \in [k..k' - 1], \\ \xi(\phi, k) \leq \sum_{k' \in k+[0..\bar{I}]} \xi(\phi_1, k') + \sum_{k' \in k+I} \xi(\phi_2, k') \end{cases} \tag{2.18}$$

Finally, the satisfaction of the specification is enforced via the following constraints

$$\xi(\phi, 0) = 1 \text{ and } \rho \geq 0. \tag{2.19}$$

Let us describe the following example that clarifies how the disjunction-centric encoding for an STL specification is captured using previous constraints over the augmented-AST, capturing the semantics of the STL specification in comparison to encoding in [133].

Example 2.2.2. *Consider the following STL formula*

$$\phi = ((s_3 \leq 0) \wedge ((s_1 \geq 0) \wedge (s_2 \geq 0))) \vee ((s_2 > 1) \wedge ((s_1 \geq 3) \vee (s_2 \geq 0))). \tag{2.20}$$

The augmented-AST shown in Fig. 2.2.2a is the equivalent version of the augmented-AST for the STL encoding used in [133]. The set of constraints that captures the semantics of ϕ by using the encoding in [133], are as follows

$$z_0^{\phi} \geq z_0^1;\ z_0^{\phi} \geq z_0^2;\ z_0^{\phi} \leq z_0^1 + z_0^2;\ z_0^1 \leq z_0^{\mu_3};\ z_0^1 \leq z_0^4;\ z_0^1 \geq 1 - 2 + z_0^4 + z_0^{\mu_3};\ z_0^4 \leq z_0^{\mu_7};$$
$$z_0^4 \leq z_0^{\mu_8};\ z_0^4 \geq 1 - 2 + z_0^{\mu_7} + z_0^{\mu_8};\ z_0^2 \leq z_0^{\mu_5};\ z_0^2 \geq z_0^6;\ z_0^2 \geq 1 - 2 + z_0^6 + z_0^{\mu_5};\ z_0^6 \leq z_0^{\mu_9};$$
$$z_0^6 \leq z_0^{\mu_1 0};\ z_0^6 \geq z_0^{\mu_1 0} + z_0^{\mu_9}$$
$$\begin{cases} s_3(k) - M(1 - z_0^{\mu_3}) \leq \pi - \rho \\ s_3(k) + M z_0^{\mu_3} \geq \pi - \rho \end{cases}, \begin{cases} s_1(k) + M(1 - z_0^{\mu_7}) \geq \pi + \rho \\ s_1(k) - M z_0^{\mu_7} \leq \pi + \rho \end{cases},$$
$$\begin{cases} s_2(k) + M(1 - z_0^{\mu_8}) \geq \pi + \rho \\ s_2(k) - M z_0^{\mu_8} \leq \pi + \rho \end{cases}, \begin{cases} s_1(k) + M(1 - z_0^{\mu_9}) \geq \pi + \rho \\ s_1(k) - M z_0^{\mu_9} \leq \pi + \rho \end{cases},$$
$$\begin{cases} s_2(k) + M(1 - z_0^{\mu_1 0}) \geq \pi + \rho \\ s_2(k) - M z_0^{\mu_1 0} \leq \pi + \rho \end{cases},$$

Instead, by using our disjunction-centric encoding, not only the number of variables required are fewer than the previous encoding, going from 11 variables (5 continuous, 6 binary) to 5 binary variables, but also the number of constraints is reduced from 20 to 11. The set of constraints capturing the disjunction-centric encoding described in (2.11)-(2.19)

and the augmented-AST in Fig. 2.2.2b are the following

$$\xi_0^{\phi} \geq \xi_0^1;\ \xi_0^{\phi} \geq \xi_0^2;\ \xi_0^2 \geq \xi_0^3;\ \xi_0^2 \geq \xi_0^4;\ \xi_0^{\phi} \leq \xi_0^1 + \xi_0^2;\ \xi_0^2 \leq \xi_0^3 + \xi_0^4;$$

$$\begin{cases} s_3(k) - M(1-\xi_0^1) \leq \pi - \rho \\ s_3(k) + M\xi_0^1 \geq \pi - \rho \end{cases}, \begin{cases} s_1(k) + M(1-\xi_0^1) \geq \pi + \rho \\ s_1(k) - M\xi_0^1 \leq \pi + \rho \end{cases},$$

$$\begin{cases} s_2(k) + M(1-\xi_0^1) \geq \pi + \rho \\ s_2(k) - M\xi_0^1 \leq \pi + \rho \end{cases}, \begin{cases} s_1(k) + M(1-\xi_0^3) \geq \pi + \rho \\ s_1(k) - M\xi_0^3 \leq \pi + \rho \end{cases},$$

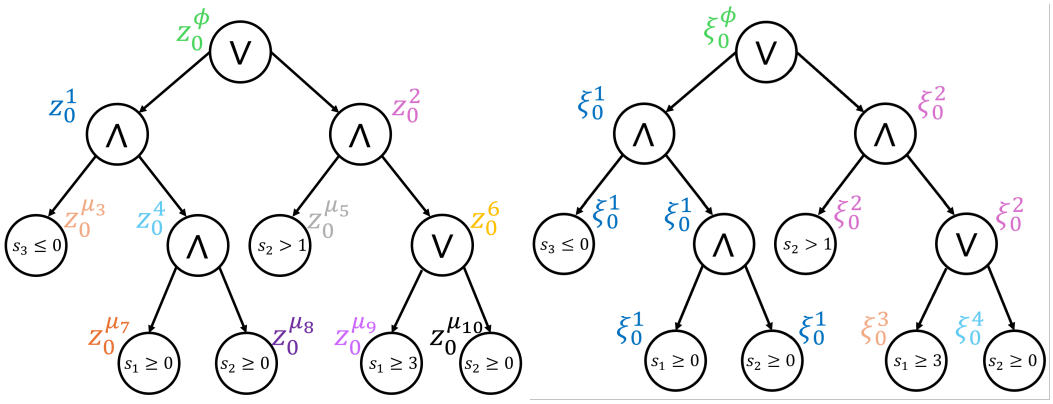
$$\begin{cases} s_3(k) + M(1-\xi_0^4) \geq \pi + \rho \\ s_3(k) - M\xi_0^4 \leq \pi + \rho \end{cases}.$$

It is important to note that the conjunction operator associated with the MILP variable z_0^4 becomes redundant because its two children nodes can collectively form a single conjunction operator with the variable z_0^1. However, the encoding presented in [133] uses independent variables for each conjunction, as the formula treats them as distinct conjunctions. In contrast, our disjunction-centric encoding approach resolves this redundancy by consolidating these into a single variable ξ_0^1, which effectively captures the conjunction of the three predicates μ_3, μ_7, and μ_8.

Proposition 2.2.1 (Disjunction-centric STL encoding Correctness)**.** *Given an STL specification ϕ and signal trajectory s, the constraints* (2.11)-(2.19) *satisfy the following properties:*

1. *$s \vDash \phi$ if constraints are feasible;*
2. *$s \nvDash \phi$ if constraints are infeasible;*
3. *the maximum value of variable ρ^{max} corresponds to the formula robustness $\rho(\phi, s, 0)$.*

Proof. 1. $s \vDash \phi$ if constraints are feasible. Note that we have imposed formula's satisfaction by making $\xi(\phi, 0) = 1$ and $\rho \geq 0$. As the constraints are defined recursively



(a) creation of MILP variables for STL encoding in [133].

(b) creation of MILP variables for disjunction-centric encoding described in Sec. 2.2.4.

Figure 2.2.2: Augmented-AST used to encode (2.20).

over the augmented-AST $\mathcal{G}$ of the STL formula ϕ, we prove that the constraints capture correctly the STL formula using structural induction starting from the base case, a predicate of the form $\mu = s \geq 0$. For *predicates*, we have $\xi(\mu, k) = 1$ then constraints in (2.11) are equivalent to $s(0) \geq \rho$ and $s(0) - M \leq \rho$, since $\pi = 0$. As we impose $\rho \geq 0$, note that the only solution for variable $s(0)$ is to be positive and has a feasible upper bound of $s(0) \leq \rho + M$, capturing satisfaction of $\phi = s \geq 0$. The case of the predicate of the form $\mu = s \leq 0$ follows similarly.

For *disjunction* operator consider a STL formula ϕ, and a subformula $\phi' = \bigvee_i \phi'_i$. Note that $\xi(\phi', k) \in \mathbb{B}$, which modulates whether ϕ' is considered for satisfaction of ϕ. If $\xi(\phi', k) = 0$ the values that the children of ϕ' take do not affect satisfaction of ϕ. In contrast if $\xi(\phi', k) = 1$ then constraints (2.14) corresponds to $1 \leq \sum_i \xi(\phi'_i, k)$ and $1 \geq \xi(\phi'_i, k) \forall \phi'_i \in \mathrm{ch}(\phi', k)$, which effectively capture that at least one child of ϕ' must be satisfied.

For *conjunction* operator, consider an STL formula ϕ, and a subformula $\phi' = \bigwedge_i \phi'_i$. We have two possible cases, for both conjunction operators bypass the variables from parents to children. Thus, $\xi(\mathrm{ch}(\phi', k)) = \{\xi(\phi', k)\}$, therefore if $\xi(\phi', k) = 1$ implies $\xi(\phi'_i, k) = 1$, $\forall (\phi'_i, k) \in \mathrm{ch}(\phi', k)$. Note that all children's nodes under a conjunc-

tion operator are obliged to take the same value since they share the same decision variable, thereby correctly capturing the semantics of conjunction.

The temporal operators follow the same procedure, considering they can be translated into a recursive definition of conjunction and disjunctions.

2. In this case, we can track variables back in the recursion and have some of the variables taking the value of zero or even one, but the problem will indicate there is a violation of $\xi(\phi, 0) = 1$, making the problem infeasible.

3. Let us consider the predicate $\mu = s \geq 0$, since it is the base case on the recursive encoding for general STL specifications. Then, (2.11) is equivalent to $s(0) \geq \rho$ and $s(0) - M \leq \rho$ in case $\xi(\mu, 0) = 1$ such that $\rho > 0$ if $s(0) > 0$. Otherwise, $s(0) - M \geq \rho$ and $s(0) \leq \rho$, in which $\rho < 0$ in case $s(0) < 0$. Note that this set of constraints captures the robustness semantics for predicates.

   In the case, $\mathrm{pa}(\mu_i, k)$ is a disjunction operator such that $\phi = \bigvee_i \mu_i$, we can consider $\xi(\mu_i, k) = (s_i(k) \geq \rho)$, $\forall(\mu_i, k) \in \mathrm{ch}(\phi, k)$, capturing that predicate node takes value one if constraint $(s_i(k) \geq \rho)$ holds, then we can translate to the following equivalent optimization problem

$$\rho^{max} = \mathrm{argmax}\, \rho, \quad \text{s.t. } \exists\ (\mu_i, k) \in \mathrm{ch}(\phi, k),\ (s_i(k) \geq \rho),$$

   where the solution is $\max_{(\mu_i,k)\in \mathrm{ch}(\phi,k)} s_i(k)$ which captures the quantitative semantics of the disjunction operator.

   In the case, $\mathrm{pa}(\mu_i, k)$ is a conjunction such that $\phi' = \bigwedge_i \mu_i$ and $\phi' \subseteq \phi$ with ϕ and STL formula. If $\xi(\phi', k) = 1$, then we can imply that $\xi(\phi', k) = (s_i(k) \geq \rho)$, $\forall(\mu_i, k) \in \mathrm{ch}(\phi', k)$ capturing that the MILP variable of the conjunction node takes value one if constraint $(s_i(k) \geq \rho)$ holds for every child of ϕ'. Then, we can translate to the

following equivalent optimization problem

$$\rho^{max} = \operatorname{argmax} \rho, \quad \text{s.t. } (s_i(k) \geq \rho),\ \forall (\mu_i, k) \in \mathrm{ch}(\phi', k),$$

where the solution is $\min_{(\mu_i,k)\in \mathrm{ch}(\phi',k)} s_i(k)$ which captures the quantitative semantics of the conjunction operator.

□

MILP encodings for wSTL

In this section, we defined the disjunction-centric encoding for wSTL specifications and compared it with our previous encoding defined in [24], which, from now on, we refer to as wSTL-MILP. Pb. 2.2.1 requires satisfying a wSTL specification (2.10) while synthesizing controllers for a linear dynamical system. As per Assump. 2.2.2, wSTL specifications do not include negation operators, making the definition of the weighted traditional robustness (2.7) simpler. In Def. 2.1.5, the gain functions for Boolean operators are represented by constants of the form $p_i^{\wedge} = 1 - \bar{p}_i$ and $p_i^{\vee} = \bar{p}_i$. Similarly, for temporal operators, the gain functions are $w^{\Box}(k') = 1 - \bar{w}(k - k')$ and $w^{\Diamond}(k') = \bar{w}(k - k')$. For ease of notation, we assume that the weights are normalized and adjusted to correspond with the gains mentioned above, such as $p_i = p_i^{\wedge}$.

In a similar way as for STL encoding in Sec. 2.2.4, the MILP is constructed through a recursion over an augmented-AST over the wSTL formula defined as follows

Definition 2.2.2 (Augmented-AST for wSTL)**.** *An augmented-AST for wSTL is represented as a directed tree denoted by $\mathcal{G}^w = (\mathcal{N}^w, \mathcal{E}^w, \mathcal{W}, \xi, \tilde{\eta})$, where $\mathcal{N}^w$ stands for the set of nodes. Each node is characterized by a tuple $n = (\varphi', k)$, where φ' represents a subformula, which could be a predicate, Boolean, or temporal operator and k is the time step at which the node is considered. We associate with each node two decision variables $\xi \in \mathbb{B}$ and $\tilde{\eta} \in \mathbb{R}$ used for the MILP encoding. Nodes may be associated with the same decision variables. The edges*

$\mathcal{E}^w \subseteq \mathcal{N}^w \times \mathcal{N}^w$ capture the relationships between nodes. Lastly, $\mathcal{W}$ is the set of weights coming from the wSTL formula such that $\mathcal{W} : \mathcal{E} \to \mathbb{R}_{>0}$. Definitions of children $\mathrm{ch}(n)$, parent $\mathrm{pa}(n)$, partial order over $\mathcal{G}^w$, and ancestors remains the same as for augmented-AST for STL in Def. 2.2.1.

The weight function $\mathcal{W}(e)$ with $e \in \mathcal{E}^w$ capturing wSTL weights is necessary for every edge from a Boolean or temporal operator to its children. For example, consider $\varphi = \wedge^w(x > 0,\, x < 2)$ with $w = [w_1,\, w_2]$. The weight w_1 of child predicate $\mu_1 = x > 0$ of conjunction φ is captured on the edge $(n_{root}, (\mu_1, 0))$ of the augmented-AST, i.e., $\mathcal{W}(n_{root}, (\mu_1, 0)) = w_1$, where $n_{root} = (\varphi, 0)$.

The algorithm for propagating MILP variables through the augmented-AST from the root to the leaves of a wSTL formula is described in Alg. 2. The method starts by traversing the nodes in topological order, line 3. If node n is the root ($\mathrm{pa}(n) = \varnothing$) then we create MILP variable $\xi(n) \in \mathbb{B}$ which captures the satisfaction of the overall formula and $\tilde{\eta}(n) \in \mathbb{R}$ for storing the modulated robustness score, lines 4-7. For operators such as disjunction, eventually, until, and release, robustness and binary satisfaction variables are created for each child of node n, and stored in variables $\xi(n_{ch}), \forall n_{ch} \in \mathrm{ch}(n)$ and $\tilde{\eta}(n_{ch}), \forall n_{ch} \in \mathrm{ch}(n)$, respectively, lines 8-11. On the other hand, if the node n corresponds to conjunction and always operators, then each child n_{ch} inherits the MILP variables of their parent $\xi(n)$ and $\tilde{\eta}(n)\cdot\mathcal{W}(n, n_{ch})$, lines 12-15. where the robustness of $n_{ch} = (\phi_{ch}, k')$ is scaled by the weight $\mathcal{W}(n, n_{ch})$ of ϕ_{ch}, For a predicate μ where $\mathrm{pa}(\mu) \neq \varnothing$, we have $\xi(\mu, k) = \xi(\mathrm{pa}(\mu, k))$ and $\tilde{\eta}(\mu, k) = \tilde{\eta}(\mathrm{pa}(\mu, k))$ which follows from lines 8-11 or lines 12-15 depending of the operation of the parent.

The MILP encoding for wSTL operates similarly as for the STL in Sec. 2.2.4, starting from the leaves representing the predicates. For the wSTL-MILP encoding, we introduce variables $z_k^\mu \in \mathbb{B}$, which assume a value of one if the predicate $\mu \in \mathbb{R}$ contributes to the satisfaction of the formula φ at time $k \in [0..k]$, and zero otherwise. Additionally, we define $\tilde{o}_k^\mu \in \mathbb{R}$ as the robustness score associated with predicate μ at time $k \in [0..K]$, where

Algorithm 2 Propagation of MILP variables over augmented-AST for wSTL.

input: $AST(\varphi)$ ▷ AST of the formula φ.
output: $\mathcal{G}^w = (\mathcal{N}^w, \mathcal{E}^w, \mathcal{W}, \xi, \tilde{\eta})$ ▷ Augmented-AST.
for $n \in \mathcal{N}$ **do** ▷ In topological order.
    if $\mathrm{pa}(n) = \varnothing$ **then**
        $\xi(n) \leftarrow create\_decision\_var(\mathbb{B})$
        $\tilde{\eta}(n) \leftarrow create\_decision\_var(\mathbb{R})$ ▷ Creates variables for root node
    end if
    if $\varphi' \in \{\vee, \Diamond, \mathcal{U}, \mathcal{R}\}$ **then**
        $\xi(n_{ch}) \leftarrow create\_decision\_var(\mathbb{B}),\ \forall n_{ch} \in \mathrm{ch}(n)$ ▷ Create new MILP binary variables for each child of n.
        $\tilde{\eta}(n_{ch}) \leftarrow create\_decision\_var(\mathbb{R}),\ \forall n_{ch} \in \mathrm{ch}(n)$ ▷ Create new $\tilde{\eta}$ variables for each child of n.
    end if
    if $\varphi' \in \{\wedge, \Box\}$ **then**
        $\xi(n_{ch}) \leftarrow \xi(n),\ \forall n_{ch} \in \mathrm{ch}(n)$ ▷ Inherit variables from parent to children of n.
        $\tilde{\eta}(n_{ch}) \leftarrow \tilde{\eta}(n) \cdot \mathcal{W}(n, n_{ch}),\ \forall n_{ch} \in \mathrm{ch}(n)$ ▷ Inherit modulated $\tilde{\eta}$ from parent to children of n.
    end if
end for

$K \geq \|\varphi\|$. Here, M represents a sufficiently large number (e.g., exceeding the largest upper bound of signals utilized in the wSTL formula φ). In the case of the disjunction-centric encoding, we use the predicate variable $\xi(\mu, k)$ and $\tilde{\eta}(\mu, k)$. Constraints for *predicates* of the form $\mu = s(k) \geq 0$ are as follows

$$\varphi = \mu \Rightarrow \underbrace{\begin{cases} s(k) + M(1 - z_k^\mu) \geq \pi + \tilde{\varrho}_k^\mu \\ s(k) - M z_k^\mu \leq \pi + \tilde{\varrho}_k^\mu \end{cases}}_{wSTL\text{-}MILP}, | \underbrace{\begin{cases} s(k) + M(1 - \xi(\mu, k)) \geq \pi + \tilde{\eta}(\mu, k) \\ s(k) - M\xi(\mu, k) \leq \pi + \tilde{\eta}(\mu, k) \end{cases}}_{Disjunction\text{-}centric}. \tag{2.21}$$

Again, the predicate constraints primarily assess whether the predicate contributes to the overall satisfaction without imposing constraints on the values that variables $s(k)$ and the robustness can take. Note that $\xi(\mu, k) = 1$ implies $\xi(\mathcal{A}(\mu, k)) = \{1\}$ and $\tilde{\varrho}(\mu, k) > 0$ implies $\tilde{\varrho}(\mathcal{A}(\mu, k)) \subset \mathbb{R}_{>0}$.

For predicates of the form $\mu = s(k) \leq 0$, the set of constraints uses the same variables as before with the following set of constraints

$$\varphi = \mu \Rightarrow \underbrace{\begin{cases} s(k) - M(1 - z_k^\mu) \leq \pi - \tilde{\varrho}_k^\mu \\ s(k) + M z_k^\mu \geq \pi - \tilde{\varrho}_k^\mu \end{cases}}_{wSTL\text{-}MILP} , | \underbrace{\begin{cases} s(k) - M(1 - \xi(\mu, k)) \leq \pi - \tilde{\eta}(\mu, k) \\ s(k) + M\xi(\mu, k) \geq \pi - \tilde{\eta}(\mu, k) \end{cases}}_{Disjunction\text{-}centric} . \tag{2.22}$$

For the *conjunction* operator, for the disjunction-centric encoding similar to the disjunction centric encoding for STL, there are no constraints needed. In the case of the wSTL-MILP, let $z_k^\varphi \in [0, 1]$ define a variable that takes the value of one if all subformulas φ_i hold at time k, zero otherwise. Let $z_k^{\varphi_i} \in [0,\ 1]$, $\tilde{\varrho}_k^{\varphi_i} \in \mathbb{R}$, and $p_i \in \mathbb{R}_{\geq 0}$ be variables defining satisfaction or violation, robustness score, and subformula weights at time $k \in [0..k]$. Therefore, we have the following set of constraints

$$\varphi = \bigwedge_i^p \varphi_i \Rightarrow \underbrace{\begin{cases} \tilde{\varrho}_k^\varphi \leq p_i \cdot \tilde{\varrho}_k^{\varphi_i}, \forall i \\ z_k^\varphi \leq z_k^{\varphi_i}, \forall i \\ z_k^\varphi \geq 1 - N + \sum_i z_k^{\varphi_i} \end{cases}}_{wSTL\text{-}MILP} , \quad | \quad \underbrace{\text{no constraints}}_{Disjunction\text{-}centric} , \tag{2.23}$$

The objective here is to ensure the satisfaction of z_k^φ if all subformulas φ_i are satisfied. The first equation sets an upper bound on the overall robustness score, ensuring it is not higher than the minimum weighted robustness of the component subformulas $\tilde{\varrho}_k^{\varphi_i}$. The second equation requires the conjunction operator variable z_k^φ to be zero if any of the subformula satisfaction variables $z_k^{\varphi_i}$ are zero. Lastly, z_k^φ takes a value of one only if all subformulas are satisfied.

In handling the *disjunction* operator in wSTL-MILP, we adopt a similar variable type as

that used for conjunction. Additionally, we introduce auxiliary variables $\hat{z}^{\varphi}_{k,i} \in \mathbb{B}$, serving the purpose of linearly formulating the “max" operator. We also introduce $M \in \mathbb{R}_{\geq}0$, representing a suitably large value, as detailed in the predicate descriptions. For the disjunction-centric encoding, we have a similar formulation that uses auxiliary variables $\hat{\xi}^{\varphi}_{k,i} \in \mathbb{B}$. The following constraints remain the same for both the wSTL-MILP and disjunction-centric encoding methodologies.

$$\varphi = \bigvee_i^p \varphi_i \Rightarrow \underbrace{\begin{cases} \tilde{\varrho}^{\varphi}_k \leq p_i \cdot \tilde{\varrho}^{\varphi_i}_k + M \cdot (1 - \hat{z}^{\varphi}_{k,i}), \forall i \\ \hat{z}^{\varphi}_{k,i} \leq z^{\varphi_i}_k, \forall i \\ \sum_i \hat{z}^{\varphi}_{k,i} \geq z^{\varphi}_k \\ z^{\varphi}_k \geq z^{\varphi_i}_k, \forall i \\ z^{\varphi}_k \leq \sum_i z^{\varphi_i}_k \end{cases}}_{wSTL-MILP}, | \tag{2.24}$$

$$\underbrace{\begin{cases} \tilde{\eta}(\varphi, k) \leq p_i \cdot \tilde{\eta}(\varphi_i, k) + M \cdot (1 - \hat{\xi}^{\varphi}_{k,i}), \forall i \\ \hat{\xi}^{\varphi}_{k,i} \leq \xi(\varphi_i, k), \forall i \\ \sum_i \hat{\xi}^{\varphi}_{k,i} \geq \xi(\varphi, k) \\ \xi(\varphi, k) \geq \xi(\varphi_i, k), \forall i \\ \xi(\varphi, k) \leq \sum_i \xi(\varphi_i, k) \end{cases}}_{Disjunction-centric}, \tag{2.25}$$

The initial equation captures the maximum robustness attained among all subformulas φ_i taken into consideration with $z^{\varphi_i}_k = 1$. The variables $\hat{z}^{\varphi}_{k,i}$ function to identify the maximum robustness, while the subsequent equation guarantees the inclusion of the subformula. Furthermore, the third equation imposes the selection of at least one subformula if φ holds. Finally, the last two equations ascertain that z^{φ}_k is assigned a value of one only if at least

one subformula is satisfied.

The behavior of the *always* operator mirrors that of the *conjunction* operator but with weights and variables spanning across time rather than subformulae in the wSTL-MILP approach. Again, for the disjunction-centric encoding, there are no constraints involving this operator. Therefore, we have the following set of constraints

$$\varphi = \square_I^w \varphi_1 \Rightarrow \underbrace{\begin{cases} \tilde{\varrho}_k^{\varphi} \leq w(\mathfrak{t}) \cdot \tilde{\varrho}_{k+\mathfrak{t}}^{\varphi_1}, \forall \mathfrak{t} \in I \\ z_k^{\varphi} \leq z_{k+\mathfrak{t}}^{\varphi_1}, \forall \mathfrak{t} \in I \\ z_k^{\varphi} \geq 1 - |I| + \sum_{\mathfrak{t} \in I} z_{k+\mathfrak{t}}^{\varphi_1}, \end{cases}}_{wSTL-MILP} \quad | \quad \underbrace{\text{no constraints}}_{Disjunction-centric}, \tag{2.26}$$

where $|I|$ denotes the length of the interval and $w(\mathfrak{t}) \in \mathbb{R}_{\geq 0}$ with $\mathfrak{t} \in I$, the temporal weights.

The set of constraints for the *eventually* operator are similar to the ones for disjunction. The following constraints remain the same for both the wSTL-MILP and disjunction-centric encoding methodologies.

$$\varphi = \Diamond_I^w \varphi_1 \Rightarrow \underbrace{\begin{cases} \tilde{\varrho}_k^{\varphi} \leq w(\mathfrak{t}) \cdot \tilde{\varrho}_{k+\mathfrak{t}}^{\varphi_1} + M \cdot (1 - \hat{z}_{k,\mathfrak{t}}^{\varphi}), \forall \mathfrak{t} \in I \\ \hat{z}_{k,\mathfrak{t}}^{\varphi} \leq z_{k+\mathfrak{t}}^{\varphi_1}, \forall \mathfrak{t} \in I \\ \sum_{\mathfrak{t} \in I} \hat{z}_{k,\mathfrak{t}}^{\varphi} \geq z_k^{\varphi} \\ z_k^{\varphi} \geq z_{k+\mathfrak{t}}^{\varphi_1}, \forall \mathfrak{t} \in I \\ z_k^{\varphi} \leq \sum_{\mathfrak{t} \in I} z_{k+\mathfrak{t}}^{\varphi_1} \end{cases}}_{wSTL\text{-}MILP} \quad | \tag{2.27}$$

$$\underbrace{\begin{cases} \tilde{\eta}(\varphi, k) \leq w(\mathfrak{t}) \cdot \tilde{\eta}(\varphi_1, k + \mathfrak{t}) + \\ \qquad\qquad M \cdot (1 - \hat{\xi}_{k,\mathfrak{t}}^{\varphi}), \forall \mathfrak{t} \in I \\ \hat{\xi}_{k,\mathfrak{t}}^{\varphi} \leq \xi(\varphi_1, k + \mathfrak{t}), \forall \mathfrak{t} \in I \\ \sum_{\mathfrak{t} \in I} \hat{\xi}_{k,\mathfrak{t}}^{\varphi} \geq \xi(\varphi, k) \\ \xi(\varphi, k) \geq \xi(\varphi_1, k + \mathfrak{t}), \forall \mathfrak{t} \in I \\ \xi(\varphi, k) \leq \sum_{\mathfrak{t} \in I} \xi(\varphi_1, k + \mathfrak{t}) \end{cases}}_{Disjunction\text{-}centric} . \tag{2.28}$$

Like before, the initial three equations prioritize selecting the maximum robustness of φ_1 across the time interval I when φ is satisfied. Subsequently, the last two equations are in place to ensure that z_k^{φ} assumes a value of one only if φ_1 is satisfied at least once within I. In the case, of the disjunction-centric encoding, it follows a similar logic.

The weights associated with the temporal operator *until* indicate a preference for the time step at which φ_2 transitions to a true value. It is a requirement that φ_1 remains true from time 0 until the start of I. Thus, for $\mathfrak{t} \in [0..\underline{I} - 1]$, the robustness is bounded above by the robustness of φ_1. Conversely, for $\mathfrak{t} \in I$, the constraints imposed on $\tilde{\varrho}_k^{\varphi}$ ensure that the robustness remains upper-bounded by the minimum value between the robustness of

φ_1 and φ_2. Additionally, the constraint governing the auxiliary variable $\hat{z}^{\varphi}_{k,\mathrm{t}}$ guarantees that φ_2 holds at some time step t, while the constraint on z^{φ}_{k} accurately encodes violation. The following constraints remain the same for both the wSTL-MILP and disjunction-centric

encoding methodologies.

$$\varphi = \varphi_1 \mathcal{U}_I \varphi_2 \Rightarrow \underbrace{\begin{cases} \tilde{\varrho}^{\varphi}_{k} \leq \tilde{\varrho}^{\varphi_1}_{k+\mathfrak{t}}, \mathfrak{t} \in [0..\underline{I}-1] \\ \tilde{\varrho}^{\varphi}_{k} \leq w(\mathfrak{t})\tilde{\varrho}^{\varphi_2}_{k+\mathfrak{t}} + M \cdot (1 - \hat{z}^{\varphi}_{k,\mathfrak{t}}), \forall \mathfrak{t} \in I \\ \tilde{\varrho}^{\varphi}_{k} \leq \tilde{\varrho}^{\varphi_1}_{k+\mathfrak{t}} + M \cdot \sum_{\ell=0}^{\mathfrak{t}} \hat{z}^{\varphi}_{k,\ell}, \forall \mathfrak{t} \in I \\ z^{\varphi}_{k} \leq z^{\varphi_1}_{k+\mathfrak{t}}, \mathfrak{t} \in [0..\underline{I}-1] \\ \hat{z}^{\varphi}_{k,\mathfrak{t}} \leq z^{\varphi_2}_{k+\mathfrak{t}}, \forall \mathfrak{t} \in I \\ \sum_{\mathfrak{t} \in I} \hat{z}^{\varphi}_{k,\mathfrak{t}} \geq z^{\varphi}_{k} \\ z^{\varphi}_{k} \leq \sum_{\mathfrak{t} \in I} z^{\varphi_2}_{k+\mathfrak{t}} \\ z^{\varphi}_{k} \leq z^{\varphi_1}_{k+\mathfrak{t}} + \sum_{\ell=0}^{\mathfrak{t}} z^{\varphi_2}_{k+\ell}, \forall \mathfrak{t} \in I \end{cases}}_{wSTL-MILP} \quad | \quad \underbrace{\begin{cases} \tilde{\eta}(\varphi, k) \leq \tilde{\eta}(\varphi_1, k+\mathfrak{t}), \mathfrak{t} \in [0..\underline{I}-1] \\ \tilde{\eta}(\varphi, k) \leq M \cdot (1 - \hat{\xi}^{\varphi}_{k,\mathfrak{t}}) + \\ \qquad\qquad w(\mathfrak{t})\tilde{\eta}(\varphi_2, k+\mathfrak{t}), \forall \mathfrak{t} \in I \\ \tilde{\eta}(\varphi, k) \leq \tilde{\eta}(\varphi_1, k+\mathfrak{t}) + M \cdot \sum_{\ell=0}^{\mathfrak{t}} \hat{\xi}^{\varphi}_{k,\ell}, \forall \mathfrak{t} \in I \\ \xi(\varphi, k) \leq \xi(\varphi_1, k+\mathfrak{t}), \mathfrak{t} \in [0..\underline{I}-1] \\ \hat{\xi}^{\varphi}_{k,\mathfrak{t}} \leq \xi(\varphi_2, k+\mathfrak{t}), \forall \mathfrak{t} \in I \\ \sum_{\mathfrak{t} \in I} \hat{\xi}^{\varphi}_{k,\mathfrak{t}} \geq \xi(\varphi, k) \\ \xi(\varphi, k) \leq \sum_{\mathfrak{t} \in I} \xi(\varphi_2, k+\mathfrak{t}) \\ \xi(\varphi, k) \leq \xi(\varphi_1, k+\mathfrak{t}) + \\ \qquad\qquad \sum_{\ell=0}^{\mathfrak{t}} \xi(\varphi_2, k+\ell), \forall \mathfrak{t} \in I \end{cases}}_{Disjunction-centric} \tag{2.29, 2.30}$$

To simplify the constraint set, we take advantage of the cumulative nature of the until

operator with respect to the fulfillment of φ_1. The last set of equations limits the variables associated with φ only until the moment when φ_2 is satisfied.

For the *release* operator, the weights denote the preference for the time step when φ_2 could be released, corresponding to the period when φ_1 holds. The robustness $\tilde{\varrho}_k^{\varphi}$ is constrained by the robustness of φ_2 multiplied by its weight until φ_1 becomes true (as described in first equation of (2.31)). Additionally, if φ_1 holds before $\overline{I}$, the maximum robustness of φ_1 serves as a bound (as indicated in the second equation of (2.31)). The variables $\hat{z}_{k,\mathfrak{t}}^{\varphi}$ ensure the accurate selection of the maximum robustness among all instances when φ_1 holds (addressed in the third and fourth equations of (2.31)). The following constraints remain the same for both the wSTL-MILP and disjunction-centric encoding methodologies.

$$\varphi = \varphi_1 \mathcal{R}_I \varphi_2 \Rightarrow \underbrace{\begin{cases} \tilde{\varrho}_k^{\varphi} \leq w(\mathfrak{t}) \cdot \tilde{\varrho}_{k+\mathfrak{t}}^{\varphi_2} + M \cdot \sum_{\ell=0}^{k+\mathfrak{t}} \hat{z}_{k,\ell}^{\varphi}, \forall \mathfrak{t} \in I \\ \tilde{\varrho}_k^{\varphi} \leq \tilde{\varrho}_{k+\mathfrak{t}}^{\varphi_1} + M \cdot (1 - \hat{z}_{k,\mathfrak{t}}^{\varphi}), \mathfrak{t} \in [0..\bar{I}] \\ \hat{z}_{k,\mathfrak{t}}^{\varphi} \leq z_{k+\mathfrak{t}}^{\varphi_1}, \forall \mathfrak{t} \in [0..\bar{I}] \\ \sum_{\mathfrak{t}=0}^{\bar{I}} \hat{z}_{k,\mathfrak{t}}^{\varphi} \leq 1 \end{cases}}_{wSTL\text{-}MILP} \quad | \quad \underbrace{\begin{cases} \tilde{\eta}(\varphi, k) \leq w(\mathfrak{t}) \cdot \tilde{\eta}(\varphi_2, k+\mathfrak{t}) + \\ \qquad M \cdot \sum_{\ell=0}^{k+\mathfrak{t}} \hat{\xi}_{k,\ell}^{\varphi}, \forall \mathfrak{t} \in I \\ \tilde{\eta}(\varphi, k) \leq \tilde{\eta}(\varphi_1, k+\mathfrak{t}) + \\ \qquad M \cdot (1 - \hat{\xi}_{k,\mathfrak{t}}^{\varphi}), \mathfrak{t} \in [0..\bar{I}] \\ \hat{\xi}_{k,\mathfrak{t}}^{\varphi} \leq \xi(\varphi_1, k+\mathfrak{t}), \forall \mathfrak{t} \in [0..\bar{I}] \\ \sum_{\mathfrak{t}=0}^{\bar{I}} \hat{\xi}_{k,\mathfrak{t}}^{\varphi} \leq 1 \end{cases}}_{Disjunction\text{-}centric} \tag{2.31, 2.32}$$

Lastly, to ensure the satisfaction of the overall wSTL formula, we impose that the top-

level formula is taken into account and its robustness is positive. For the disjunction-centric encoding, we have

$$\xi(\varphi, 0) = 1 \text{ and } \tilde{\eta}(\varphi, 0) \geq 0. \tag{2.33}$$

In the case of wSTL-MILP the equivalent constraints are $z_0^{\varphi} = 1$ and $\tilde{\varrho}_0^{\varphi} \geq 0$.

The following example clarifies how the disjunction-centric encoding for an STL specification is captured using previous constraints over the augmented-AST, capturing the semantics of the wSTL formula, compared to wSTL-MILP.

Example 2.2.3. *Consider the following wSTL formula*

$$\varphi = \vee^{p_1}(\wedge^{p_2}(s_3 \leq 0,\ s_1 \geq 0),\ s_1 \geq 3), \quad \textit{with } p_1 = [p_{11}, p_{12}],\ \textit{and } p_2 = [p_{21}, p_{22}]. \tag{2.34}$$

The augmented-AST shown in Fig. 2.2.3a is the equivalent version of the augmented-AST for the wSTL encoding used in wSTL-MILP. The wSTL-MILP set of constraints that captures φ are the following

$$\tilde{\varrho}_0^{\varphi} \leq p_{11} \cdot \tilde{\varrho}_0^1 + M \cdot (1 - \hat{z}_{0,1}^{\varphi});\ \hat{z}_{0,1}^{\varphi} \leq z_0^1;\ \tilde{\varrho}_0^{\varphi} \leq p_{12} \cdot \tilde{\varrho}_0^2 + M \cdot (1 - \hat{z}_{0,2}^{\varphi});\ \hat{z}_{0,2}^{\varphi} \leq z_0^2;$$

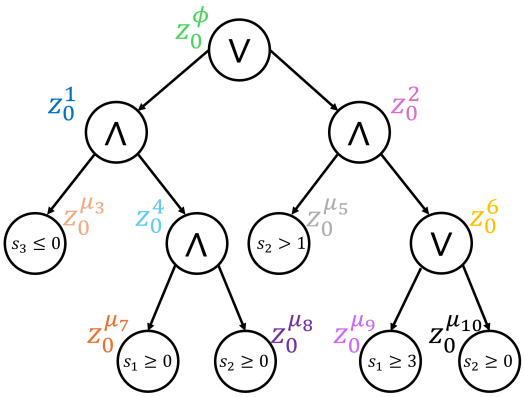
$$\hat{z}_{0,2}^{\varphi} + \hat{z}_{0,2}^{\varphi} \geq z_0^{\varphi};\ z_0^{\varphi} \geq z_0^1;\ z_0^{\varphi} \geq z_0^2;\ z_0^{\varphi} \leq z_0^1 + z_0^2;\ \tilde{\varrho}_0^1 \leq p_{21} \cdot \tilde{\varrho}_0^{\mu_3};$$

$$\tilde{\varrho}_0^1 \leq p_{22} \cdot \tilde{\varrho}_0^{\mu_4};\ z_0^1 \leq z_0^{\mu_3};\ z_0^1 \leq z_0^{\mu 4};\ z_0^1 \geq 1 - N + z_0^{\mu_3} + z_0^{\mu 4}$$

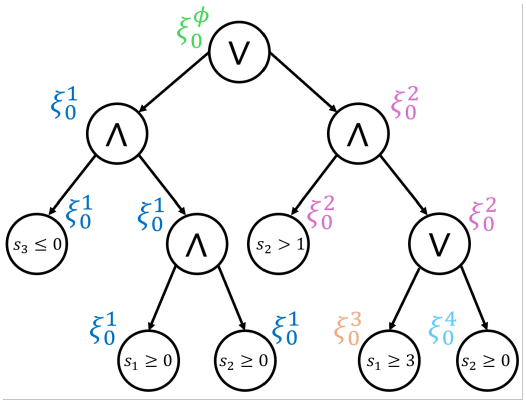
$$\begin{cases} s_1(k) + M(1 - z_0^{\mu_2}) \geq 3 + \tilde{\varrho}_0^{\mu_2} \\ s_1(k) - M z_0^{\mu_2} \leq 3 + \tilde{\varrho}_0^{\mu_2} \end{cases}; \begin{cases} s_3(k) - M(1 - z_0^{\mu_3}) \leq -\tilde{\varrho}_0^{\mu_3} \\ s_3(k) + M z_0^{\mu_3} \geq -\tilde{\varrho}_0^{\mu_3} \end{cases};$$

$$\begin{cases} s_1(k) + M(1 - z_0^{\mu_4}) \geq \tilde{\varrho}_0^{\mu_4} \\ s_1(k) - M z_0^{\mu_4} \leq \tilde{\varrho}_0^{\mu_4} \end{cases},$$

Instead, by using our disjunction-centric encoding, not only is the number of variables required fewer than the previous encoding, going from 10 variables to 6 variables, but



(a) creation of MILP variables for wSTL-MILP encoding..



(b) creation of MILP variables for disjunction-centric encoding described in Sec. 2.2.4.

Figure 2.2.3: Augmented-AST used to encode (2.34).

also the number of constraints is reduced from 16 to 11. The set of constraints capturing the disjunction-centric encoding described in (2.21)-(2.33) *and the augmented-AST in Fig. 2.2.3b are the following*

$$\tilde{\eta}_0^\varphi \leq p_{11} \cdot \tilde{\eta}_0^1 + M \cdot (1 - \hat{\xi}_{0,1}^\varphi);\ \hat{\xi}_{0,1}^\varphi \leq \xi_0^1;\ \tilde{\eta}_0^\varphi \leq p_{12} \cdot \tilde{\eta}_0^2 + M \cdot (1 - \hat{\xi}_{0,2}^\varphi);\ \hat{\xi}_{0,2}^\varphi \leq z_0^2;$$

$$\hat{\xi}_{0,2}^\varphi + \hat{\xi}_{0,2}^\varphi \geq \xi_0^\varphi;\ \xi_0^\varphi \geq \xi_0^1;\ \xi_0^\varphi \geq \xi_0^2;\ \xi_0^\varphi \leq \xi_0^1 + \xi_0^2;$$

$$\begin{cases} s_1(k) + M(1 - \xi_0^{\mu_2}) \geq 3 + \tilde{\eta}_0^{\mu_2} \\ s_1(k) - M\xi_0^{\mu_2} \leq 3 + \tilde{\eta}_0^{\mu_2} \end{cases}; \begin{cases} s_3(k) - M(1 - \xi_0^1) \leq -p_{21} \cdot \tilde{\eta}_0^1 \\ s_3(k) + M\xi_0^1 \geq -\tilde{\eta}_0^1 \end{cases};$$

$$\begin{cases} s_1(k) + M(1 - \xi_0^1) \geq p_{22} \cdot \tilde{\eta}_0^1 \cdot \\ s_1(k) - M\xi_0^1 \leq \tilde{\eta}_0^1 \end{cases}.$$

Proposition 2.2.2 (Correctness)**.** *Given an wSTL specification φ and signal trajectory s, the constraints* (2.21)-(2.33) *satisfy the following properties:*

1. *$s \vDash \varphi$ if constraints are feasible;*
2. *$s \nvDash \varphi$ if constraints are infeasible;*
3. *the maximum values of variables $\tilde{\varrho}_0^\varphi$ and $\tilde{\eta}(\varphi, 0)$ such that the set of constraints is*

feasible are given by the weighted robustness $\tilde{\rho}(\varphi, s, 0)$*, i.e.,* $\tilde{\varrho}_0^{\varphi} = \tilde{\rho}(\varphi, s, 0)$ *and* $\tilde{\eta}(\varphi, 0) = \tilde{\rho}(\varphi, s, 0)$.

Proof. 1. $s \vDash \varphi$ if constraints are feasible. The satisfaction of the specification is imposed by the constraints $\xi(\varphi, 0) = 1$ and $\tilde{\eta}(\varphi, 0) \geq 0$ as defined by (2.21). As the constraints are defined recursively over the augmented-AST $\mathcal{G}^w$ of the wSTL formula φ, we prove that constraints correctly capture the wSTL formula starting from the base case, a predicate of the form $\mu = s \geq 0$. For *predicates*, we have $\xi(\mu, 0) = 1$ then constraints in (2.21) are equivalent to $s(0) \geq \tilde{\eta}(\mu, 0)$ and $s(0) - M \leq \tilde{\eta}(\mu, 0)$, since $\pi = 0$. As we impose $\tilde{\eta}(\varphi, 0) \geq 0$, it follows that $s(0)$ takes positive values and has a feasible upper bound $s(0) \leq M + \tilde{\eta}(\mu, 0)$, thus correctly capturing the satisfaction of $\varphi = s \geq 0$.

For a *disjunction* operator, consider a wSTL formula φ and a subformula $\varphi' = \bigvee_i \varphi'_i$ such that Note that $\xi(\varphi', k) \in \mathbb{B}$, which modulates whether or not φ' is considered for satisfaction of ϕ. If $\xi(\varphi', k) = 0$ the value children nodes of φ' do not affect the satisfaction of φ. In contrast if $\xi(\varphi', k) = 1$ then constraints (2.24) corresponds to $1 \leq \sum_i \xi(\phi'_i, k)$ and $1 \geq \xi(\phi'_i, k)$, $\forall (\phi'_i, k) \in \mathrm{ch}(\phi', k)$, which effectively capture that at least one child of ϕ' must be satisfied.

For *conjunction* operator, consider a wSTL formula φ, and a subformula $\varphi' = \bigwedge_i \varphi'_i$ we have two possible cases, for both conjunction operators bypass the variables from parents to children. Thus, $\xi(\mathrm{ch}(\varphi', k)) = \{\xi(\varphi', k)\}$, therefore if $\xi(\varphi', k) = 1$ implies $\xi(\varphi'_i, k) = 1 \forall (\varphi'_i, k) \in \mathrm{ch}(\varphi', k)$. Note that all children nodes under a conjunction operator are obliged to take the same value since they share the same decision variable thereby correctly capturing the semantics of conjunction.

The temporal operators follow the same procedure, considering they can be translated into a recursive definition of conjunction and disjunctions.

2. In this case, we can track variables back in the recursion and have some of the vari-

ables taking the value of zero or even one, but the problem will indicate there is a violation of $\xi(\varphi, 0) = 1$, making the problem infeasible.

3. Let us consider the predicate $\mu = s \geq 0$, since it is the base case with the help of which general wSTL formulae can be recursively encoded. Then, (2.21) is equivalent to $s(0) \geq \tilde{\eta}(\mu, 0)$ and $s(0) - M \leq \tilde{\eta}(\mu, 0)$ in case $\xi(\mu, 0) = 1$ such that $\tilde{\eta}(\mu, 0) > 0$ if $s(0) > 0$. Otherwise, $s(0) - M \geq \tilde{\eta}(\mu, 0)$ and $s(0) \leq \tilde{\eta}(\mu, 0)$, in which $\tilde{\eta}(\mu, 0) < 0$ in case $s(0) < 0$. Note that this set of constraints captures the robustness semantics for predicates.

   In the case, $pa(\mu_i, k)$ is a disjunction operator such that $\varphi = \bigvee_i \mu_i$, we can consider $\xi(\mu_i, k) = (s_i(k) \geq \tilde{\eta}(\mu_i, k))$, $\forall(\mu_i, k) \in \mathrm{ch}(\varphi, k)$, capturing that predicate node takes value one if constraint $(s_i(k) \geq \tilde{\eta}(\mu_i, k))$ holds in which case $\xi(\mu_i, k)$ and in turn, $\hat{\xi}^{\mu_i}_{k,i}$ are true. Thus we can translate to the following equivalent optimization problem

$$\tilde{\eta}(\varphi, 0) = \mathrm{argmax}\, p_i \cdot \tilde{\eta}(\mu_i, k), \quad \text{s.t. } \exists\, (\mu_i, k) \in \mathrm{ch}(\varphi, k) \text{s.t. } (s_i(k) \geq \tilde{\eta}(\mu_i, k)),$$

   where the solution is $\max_{(\mu_i,k)\in \mathrm{ch}(\varphi,k)} s_i(k)$ which captures the quantitative semantics of the disjunction operator.

   In the case, $pa(\mu_i, k)$ is a conjunction such that $\varphi' = \bigwedge_i \mu_i$ and with φ an wSTL formula. If $\xi(\varphi', k) = 1$, then we can imply that $\xi(\varphi', k) = (s_i(k) \geq \tilde{\eta}(\mu_i, k))$, $\forall(\mu_i, k) \in \mathrm{ch}(\varphi', k)$ capturing that the MILP variable of the conjunction node takes value one if constraint $s_i(k) \geq \tilde{\eta}(\mu_i, k)$ holds for every children of ϕ'. Then, we can translate to the following equivalent optimization problem

$$\tilde{\eta}(\varphi, 0) = \mathrm{argmax}\, p_i \cdot \tilde{\eta}(\mu_i, k), \quad \text{s.t. } \exists\, (\mu_i, k) \in \mathrm{ch}(\varphi, k) \text{s.t. } (s_i(k) \geq \tilde{\eta}(\mu_i, k)),$$

   where the solution is $\min_{(\mu_i,k)\in \mathrm{ch}(\varphi',k)} s_i(k)$ which captures the quantitative semantics

of the conjunction operator.

□

2.2.5 Case Studies

In this section, we examine multiple case studies that showcase the functionality and performance of the framework. First, we have a case study considering control synthesis subject to a cost function and STL and wSTL formulae. Second, we perform a sensitivity analysis of how changes in the weights of wSTL formulae modulate the outcome of control synthesis. Finally, we perform a comprehensive time performance comparison of the encodings under multiple considerations.

All computations for the case studies were conducted on a computer featuring 20 cores at 3.7 GHz, with 64 GB of RAM. To solve the MILP, we utilized Gurobi [66]. For the MILP encoding of STL and wSTL specifications, we used PyTeLo [26] which employs ANTLRv4 [124] for parsing.

Control synthesis

In this section, we present a case study illustrating the control synthesis of STL and wSTL specifications, where the cost function aims to minimize deviation from the origin and control signal usage. For both case studies, we utilize the disjunction-centric encodings for STL and wSTL. This choice is motivated by the absence of discernible solution differences compared to solutions of encodings in [133] and [24], respectively. Let us consider a linear dynamics system described by (2.9), with state variables denoted as $s(k) = [s_x(k), s_y(k)]^\intercal \in \mathbb{R}^2$. The state variables are bounded within $-9 \leq s_x \leq 9$ and $-9 \leq s_y \leq 9$. The control input is represented as $u(k) = [u_x(k), u_y(k)]^\intercal \in \mathbb{R}^2$, with control bounds of $-5 \leq u_x \leq 5$ and $-5 \leq u_y \leq 5$. The linear dynamical system considered in this case study,

along with the initial state variables, are as follows

$$s(k+1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot s(k) + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot u(k) + D, \tag{2.35}$$

$$s(0) = [0,0]^\top, \tag{2.36}$$

For simplicity and clarity in interpreting the results, we set $D = [0,0]^\top$.

The control synthesis for either an STL or wSTL specification for dynamical systems necessitates resolving Pb. 2.2.1. Here, the cost function $J(s,u)$ is formulated to prioritize norm-one objectives for signals and control inputs aimed at minimizing deviation from zero. Consequently, the resulting formulation for a wSTL formula is as follows

$$\begin{aligned} u^* = \underset{u}{\operatorname{argmin}} \quad & \alpha^\top \|s\|_1 + \beta^\top \|u\|_1 - \lambda\, \tilde{\rho}(\varphi, s) \\ \text{s.t.} \quad & (2.35) - (2.36) \quad \text{(linear discrete dynamics)}, \\ & s \vDash \varphi \quad \text{(mission specification satisfaction)}. \end{aligned} \tag{2.37}$$

Note that the problem is identical for an STL specification, with the only difference being the replacement of the wSTL specification φ with the STL specification ϕ. The variable $\tilde{\rho}(\varphi, s)$ represents the robustness of the formulae, where s denotes the dynamics variables coupled to the predicate variables. Additionally, we introduce regularization weights $\lambda \geq 0$, $\alpha \geq 0$, and $\beta \geq 0$. When minimizing the cost function and setting $\lambda = 0$, $\alpha = 0$, and $\beta = 0$, we address the feasibility case, where the primary focus lies in achieving satisfaction with the formula while adhering to the dynamics. On the other hand, if $\lambda > 0$ and $\alpha = \beta = 0$, the objective shifts towards maximizing robustness. However, if $\lambda > 0$, $\alpha > 0$, and $\beta > 0$, the objective becomes three-fold: ensuring the satisfaction of the formula while maximizing robustness, minimizing signal deviation from zero, and conserving control signal usage. The regularization scheme is defined by the disparity between the values of λ, α, and β,

highlighting the user's priorities.

STL control synthesis:

Let us first consider the case of control synthesis with the following STL formula

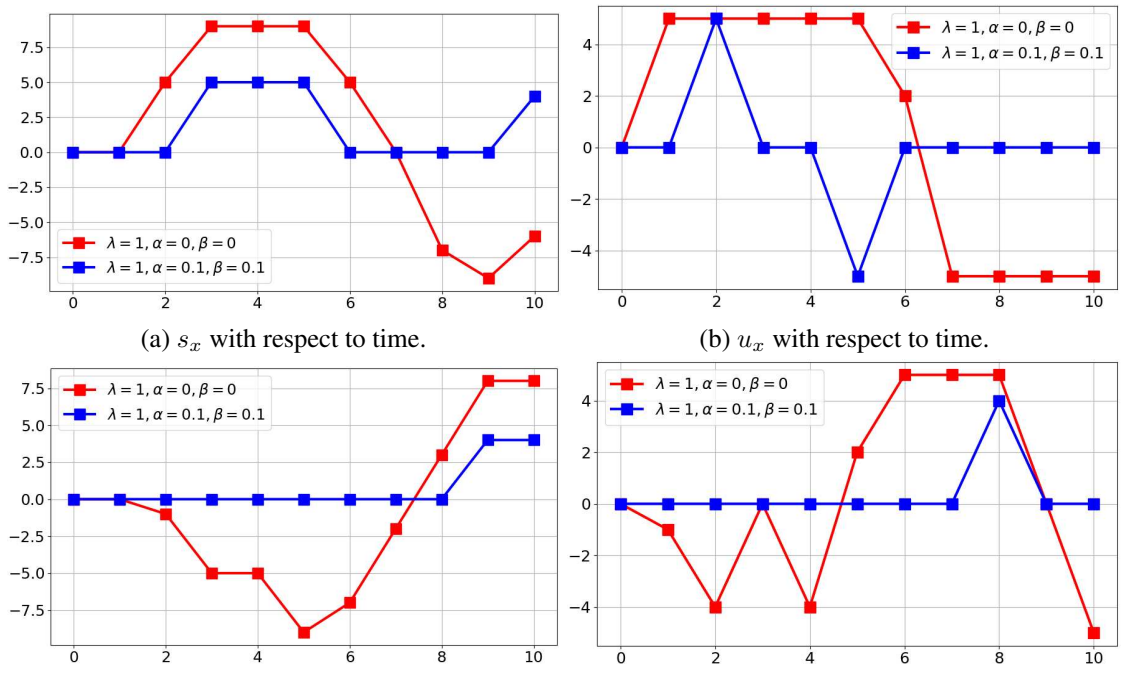
$$\phi = \square_{[3,5]}(s_x >= 3) \wedge ((\square_{[9,10]} s_y >= 2) \vee (\square_{[9,10]} s_y <= -4)).$$

In Fig. 2.2.4, we present two examples demonstrating different regularization parameter settings in (2.37). The first example focuses solely on maximizing robustness, indicated by the red lines with $\lambda = 1$, $\beta = 0$, and $\alpha = 0$. The other example aims to simultaneously maximize robustness, minimize control signal usage, and reduce the deviation of variables s_x and s_y from zero if possible, illustrated by the blue lines with parameters $\lambda = 1$, $\beta = 0.1$, and $\alpha = 0.1$. Notably, when incorporating the minimization of control signal usage for u_x and u_y, along with the reduction of deviation from zero for signals s_x and s_y, the solution and process of control signal generation changes compared to when prioritizing robustness alone. The first subformula $\phi_1 = \square_{[3,5]}(s_x \geq 3)$ is fulfilled by both trajectories. However, considering the cost function, the blue trajectory satisfies this subformula with a narrower satisfaction margin due to its emphasis on minimizing deviation from zero. In the second subformula $\phi_2 = ((\square_{[9,10]} s_y >= 2) \vee (\square_{[9,10]} s_y <= -4))$, a disjunction of two predicates with identical time bounds is posed. The predicate $(s_y \geq 2)$ is chosen because it offers the maximum satisfaction margin. It is important to highlight that despite the differences in the generated trajectories, both solutions successfully fulfill the required specifications.

wSTL control synthesis:

Here, we consider the same parameters as in the previous case study and the optimization problem (2.37) but for the following wSTL formula

$$\varphi = \wedge^{p_1}(\square^{w_1}_{[3,5]} s_x >= 3,\ (\vee^{p_2}(\square^{w2}_{[9,10]} s_y >= 2,\ \square^{w_3}_{[9,10]} s_y <= -4))),$$



(a) s_x with respect to time.

(b) u_x with respect to time.

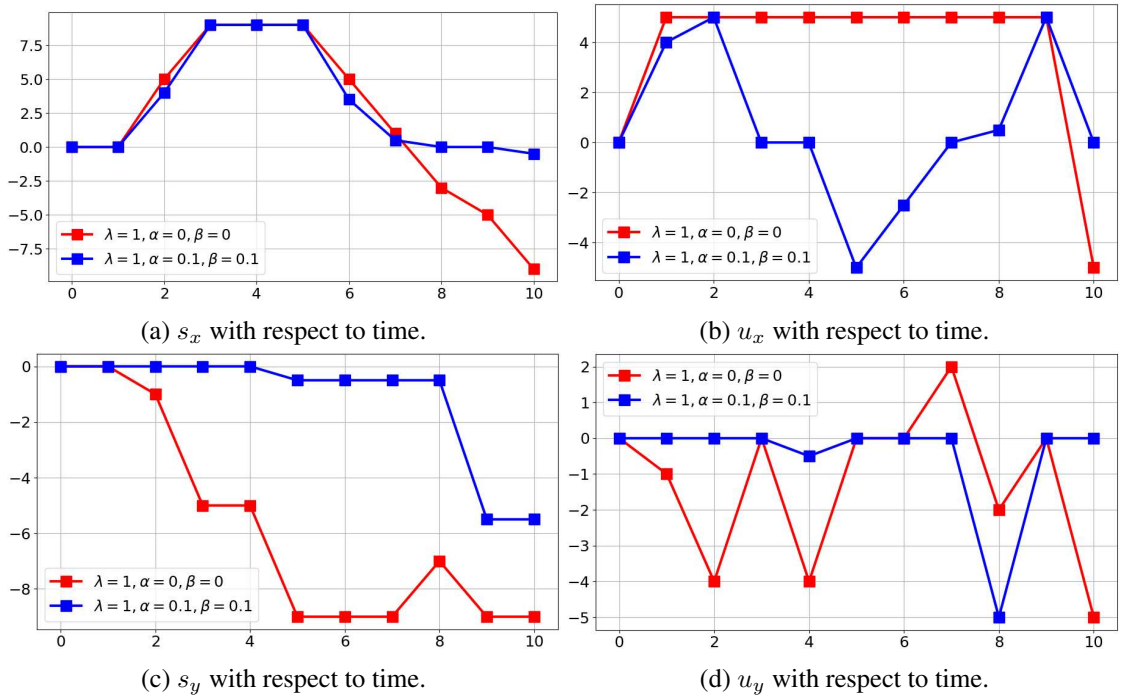
(c) s_y with respect to time.

(d) u_y with respect to time.

Figure 2.2.4: Control synthesis for satisfaction of ϕ coupled with dynamic constraints and control saturation. Red lines maximize only robustness, while blue lines maximize robustness and minimizing control signal generation and signal deviation.

with $p_1 = [1, 1]$, $w_1 = [2, 2, 2]$, $w_2 = w_3 = [2, 2]$, and $p_2 = [0.2, 4]$.

In Fig. 2.2.5, we can see the solution for wSTL control synthesis with the same parameters as the previous case study. Note that for the first subformula $\varphi_1 = \square^{w_1}_{[3,5]} s_x \geq 3$, the trajectory that maximizes robustness while minimizing deviation of signal s_x and usage of control inputs (blue trajectory) closely resembles the trajectory that just maximizes robustness (red trajectory, with $\alpha = \beta = 0$). This similarity arises because the weights prioritize maximizing robustness, given that the scaled robustness is larger than the cost function with relatively small parameters ($\alpha = 0.1$ and $\beta = 0.1$). On the other hand, in the second subformula $\varphi_2 = (\vee^{p_2}(\square^{w2}_{[9,10]} s_y \geq 2, \square^{w_3}_{[9,10]} s_y \leq -4))$, the blue trajectory exhibits a distinct solution compared to the equivalent STL trajectory. This divergence stems from assigning considerably higher priority to the second option in the disjunction. Additionally, notice that the satisfaction margin is not as large as in the other predicate ($\tilde{\rho}(s_y \geq 2, s_y) = (9 - 2) > (9 - 4) = \tilde{\rho}(s_y \leq -4, s_y)$), leading to the trajectory not reaching



(a) s_x with respect to time.

(b) u_x with respect to time.

(c) s_y with respect to time.

(d) u_y with respect to time.

Figure 2.2.5: Control synthesis for satisfaction of φ coupled with dynamics constraints and control saturation. Red lines maximize only robustness, while blue lines maximize robustness while minimizing control signal generation and signal deviation.

its maximum value possible and being influenced by the cost function parameters α and β to stay closer to zero while satisfying the predicate.

Weights sensitivity analysis

In this section, we provide a detailed empirical analysis of the change in the system's behavior as induced by the specified weights. In [24], we demonstrated a simple example wherein modulating the weights over a wSTL formula resulted in different trajectories passing through either side of the obstacle in the environment. We now present a detailed study wherein the weight modulation results in a diverse set of solutions in an environment with more obstacles as compared to [24]. Note that trajectory synthesis is influenced by the formula, the weights over logical and temporal operators as well as the structure of the environment.

Consider the specification of reaching a goal region G between 10 to 20 time units

while avoiding the obstacles $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ expressed as follows

$$\varphi = \wedge^p \left((\Box^{w_1}_{[0,20]} \mathcal{O}_1^c), (\Box^{w_2}_{[0,20]} \mathcal{O}_2^c), (\Box^{w_3}_{[0,20]} \mathcal{O}_3^c), (\Box^{w_G}_{[10,20]} G) \right). \tag{2.38}$$

where $p = [p_1, p_2, p_3, p_G]$ and $p_G = 0.5$. For each obstacle subformula, we vary the associated weight p_j while keeping the other weight fixed at 0.5, $j \in [1..3]$. The set of values for parameter p_j is $P_j = \{p_{j,\ell}\}_{\ell=1}^N$. For this example, we choose $N = 20$ steps, and P_j contains evenly spaced values between 0 and 1.

For a fixed weight on the temporal operators $w_j(k) = 1$, $j \in \{1, 2, 3, G\}$, $k \in [0..20]$, we gradually vary the importance for each subformula while maintaining the same importance over the rest of the subformulae to inspect the effect of weights on the resulting trajectories as shown in Fig. 2.2.6, Fig. 2.2.7, and Fig. 2.2.8. Specifically, let us consider the case of changing the importance for avoiding obstacle $\mathcal{O}_1$ by modulating the importance of $\Box^w_{[0,20]} \mathcal{O}_1^c$ using P_1. For $0 \leq p_1 \leq 0.5$, (2.38) suggests higher importance to the rest of the subformulae. Consequently, the trajectories in Fig. 2.2.6(a), (b) indicate more importance for avoiding $\mathcal{O}_2$ and $\mathcal{O}_3$. However, as the importance for avoiding $\mathcal{O}_1$ increases $p_1 \geq 0.5$, the trajectories move closer to $\mathcal{O}_2$ and $\mathcal{O}_3$ and away from $\mathcal{O}_1$. Similar observations can be made for the rest of the subformulae as shown in Fig. 2.2.7 and Fig. 2.2.8.

To further characterize how the neighboring trajectories are getting closer to or further from each other as the corresponding weights gradually change, we define the maximum deviation between neighboring trajectories as follows:

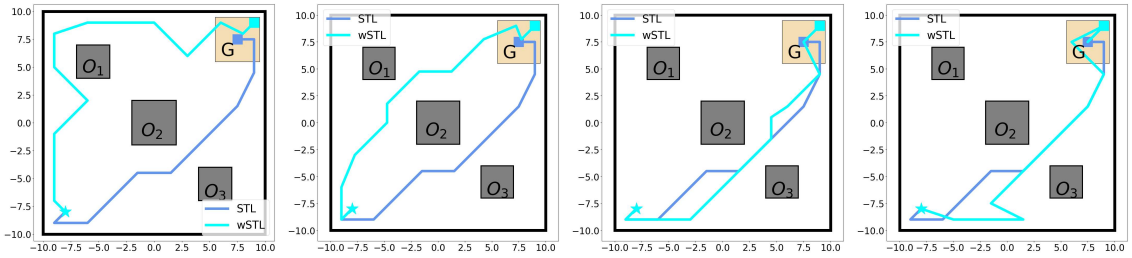
$$\Delta_\oplus = \left\| \frac{s_{\oplus,\ell+1} - s_{\oplus,\ell}}{p_{\ell+1} - p_\ell} \right\|_\infty,$$

where $\oplus \in \{x, y\}$ denotes the x and y-components of signals, $\Delta_\oplus$ denotes the maximum deviation for the respective component of the synthesized signals. For this case, the control

bounds are $-2 \leq u_x \leq 2$ and $-2 \leq u_y \leq 2$. The maximum deviation is crucial as it quantifies the magnitude of deviation along each component of the synthesized signal for an increase in importance.

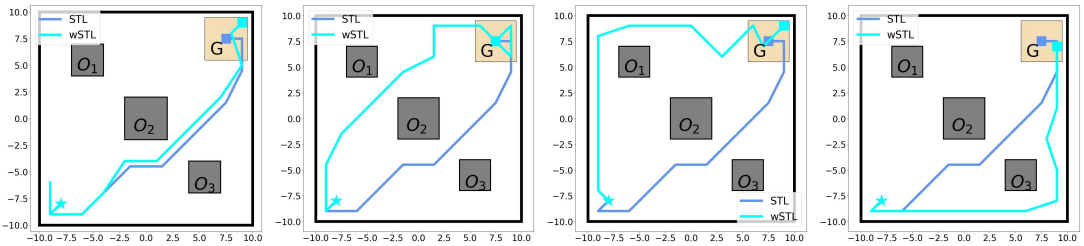
The results, depicted in Fig. 2.2.10, indicate that the maximum deviation along $x-$component of the signal occurs at $p = 0.5$. This is evident from Fig. 2.2.9 since the trajectories for values closer to 0.5 lie on either side of $\mathcal{O}_2$ causing a high deviation in their $x-$coordinates. Similarly, the maximum deviation varies significantly throughout weight modulations. The deviations are higher for weight values wherein the trajectories shift from traversing from one side of the obstacles to the other.

Fig. 2.2.11 depicts the numerical forward difference of the objective with respect to varying weight values $p_\ell \in [0, 1]$. Note that since the computation of the robustness $\tilde{\rho}(\varphi, s)$ involves taking maximum and/or minimum over the robustness values of sub-formulae, the effect of weights on the resulting trajectory is non-trivial.



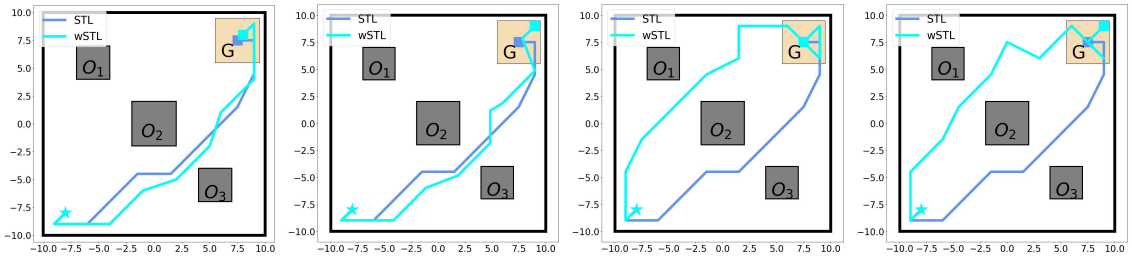
(a) $p_1 = 0$ (b) $p_1 = 0.3$ (c) $p_1 = 0.7$ (d) $p_1 = 0.9$

Figure 2.2.6: Sensibility analysis avoiding region E while varying p_1



(a) $p_2 = 0$ (b) $p_2 = 0.5$ (c) $p_2 = 0.8$ (d) $p_2 = 0.9$

Figure 2.2.7: Sensibility analysis avoiding region E while varying p_2



(a) $p_3 = 0$ (b) $p_3 = 0.2$ (c) $p_3 = 0.5$ (d) $p_3 = 0.9$

Figure 2.2.8: Sensibility analysis avoiding region E while varying p_3

Time performance analysis

This section presents a comprehensive runtime performance comparison of the various STL and wSTL encodings discussed in this study. Firstly, we analyze the time required to solve the specification as the size of the STL or wSTL formula grows. Next, we investigated a fixed formula with the time bounds of the temporal operators increasing. Finally, the time performance when all Boolean and temporal operators have unit weights was compared against cases where weights are randomly assigned.

In all the case studies, we conducted 15 iterations for each scenario and computed the average to derive the time per variable variation across these iterations.

Time performance for growing formula:

Here, we present a runtime performance comparison between STL and wSTL encodings with fixed weights, gradually increasing the size of the mission specification. Consider six variables x, y, z, u, v, and w, each ranging from -9 to 9. The STL formula ϕ and wSTL formula φ for the performance comparison are as follows

$$\phi = \bigwedge_{i=1}^{n} \mathcal{T}_I \left(\left(\mathfrak{s}_1 \otimes_1 \Xi_1 \right) \mathcal{L} \left(\mathfrak{s}_2 \otimes_2 \Xi_2 \right) \right), \tag{2.39}$$

$$\varphi = \bigwedge_{i=1}^{n} \mathcal{T}_I^w \left(\mathcal{L}^p \left(\left(\mathfrak{s}_1 \otimes_1 \Xi_1 \right), \left(\mathfrak{s}_2 \otimes_2 \Xi_2 \right) \right) \right), \tag{2.40}$$

where $\mathcal{T} \in \{\Box, \Diamond\}$, $\mathcal{L} \in \{\wedge, \vee\}$, $\mathfrak{s}_1$ and $\mathfrak{s}_2 \in \{x, y, z, u, v, w\}$, $\otimes \in \{<, \leq, >, \geq\}$, Ξ_1 and $\Xi_2 = \text{rand}(-8, 8)$ are randomly chosen variables, n is an iterator increasing from 1 to 200, and the

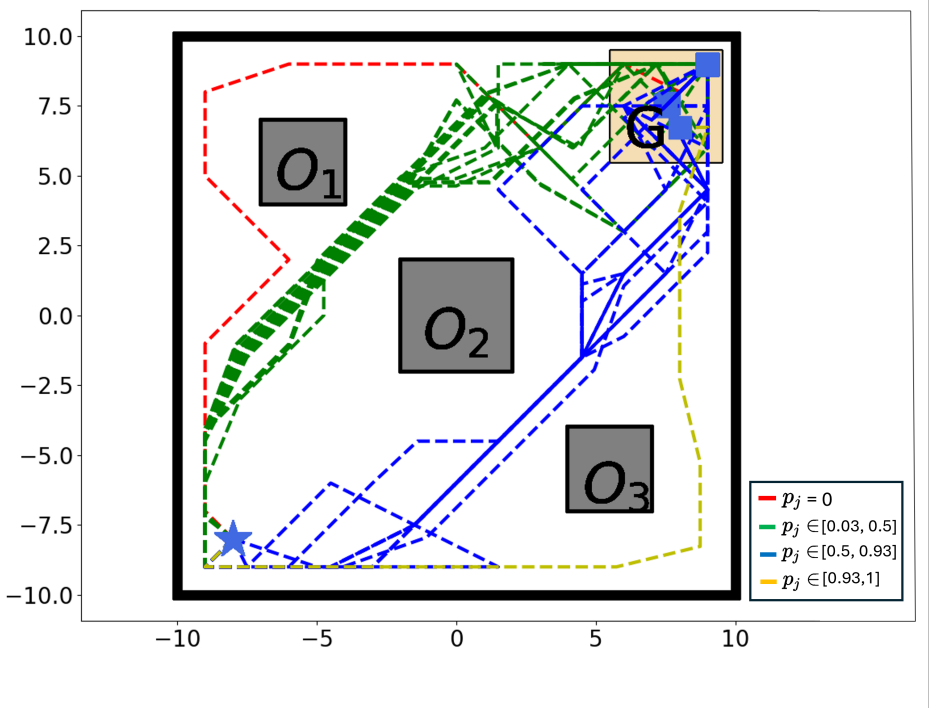
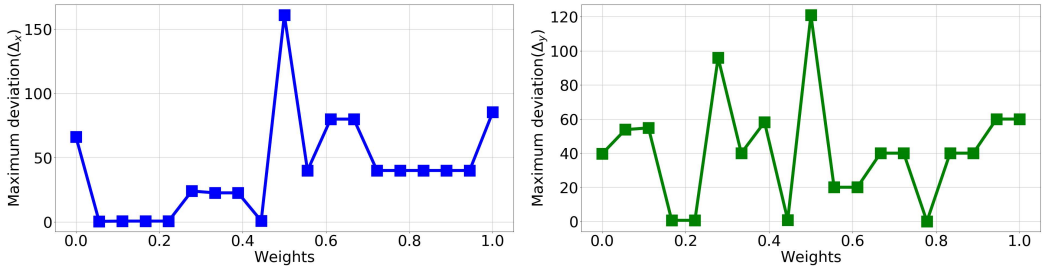


Figure 2.2.9: Changes in weight result in trajectories that are topologically similar or different. The figure shows the ranges of weights for which similar trajectories are obtained.

time interval of the temporal operator is randomly defined as $I = [n{+}4..(n{+}4{+}\text{rand}(1,5))]$. The weights p for Boolean and w for temporal operators are all set to one such that there are no specified priorities or preferences. In Fig. 2.2.12, we compare the runtime performance for increasing STL and wSTL with random weights for the four encodings, disjunction-centric STL and wSTL-MILP, [133], and [24]. The performance of STL grows linearly and is generally faster than wSTL encodings. However, a slight discrepancy between STL and wSTL is expected due to the additional constraints required in wSTL to capture the formula's preferences and importance. Note that the disjunction-centric encodings exhibit individually improved time performance compared to the traditional encodings for both disjunction-centric STL with respect to STL [133] and disjunction-centric wSTL with respect to wSTL-MILP [24].

Time performance for growing time horizon:



(a) Maximum deviation Δ_x with respect to weights.

(b) Maximum deviation Δ_x with respect to weights.

Figure 2.2.10: Maximum deviation Δ_x and Δ_y with respect to weights.

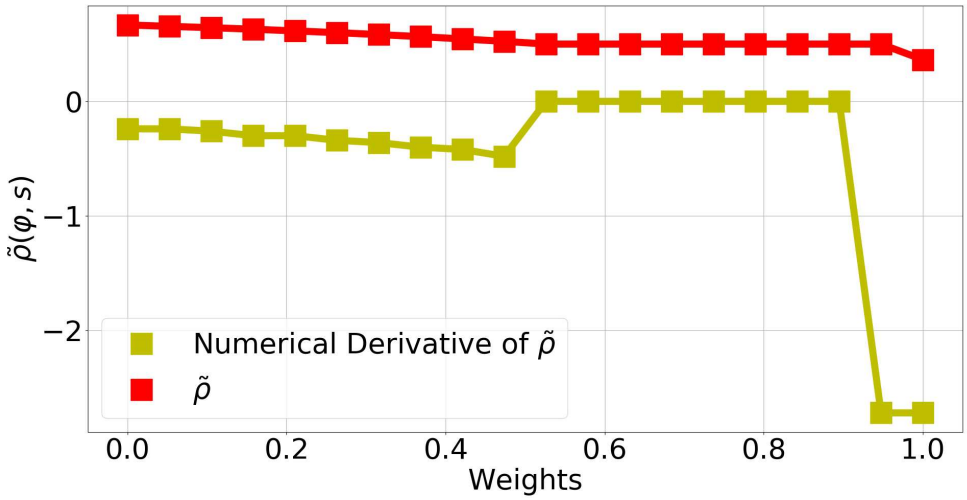


Figure 2.2.11: The objective and its numerical partial derivative with respect to weights.

Here, we present a runtime performance comparison of STL and wSTL encodings for a fixed STL and corresponding wSTL formula with fixed weights. We consider STL and wSTL formulae with only *conjunction* and *always* operators since these are the more sensible for the complexity of disjunction-centric encodings as they have no additional constraints for capturing their semantics. We consider the same variables and bounds as the previous case study. The formulae are outlined as follows

$$\begin{aligned}\phi = \square_{[1,\tau]}\,(x \le 9 \wedge x \ge 3) \wedge \square_{[1,\tau]}(y \le 9 \wedge y \ge 2) \wedge \square_{[1,\tau]}(z \le 9 \wedge z \ge 3) \wedge \\ \square_{[1,\tau]}\,(u \le 9 \wedge u \ge 3) \wedge \square_{[1,\tau]}(v \le 9 \wedge v \ge 3) \wedge \square_{[1,\tau]}(w \le 9 \wedge w \ge 3)\end{aligned} \tag{2.41}$$

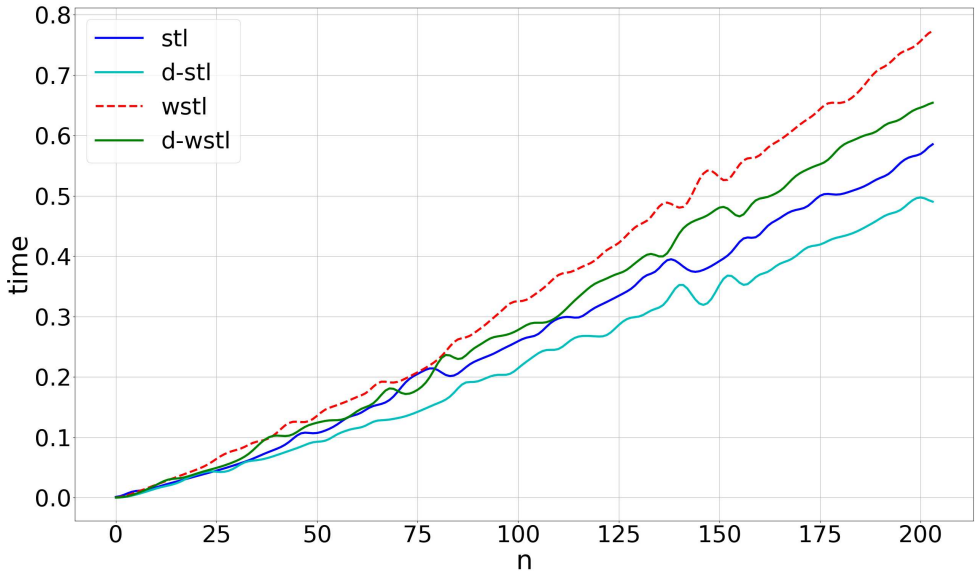


Figure 2.2.12: Time performance of growing random STL (2.39) and wSTL (2.40) formulae. STL label corresponds to encoding in [133], d-stl label corresponds to disjunction-centric encoding in Sec. 2.2.4, wstl corresponds to MILP-wSTL and d-wstl to disjunction-centric wSTL Sec.2.2.4.

$$\begin{aligned}\varphi = \wedge^{w_3}\,&(\square^{w_1}_{[1,\tau]}(\wedge^{w_2}(x \le 9,\, x \ge 3)), \square^{w_1}_{[1,\tau]}(\wedge^{w_2}(y \le 9,\, y \ge 3)),\\ &\square^{w_1}_{[1,\tau]}\,(\wedge^{w_2}(z \le 9,\, z \ge 3)), \square^{w_1}_{[1,\tau]}(\wedge^{w_2}(u \le 9,\, u \ge 3)),\\ &\square^{w_1}_{[1,\tau]}\,(\wedge^{w_2}(v \le 9,\, v \ge 3)), \square^{w_1}_{[1,\tau]}(\wedge^{w_2}(w \le 9,\, w \ge 3)))\end{aligned} \tag{2.42}$$

where $\bar{I} = \tau \in \mathbb{Z}_{\ge 0}$ represents the time limit for all always operators within the formula, and it is set as a growing constant ranging from five to four hundred, spanning across discrete time steps.

In Fig. 2.2.13, the runtime performance of both STL and wSTL encodings, as per equations (2.41) and (2.42) are shown, respectively. As anticipated, the disjunction-centric encodings for both STL and wSTL demonstrate a better performance than their standard counterparts, owing to the significant reduction in the number of variables required to capture the formula. Notably, in the disjunction-centric encodings, only one binary variable is needed for the root, which is then propagated through all conjunctions and always operators until it reaches the predicate nodes. In contrast, the STL encoding [133] and wSTL-MILP encoding [24] necessitate at least one for STL and two for wSTL new MILP variables per

child in every conjunction, always, and predicate operator.

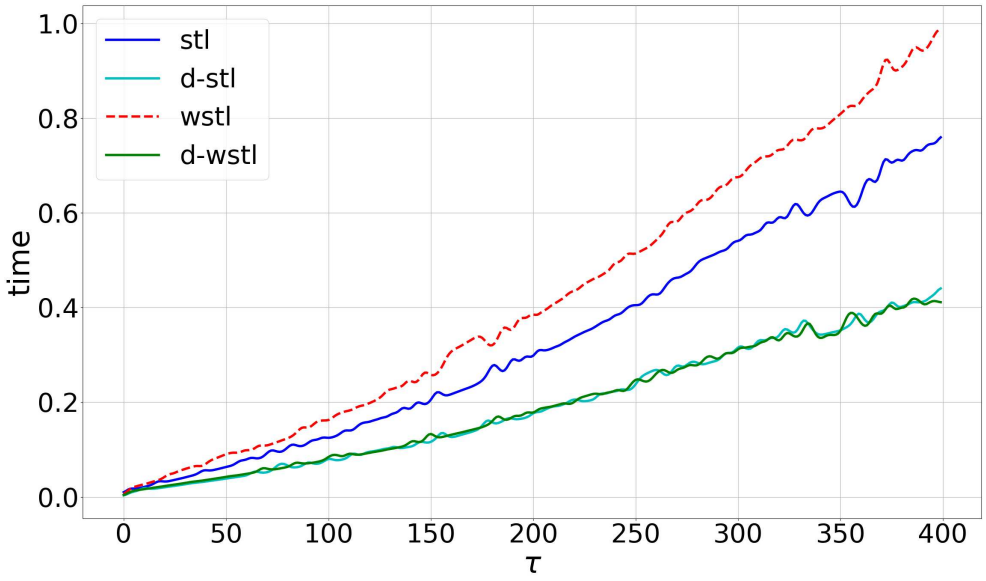


Figure 2.2.13: Time performance of STL (2.41) and wSTL (2.42) formulae with growing time horizon. STL label corresponds to encoding in [133], d-stl label corresponds to disjunction-centric encoding in Sec. 2.2.4, wstl corresponds to wSTL-MILP and d-wstl to disjunction-centric for wSTL Sec.2.2.4.

Time performance for varying weights for wSTL formulae:

Here, we compare the runtime performance between the wSTL-MILP approach and the disjunction-centric encoding for wSTL, employing fixed and random weights for both Boolean and temporal operators. We focus on the wSTL formula outlined in equation (2.40), maintaining consistent variable definitions while introducing random weights for Boolean predicates and temporal operators. Unlike the previous case study, where unit weights were exclusively used, here, we explore a broader range of possibilities. Specifically, we randomly select weights p for Boolean operators and weights w for temporal operators from the interval $(0, 1]$. In Fig. 2.2.14, we analyze how the performance of both encoding strategies wSTL-MILP and disjunction-centric encoding for wSTL are affected by varying weights. Notably, the best performance is observed when employing the disjunction-centric encoding with unit weights assigned to both Boolean and temporal operators. Furthermore, irrespective of whether the weights are held constant or randomly

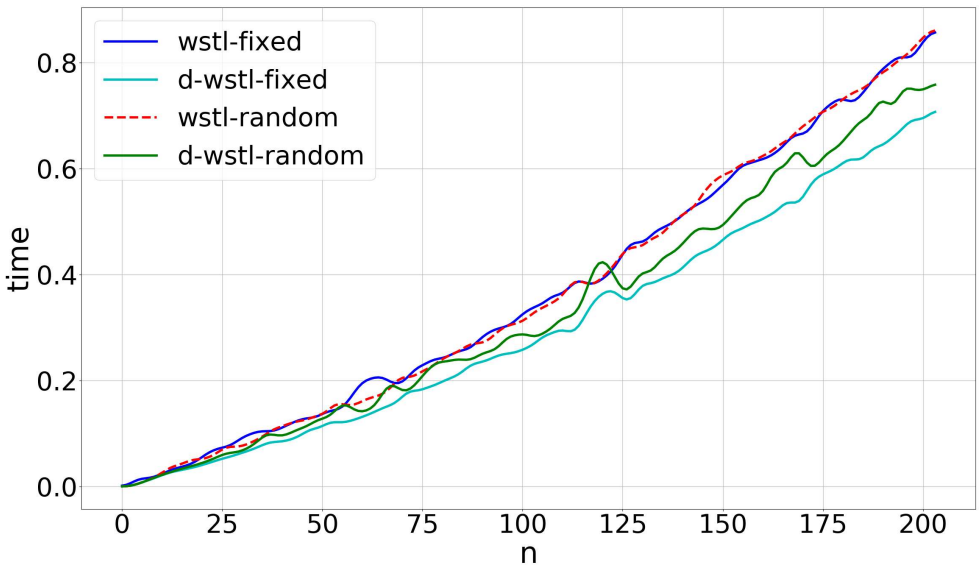


Figure 2.2.14: Time performance of wSTL formulae with growing time horizon. wstl-fixed and d-wstl-fixed labels correspond to wSTL-MILP and disjunction-centric encoding for (2.40) formula with all weights equal one. In contrast, wstl-random and d-wstl-random consider random weights.

assigned, the disjunction-centric encodings consistently outperform the wSTL-MILP encoding. Moreover, as previously demonstrated in [24], there is a minimal disparity between using the unit or random weights for the wSTL-MILP encoding.

Complexity comparison

The performance of a MILP encoding typically hinges on the quantity of binary and integer variables; nonetheless, minimizing the number of continuous variables can also influence performance. In the case of the STL encoding in [133], the number of binary variables $\Gamma^{\mathbb{B}}_{STL}(\phi)$ for a given STL formula ϕ, corresponds to the number of leaves in the augmented-AST bounded by $\Gamma_{pred}(\phi) \leq \Gamma^{\mathbb{B}}_{STL}(\phi) \leq \Gamma_{pred}(\phi) \cdot (\|\phi\| + 1)$, where $\Gamma_{pred}(\phi) = |\{n \mid n \in \mathcal{N} \wedge \mathrm{ch}(n) = \varnothing\}|$ represents the total number of predicates in the specification. For the disjunction-centric STL encoding in Sec. 2.2.4, the number of binary variables $\Gamma^{\mathbb{B}}_{d\text{-}STL}$, scales with the number of disjunctive operators as follows $\Gamma^{\mathbb{B}}_{d\text{-}STL} = \sum_{\phi' \in \Gamma_\vee(\phi)} |\mathrm{ch}(\phi')| + \sum_{\phi' \in \Gamma_\diamond(\phi)} |I| \leq |\phi| + (\|\phi\| + 1) \cdot \Gamma_\diamond(\phi)$, where $I_{\phi'}$ is the time interval associated with ϕ', $\Gamma_\vee(\phi) = |\{n = (\phi', k) \mid n \in \mathcal{N} \wedge \phi' = \vee\}|$ and $\Gamma_\diamond(\phi) = |\{n = (\phi', k) \mid n \in \mathcal{N} \wedge \phi' = \diamond_I\}|$

are the number of disjunctions and eventually operators in the formula. Note that in most cases $\Gamma^{\mathbb{B}}_{d-STL} < \Gamma^{\mathbb{B}}_{STL}$, specially for a formula with conjunctive operators. However, this is not always true. Consider the counter-example $\phi = \Diamond_{[0,1]}\mu_1 \vee \Diamond_{[0,1]}\mu_2$ with $\Gamma^{\mathbb{B}}_{d-STL} = 6$ and $\Gamma^{\mathbb{B}}_{STL} = 4$.

In contrast, the number of continuous variables for encoding an STL formula ϕ with the encoding in [133] is equivalent to the number of nodes in the augmented-AST, $\Gamma^{\mathbb{R}}_{STL} = |\mathcal{N}|$. For the disjunction-centric encoding in Sec. 2.2.4 it is equivalent to the number of predicates $\Gamma^{\mathbb{R}}_{d-STL} = \Gamma_{pred}(\phi)$. Note that in general, $\Gamma^{\mathbb{R}}_{d-STL} < \Gamma^{\mathbb{R}}_{STL}$ except for the trivial case of having a single predicate in which the number of continuous variables is equal. For wSTL-MILP and disjunction-centric wSTL encodings, the complexity comparison is equivalent to the STL and disjunction-centric STL since the additional variables to track the robustness and auxiliary variables are created for both encodings. Therefore, for wSTL, the disjunction-centric approach generally has fewer binary variables and always fewer continuous variables.

2.2.6 Conclusions and Future Work

We introduce an efficient disjunction-centric Mixed Integer Linear Programming (MILP) formulation for control synthesis with Signal Temporal Logic (STL) and its extension, Weighted Signal Temporal Logic (wSTL), considering linear predicates and dynamics. Our approach demonstrates superior efficiency by requiring fewer variables than standard frameworks in the literature for capturing the semantics of STL and wSTL formulae. Additionally, we have proven the correctness of our disjunction-centric encodings. Moreover, we have highlighted the unique capabilities of wSTL formulae in capturing preferences and importance over Boolean and temporal operators, which are critical for scenarios involving prioritized tasks and temporal precedence. By conducting a sensitivity analysis, we have elucidated how changes in the weights assigned to wSTL formulae can alter the outcome of control synthesis, providing valuable insights into solution modification. Finally,

we have comprehensively assessed the MILP encodings' time performance across various scenarios, including growing formulae, increasing time horizons, and considering random and fixed weights for wSTL formulae. Our results demonstrate that the disjunction-centric encoding consistently outperforms standard encodings in the literature, affirming its effectiveness and efficiency in control synthesis applications. In future work, we aim to delve deeper into analytical sensitivity analyses of weights. This exploration will provide a deeper understanding of how subtle modifications in weights can yield complex modulations in solutions. We intend to investigate the applicability of such analyses to switching systems, thereby broadening the scope of our research and uncovering potential applications in diverse domains.

Chapter 3

Modeling and Planning for Complex Dynamics in Multirobot Systems.

Building on the formal synthesis and temporal logic framework established in the previous chapter, Chapter 3 focuses on practically modeling complex multi-robot dynamics in real-world scenarios. This chapter addresses the inherent challenges of deploying multi-robot systems in environments characterized by intricate interactions, diverse capabilities, and dynamic constraints. The chapter begins by exploring swarm control strategies, where groups of robots operate collectively, exhibiting behaviors such as splitting and merging to navigate constrained environments. This section demonstrates how temporal logic can be leveraged to manage emergent swarm behaviors, ensuring coordinated motion despite not controlling individual agent decision-making. Next, we delve into the domain of modular robotics. Here, robots are designed with reconfigurable capabilities that enable them to adapt their structure and function to meet specific task requirements. We present models that capture the nuances of modular reconfiguration, allowing for effective tool manipulation and cooperative task execution. The chapter further expands the discussion to heterogeneous teams of robots tasked with complex operations such as warehouse logistics and resource transportation. These scenarios introduce additional layers of complexity,

including coordination among robots with different capabilities, resource allocation challenges, and energy constraints. We introduce network flow formulations and other modeling techniques that capture these dynamics, demonstrating how high-level temporal logic specifications can be translated into practical, scalable control strategies.

This chapter comprehensively explores the methods used to model the complex dynamics inherent in multi-robot systems. By integrating rigorous formal methods with practical modeling techniques, we bridge the gap between high-level mission planning and the real-world execution of multi-robot operations, paving the way for more adaptive, resilient, and scalable robotic systems.

3.1 Temporal Logic Swarm Control with Splitting and Merging

Here, we present an agent-agnostic framework to control swarms of robots tasked with temporal and logical missions expressed as Metric Temporal Logic (MTL) formulas. We consider agents that can receive global commands from a high-level planner, but no inter-agent communication. Moreover, agents are grouped into sub-swarms whose number can vary over the mission time horizon due to splitting and merging. However, a strict upper bound on the maximum number of sub-swarms is imposed to ensure their safe operation in the environment. We propose a two-phase approach. In the first phase, we compute the trajectories of the sub-swarms, splitting and merging actions using a Mixed Integer Linear Programming approach that ensures the satisfaction of the MTL specification with minimal swarm division over the mission time horizon. Moreover, it enforces the upper bound on the number of sub-swarms. In the second phase, splitting fractions for sub-swarms resulting from splitting actions are computed. A distributed randomized protocol with no interagent communication ensures agent assignments matching the splitting fractions. Finally, we show the operation and performance of the approach in simulations with multiple tasks

that require swarm splitting or merging.

3.1.1 Introduction

In recent years, there has been intense interest in studying robot swarms [53, 1, 107, 152, 121]. Due to their great capacity to handle multiple tasks and offer robustness and resiliency over agent failures. The emerging collective behavior from the interaction of multiple agents gives swarm robotic systems many potential applications ranging from exploration, mapping, and surveillance to search and rescue [23, 52]. Despite these advantages, controlling many agents and commanding their decisions is still challenging in both low-level and high-level control. For complex missions, temporal logics (TL) are a useful specification language for extending swarm behavior to time-varying, task-oriented goals [68, 49, 116].

Several works have considered high-level objectives for swarms given as temporal logic goals [171, 170, 36, 84, 135]. However, these usually consider specifications for swarms as a whole and are not concerned with agent constraints either in the control or communication [35, 115, 169] (e.g., if agents in a swarm are unable to communicate, methods that rely on the dissemination of information will not work). In contrast, we propose a high-level control framework for swarms of aerial robots aware of low-level control and agent communication constraints. Global plans are broadcast to all agents since it is impractical and cumbersome to specify the precise plans for each agent. Plans satisfy complex temporal logic tasks with timing constraints expressed in Metric Temporal Logic (MTL) [18]. Agents are grouped into sub-swarms, and we allow them to split and merge as needed to fulfill mission tasks, e.g., when agents must be at multiple locations simultaneously.

To solve the high-level swarm planning problem, we propose a Mixed Integer Linear Programming (MILP) approach that considers the satisfaction of the MTL specification by finding a motion plan for swarms able to split and merge them as required. For the motion plan in the MILP formulation, we take inspiration from flow equality balance equations as used in [95, 76, 69, 175]. However, swarms may split into multiple sub-swarms to satisfy

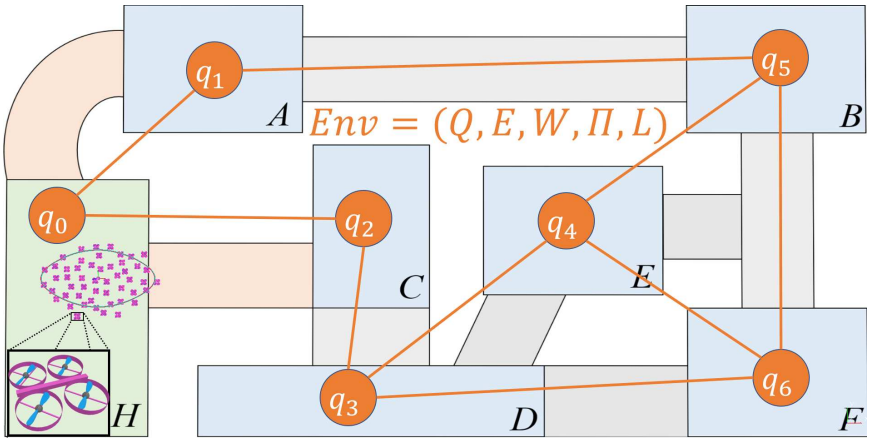


Figure 3.1.1: Example environment. Agents begin in the region H and must visit the labeled blue regions while navigating the narrower gray and orange passages.

simultaneous tasks or merge if splitting is no longer necessary and equality constraints cannot capture these flow dynamics. Instead, we propose inequality constraints that control the flow of swarms while allowing the creation and destruction of swarms, modeling splitting and merging actions.

Here, agents cannot communicate with each other. Thus, only global commands can be used to distribute agents to sub-swarms resulting from splitting and merging actions. Therefore, we propose a distributed randomized method to split large swarms. We build a Directed Acyclic Graph (DAG) from the MILP solution that captures the division of the swarm over time. We use the DAG to compute splitting fractions of sub-swarms that balance their sizes over the mission. Agents use the randomized protocol that samples assignments based on the splitting fractions without inter-agent communication.

The paper's contributions include: 1) We propose an efficient MILP approach to solve the planning problem, considering swarm splitting and merging behaviors for satisfying tasks while constraining the maximum number of simultaneous existing swarms and minimizing unnecessary splitting or traveling. Additionally, standard flow dynamics equations are modified to flow inequalities considering that due to the merging and splitting actions, node and edge equations could not be balanced. 2) We develop a randomized distributed method to split large swarms when communication between agents is impossible. 3) We

develop a method to assign splitting fractions such that the sub-swarms are balanced, given a motion plan from the MILP. 4) We show the performance of the proposed MILP method and the balanced splitting and merging algorithm.

3.1.2 Problem Formulation

In this section, we formulate the planning problem for swarms of aerial robots tasked with rich temporal logic specifications to visit places of interest in an environment. The initial swarm can be split into sub-swarms during the mission horizon to satisfy tasks that require multiple swarms at different locations in the specification. Also, multiple sub-swarms can merge to form a single swarm if splitting is no longer required. Thus, the number of sub-swarms active in the environment may be time-varying. We assume there is an upper bound in the number of swarms we can have simultaneously in the mission. The swarm is not infinitely divisible, and the Unmanned Aerial Vehicles (UAVs) can only fly at a limited number of altitudes for safety reasons. Moreover, splitting the swarm as few times as possible is desirable and just when the mission specification requires it. Here, we introduce the models for the environment, agents, swarms, splitting and merging behaviors, and tasks that define the planning problem for the swarm of robots. First, we define the *environment* capturing the motion of swarms between locations of interest.

Definition 3.1.1 (Environment)**.** *The environment is abstracted as a labeled transition system,* $Env = (\mathcal{Q}, \mathcal{E}, \mathcal{W}, \Pi, \mathcal{L})$*, where* $\mathcal{Q}$ *is the set of regions in the environment,* $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ *is the set of edges defining adjacent regions that swarms can travel between,* $\mathcal{W} : \mathcal{E} \to \mathbb{Z}_{\geq 1}$ *maps each transition in* $\mathcal{E}$ *to its travel duration, a stationary swarm is modeled as an unit-weight self-transition,* Π *is a set of atomic propositions*[1]*, and* $\mathcal{L} : \mathcal{Q} \to 2^{\Pi}$ *is a labeling function.*

In Fig. 3.1.1, we show a scenario with multiple regions of interest and the corridors that

[1] Atomic propositions are defined in Sec. 3.1.2

connect the regions, which generates the abstracted environment Env in orange.

Agents' dynamics and sensing

We consider a set of agents $\mathcal{A} = \{\mathbf{x}_1, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N\}$ where $|\mathcal{A}| = N$ is the total number of agents available in the environment. Each agent's state is given by the location in the environment, either a state $q \in \mathcal{Q}$ or an edge $e \in \mathcal{E}$ that $\mathbf{x}_i$ is traversing, and an altitude $h_i \in \{1, \ldots, N_s\}$, where N_s is the number of altitude levels that agents are allowed to fly at. The state of a single agent is given by the tuple $\mathbf{x}_i = (q/e, h_i)$. Altitudes h_i capture the level number and not quantity which is not required for planning. We assume that there are a bounded number N_s of altitude levels h that agents can occupy. This requirement captures division or airspace for safe navigation, e.g., quadrotors need at least one-meter vertical separation to avoid downwash effects. A sub-swarm is defined by the agents $\mathbf{x}_i$ that share altitude $h_i = h$. Thus, we differentiate between sub-swarms based on their altitudes. Moreover, the maximum number of sub-swarms at any given time during the mission is bounded by N_s.

Although we are particularly interested in the high-level planning for swarms under temporal and logic-constrained specifications, we consider some assumptions in the lower-level control that drive modeling and solution design. Note that implementation with real robots requires low-level controllers to execute high-level plans. This can be implemented using standard formation control [54, 113] and attitude control [96, 139] methods.

Assumption 3.1.1. *We assume agents cannot communicate with each other, avoiding direct interaction between agents. Instead, we consider agents perform transitions between regions in $\mathcal{Q}$ using controllers that require only local information, e.g., formation control using the relative pose of neighbors by sensing them [147, 73].*

Assumption 3.1.1 is crucial for defining and modeling the splitting and merging swarm actions described in the following sub-section.

Swarms' state and actions

In this section, we are interested in commanding the entire swarm (or swarms) instead of individual agents. Multiple sub-swarms may exist due to swarms' splitting or merging as demanded by the specification. For instance, to fulfill tasks that need to be satisfied in overlapping time intervals, we require multiple sub-swarms to be present in disjoint regions simultaneously. We take inspiration from recent work on controlling swarms as abstract objects in the environment that allow us to send global commands to the swarm, not individual agents. As is shown in Fig. 3.1.1, we abstract the swarm [9, 114, 145, 178, 19] as ellipses, where the mean and covariance of the agents' position represent the center and shape of the swarm, respectively.

The center of the swarm μ and shape Σ are required for the low-level control of the swarms' position, heading, and size using communication-free formation control techniques [120, 42, 59]. However, since we abstract the environment as a transition system Env, μ, and Σ are unnecessary for high-level planning. Instead, we use the location in the environment described by a state $q \in \mathcal{Q}$ or an edge $e \in \mathcal{E}$ to capture the sub-swarms' states and motion.

Definition 3.1.2 (Swarm)**.** *A swarm is a tuple* $\mathbf{s} = (\mathcal{A}_s, h_s, q/e)$ *with* $\mathcal{A}_s \subseteq \mathcal{A}$ *being the set of agents that belong to the swarm* s *and that share common altitude* h_s*, and* q/e *represents that the swarm is either at state* $q \in \mathcal{Q}$ *or traversing edge* $e \in \mathcal{E}$*.*

The set of active sub-swarms at time k is denoted by $\mathcal{S}(k) = \{\mathbf{s}_0, \ldots, \mathbf{s}_i, \ldots\}$ which is time-varying. However, we assume there is an a priori known upper-bound $N_s \geq |\mathcal{S}(k)|$, for all $k \in \mathbb{Z}_{\geq 0}$, which is the maximum number of swarms that can exist simultaneously in the environment. The number of sub-swarms changes with time depending on the mission's requirements. Sub-swarms are created and deleted according to splitting or merging actions defined as follows.

Definition 3.1.3 (Splitting action)**.** *Splitting is the action of taking a swarm* $\mathbf{s}_i \to \{\mathbf{s}_{i_1}, \ldots,$

$\mathbf{s}_{i_n}\}$ *and transforming it into* n *sub-swarms such that* $\mathcal{A}_{s_i} = \bigcup_{\ell=1}^{n} \mathcal{A}_{s_{i_\ell}}$ *and* $\mathcal{A}_{s_\ell} \cap \mathcal{A}_{s_{\ell'}} = \varnothing$, $\forall \ell \neq \ell' \in \{i_1, \ldots, i_n\}$. *Additionally, all of the resulting sub-swarms have different altitudes,* $h_{s_\ell} \neq h_{s_{\ell'}}$, $\forall \ell \neq \ell' \in \{i_1, \ldots, i_n\}$.

Splitting action will occur when time-overlapping tasks require multiple swarms at different states $q \in \mathcal{Q}$ in the environment Env.

Definition 3.1.4 (Merging action)**.** *Merging is the action of taking a set of* n *sub-swarms* $\{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\} \rightarrow \mathbf{s}_i$ *and transforming them into a single swarm such that* $\mathcal{A}_{s_i} = \bigcup_{\ell=1}^{n} \mathcal{A}_{s_{i_\ell}}$. *Additionally, all of the agents converge to the same altitude* $h_j = h_{s_1}$, $\forall j \in \mathcal{A}_{s_i}$, *of the first sub-swarm by convention.*

Merging sub-swarms is desirable when multiple sub-swarms are not required for mission satisfaction since it may reduce the computational cost of tracking and controlling them. To simplify the problem, we assume that merging and splitting actions can only occur at states $q \in \mathcal{Q}$ and not while traversing the edges $e \in \mathcal{E}$.

Mission Specification Using Metric Temporal Logic

We define the semantics of MTL with respect to the swarms' trajectories (plan) $\boldsymbol{\alpha}$.

Definition 3.1.5 (Plan)**.** *A plan is the* joint state trajectory *generated by the sequence of swarms navigating in the environment* $\boldsymbol{\alpha} = \boldsymbol{\alpha}_0 \boldsymbol{\alpha}_1 \boldsymbol{\alpha}_2 \ldots \boldsymbol{\alpha}_{\|\phi\|}$, *where* $\boldsymbol{\alpha}_k = \mathcal{S}(k)$ *and each sub-swarm* $s_i = (\mathcal{A}_{s_i}, h_{s_i}, q_i) \in \mathcal{S}(k)$ *at time* k *either (1) moves in the environment using edge* $e = (q_i, q')$, *i.e.,* $(\mathcal{A}_{s_i}, h_{s_i}, q') \in \mathcal{S}(k + \mathcal{W}(e))$, *(2) performs a split action* $\mathbf{s}_i \rightarrow \{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\}$, *i.e.,* $(\mathcal{A}_{s_\ell}, h_{s_\ell}, q) \in \mathcal{S}(k+1)$, $\forall \ell \in \{i_1, \ldots i_n\}$, *or (3) performs a merging action* $\{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\} \rightarrow \mathbf{s}_j$, *i.e.,* $(\bigcup_{\ell=i_1}^{i_n} \mathcal{A}_{s_\ell}, h_{s_1}, q) \in \mathcal{S}(k+1)$ *and* $i \in \{i_1, \ldots, i_n\}$.

For proposition $\pi \in \Pi$,

$$\boldsymbol{\alpha}_k = (\mathbf{s}_i = (\mathcal{A}_s, h_s, q_i), k) \vDash \pi \iff \pi \in \mathcal{L}(q_i)\,, \tag{3.1}$$

meaning that an atomic proposition is satisfied if at least one swarm is in a region q_i labeled with that atomic proposition π. The semantics of the remaining operators is defined as usual [87, 81].

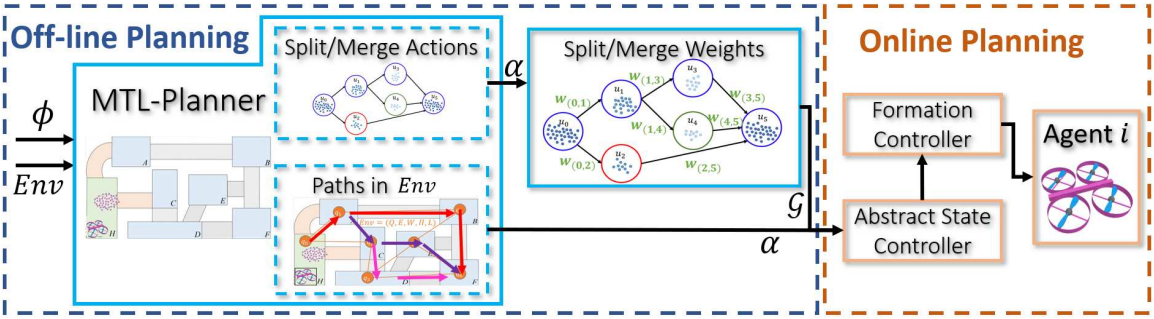


Figure 3.1.2: Block diagram of proposed control framework.

Problem

For simplicity, below, we define two problems that describe (1) how swarms split, merge, and move in the environment and (2) what fraction of agents each sub-swarm has. Note that in a plan $\boldsymbol{\alpha}$, swarms are created as required by the specification while merging is enforced via cost function $\mathcal{J} = \sum_{k=0}^{\|\phi\|} |\mathcal{S}(k)|$.

Problem 3.1.1. *Given a team of N agents, an abstracted environment Env, swarm splitting upper-bound N_s, and an MTL specification ϕ, find plan $\boldsymbol{\alpha}$ for the creation of swarms and their trajectories $\mathbf{s}_i(k)$ such that ϕ is satisfied and cost $\mathcal{J}$ is minimized.*

The sequence of actions in $\boldsymbol{\alpha}$ involves not only the sequence of how to traverse the environment to satisfy the mission but also the sequence of splitting and merging actions performed by the swarm. If the number of sub-swarms at any time during the mission is close to the upper-bound N_s, it is necessary to ensure an appropriate number of agents for all of them. Note that we do not assign roles to agents or pre-allocate them to any swarm since we assume there will be no communication between agents as described in Assumption 3.1.1. In contrast, agents must find a way to decide which of the swarms to join. Formally, we define this problem as follows.

Problem 3.1.2. *Given a symbolic plan $\boldsymbol{\alpha}$ which provides the information about the number of swarms $\mathcal{S}(k)$ at all times, find the appropriate splitting fractions for all of the agents such that any swarm at any time during the mission has a predefined minimum number of agents. Additionally, find a protocol without communication between agents that enforces the splitting fractions.*

3.1.3 Control Synthesis Solution

This section discusses the components required to solve Problem 3.1.1 and Problem 3.1.2. We first solve the problem of finding a plan $\boldsymbol{\alpha}$, the sequence of actions for visiting the regions and splittings or merging actions of the swarm that satisfies ϕ. Then, given such a sequence of splitting or merging actions from the initial swarm throughout the mission horizon $\|\phi\|$, we propose an algorithm for assigning splitting fractions such that the sub-swarms are balanced given a motion plan.

MILP Encoding for Swarm Planning

Here, we formulate a MILP encoding for capturing satisfaction of the MTL specification, which not only indicates the times when a region needs to be visited but also a sequence of splittings and merging actions of the swarm in the environment. First, we define integer variables $z_{q,k}$ and $z_{e,k} \in \mathbb{Z}$ to indicate the number of sub-swarms at state $q \in \mathcal{Q}$ or edge $e \in \mathcal{E}$ at time $k \in \{0, \ldots, \|\phi\|\}$, respectively. We set the initial condition at $k = 0$ by setting $z_{q_0,0} = 1$, $z_{q,0} = 0$ for all $q \neq q_0$ which states that the entire swarm starts at q_0. The number of sub-swarms we have in the environment at any time is not larger than N_s, which induces the constraints $z_{q,k}, z_{e,k} \in [0, N_s]$, for all $q \in \mathcal{Q}$, $e \in \mathcal{E}$ and $k \leq \|\phi\|$. Taking into account the

swarm variables, we capture the flow dynamic constraints in the environment as follows

$$z_{q,k} \geq z_{e,k-W(e)},\ \forall e = (q', q) \in \mathcal{E},\ k \leq \|\phi\|, \tag{3.2}$$

$$z_{e,k-W(e)} \leq z_{q,k-W(e)},\ \forall e = (q, q') \in \mathcal{E},\ k \leq \|\phi\|, \tag{3.3}$$

Notice that (3.2) and (3.3) ensure that swarms will go to required regions. However, to make sure the swarms can only travel to neighboring nodes, the creation (splitting) of swarms will occur only at states, and merging swarms are all present at the same state, we impose the following constraints

$$z_{q,k} \leq \sum_{e=(q',q)\in\mathcal{E}} z_{e,k-W(e)}, \forall q \in \mathcal{Q},\ k \leq \|\phi\|, \tag{3.4}$$

$$\sum_{e=(q,q')\in\mathcal{E}} z_{e,k} \geq z_{q,k},\ \forall q \in \mathcal{Q},\ k \leq \|\phi\|. \tag{3.5}$$

The flow constraints capture how a swarm (swarms) travel through the environment Env. However, they do not constrain the number of sub-swarms we have at any time. For this purpose, we introduce the following split-bound constraint

$$\underbrace{\sum_{e\in\mathcal{E}} \sum_{k'\in H(e,k')} z_{e,k'}}_{\text{Number of sub-swarms at time k}} \leq N_s, \tag{3.6}$$

where $H(e,k) = \{\{k - W(e) + 1, ..., K\} \cap \mathbb{Z}_{\geq 0}\}$ which is the history of departures for edge e at time k.

Then, we formulate Problem 3.1.1 as the following MILP optimization problem

$$\begin{aligned} &\min \sum_{k=0}^{\|\phi\|} \left(\sum_{q \in Q} z_{q,k} + \sum_{e \in \mathcal{E}} z_{e,k} \right), \\ &\text{subject to} \\ &\text{Swarm flow dynamics (3.2) – (3.5),} \\ &\text{Split bound (3.6),} \\ &\text{Mission satisfaction: } \{z_{q,k}\} \vDash \phi. \end{aligned} \tag{3.7}$$

Note that (3.7) discourages the unnecessary swarm splitting and traveling in the environment over time horizon $\|\phi\|$. Mission satisfaction $\{z_{q,k}\} \vDash \phi$ is encoded similarly to [95, 29, 30], and we omit it for brevity.

Swarm Splitting and Merging Actions Weights From the swarm planning stage, we obtain sequences of swarm splitting and merging actions of sub-swarms necessary to satisfy the mission specification. The sequences are captured as a Directed Acyclic Graph (DAG) $G = (V, E, w)$, where V is the set of nodes, and each node is defined by a tuple $u = (q, k)$ meaning that there is a swarm at location q in the environment at time k. Edges $E \subseteq V \times V$ capture splitting (i.e., multiple outgoing edges from a state u) and merging (i.e., multiple incoming edges to a state u). Let $\mathcal{N}_u^+ = \{v \mid (u, v) \in E\}$ and $\mathcal{N}_u^- = \{v \mid (v, u) \in E\}$ be the sets of outgoing and incoming edges of state u. A leaf node u has no outgoing neighbors, $\mathcal{N}_u^+ = \varnothing$. The weights $w : E \to \mathbb{R}_{>0}$ represent splitting fractions such that the unit-sum constraint $\sum_{v \in \mathcal{N}_u^+} w((u, v)) = 1$ holds for all non-leaf $u \in V$ including the starting node $u_0 = (q_0, 0)$. Thus, $w(u) \in (0, 1]$ for all $u \in V$.

We compute splitting fractions proportional to the requirements on parallel paths in the DAG G to balance swarm sizes over locations and time. Formally, we capture this using a partial order [44]. We define the partial order $\preccurlyeq$ over U such that $u \preccurlyeq v$ if a path exists

from u to v, i.e., if u is an ancestor of v or v is a descendent of u. The sets of ancestors and descendants are $\mathcal{H}_u = \{v \mid v \preccurlyeq u\}$ and $\mathcal{F}_u = \{v \mid u \preccurlyeq v\}$, respectively. Let $p : V \to (0, 1]$ function such that $p(u)$ is the fraction of the swarm agents at node $u \in V$.

Algorithm 3 computes proportions p backward from leaf nodes in G, and then computes the splitting fraction weights w. First, the method computes the number of ancestors and descendants of nodes for all nodes $u \in V$, lines 3-8. Moreover, we compute the total number of ancestors of all leaf nodes $\mathcal{H}_{max}$ at line 6. For all leaf nodes u with $\mathcal{N}_u^+ = \varnothing$, the proportion $p(u) = \frac{|\mathcal{H}_u|}{\mathcal{H}_{max}}$, line 9. For all other nodes, we proceed in inverse topological order (i.e., from leaves towards u_0) and compute $p(u) = \sum_{v \in \mathcal{N}_u^+} \frac{p(v)}{|\mathcal{N}_u^-|}$, line 10. Finally, the weights are computed as $w((v, u)) = \frac{p(u)}{p(v)}$ for all edges $(v, u) \in E$, line 11. It follows from Algorithm 3 that the returned weights w satisfy the unit-sum constraints.

Algorithm 3 Splitting and Merging Weights over DAG

1: **input:** $G = (V, E)$ ▷ DAG from MILP solution.
2: **output:** $G = (V, E, w)$ ▷ DAG with splitting fraction weights.
3: **for** $u \in V$ **do**
4: $|\mathcal{H}_u| = ancestors(\mathcal{G}, u)$
5: $|\mathcal{F}_u| = descendants(\mathcal{G}, u)$
6: **if** $|\mathcal{F}_u| = \varnothing$ **then** $\mathcal{H}_{max} = \mathcal{H}_{max} + |\mathcal{H}_u|$
7: **end if**
8: **end for**
9: For all $u \in leaves$, $p_u = |\mathcal{H}_i|/\mathcal{H}_{max}$
10: For all $u \in V \setminus leaves$, compute $p(u) = \sum_{v \in \mathcal{N}_u^+} \frac{p(v)}{|\mathcal{N}_u^-|}$ in inverse topological order.
11: For all $(u, v) \in E$ compute weights $w((v, u)) = \frac{p_u}{p_v}$, $\forall (v, u) \in E$ in a topological order.

Protocol: Finally, we define the communication-free protocol that agents implement to determine what sub-swarms they belong to when receiving split and merge actions. Communication is restricted between agents, but they receive global commands from the MTL-Planner, see Fig. 3.1.2. For *splitting*, all agents receive the command $\mathbf{s}_i \to \{\mathbf{s}_{i_1}, \ldots, \mathbf{s}_{i_n}\}$ containing the altitude h_{s_i} of the sub-swarm $\mathbf{s}_i$ to be split, heights h_ℓ for all resulting sub-swarms $\mathbf{s}_\ell$, $\ell \in \{i_1, \ldots, i_n\}$ and the splitting fraction weights $w((u, v))$, $v \in \mathcal{N}_u^+$, where node u in DAG G corresponds to the splitting action. Note that $n = |\mathcal{N}_u^+|$. If an agent is at altitude

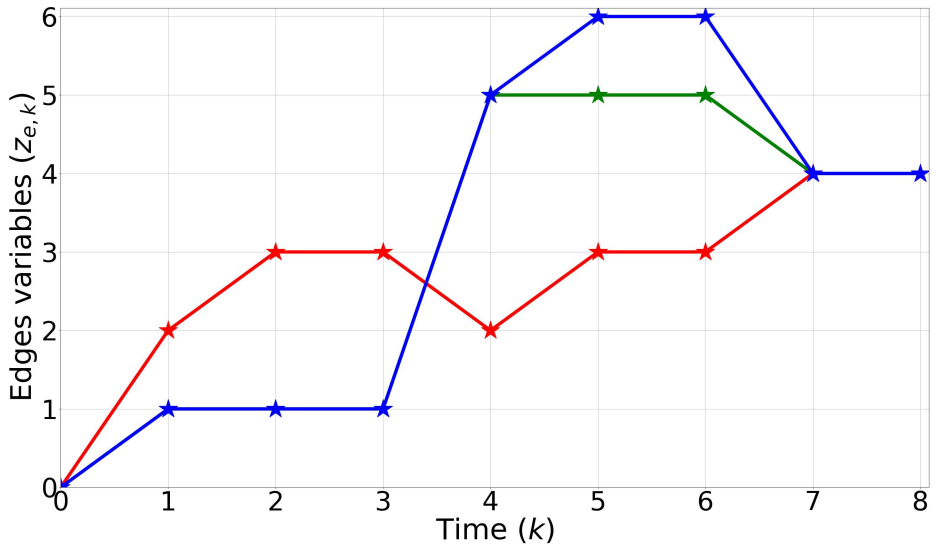


Figure 3.1.3: Edges variable solution from case study 1.

h_{s_i}, i.e., belongs to s_i, then it samples the set of sub-swarms $\{\mathrm{s}_{i_1}, \ldots, \mathrm{s}_{i_n}\}$ with probabilities $w((u, v))$ and moves to the altitude $h_{\bar{s}}$, where each neighbor node v corresponds to a sub-swarm s_ℓ, $\ell \in \{i_1, \ldots, i_n\}$, and $\bar{s}$ is the randomly sampled sub-swarm. For *merging*, all agents receive the command $\{\mathrm{s}_{i_1}, \ldots, \mathrm{s}_{i_n}\} \to \mathrm{s}_i$ containing the altitudes h_{s_ℓ} of sub-swarms s_ℓ to be merged, $\ell \in \{i_1, \ldots, i_n\}$. In this case, all agents with altitudes h_{s_ℓ} move to altitude $h_{s_{i_1}}$ of resulting sub-swarm s_i. Note that the protocol guarantees the provided swarm splitting fractions only in expectation. However, it does not require any communication between agents.

3.1.4 Case Studies

This section describes three case studies showing how swarm planning and control work and their performance. First, we consider a small mission specification to showcase the functionality of the high-level planner and then the low-level control. Second, we specify a more complex specification that requires the swarm to split and merge multiple times to show the correctness of the merging and splitting algorithm described in Sec. 3.1.3. Finally, we test scalability and run-time performance by increasing the mission specification size

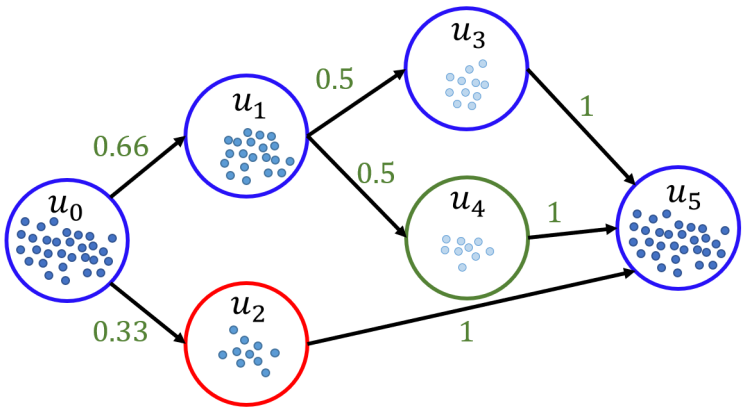


Figure 3.1.4: Split and merge generated DAG for case study 1.

gradually. All computation was performed on a PC with 20 cores at 3.7GHz with 64 GB of RAM. We used Gurobi [66] as MILP solver and CoppeliaSim [132] as simulator.

Case Study 1: (functionality showcase)

Here, we consider the Env in Fig. 3.1.1, the number of robots in the swarm $N = 80$. The swarm is tasked to satisfy the following mission specification $\phi = (\square_{[1,4]} q_1 \wedge \square_{[2,4]} q_3) \wedge \square_{[6,7]}(q_3 \wedge q_5 \wedge q_6) \wedge \square_{[8,9]} q_4$, with initial location of the swarm q_0, and maximum simultaneously existing swarms $N_s = 3$. Under these conditions, we get the solution plan shown in Fig. 3.1.3. Initially, the entire swarm is in q_0 at time $k = 0$. Next, it splits in two, represented here as red and blue lines traversing to states q_1 and q_2 at $k = 1$. Both swarms satisfy the first part of the specification, and then the blue swarm splits again at $k = 4$, where the green line represents the formation of a new swarm to satisfy the specification requesting sub-swarms in three different locations simultaneously. Lastly, the three swarms merge in q_4. The MILP solution computed a solution with an objective cost value of 39, and it took 0.01 seconds to compute, and 228 and 17 integer and binary variables, respectively.

From the MILP solution, we can compute the DAG $\mathcal{G} = (V, E, w)$ that describes the swarm splitting and merging actions in Sec. 3.1.3 and shown in Fig. 3.1.4. Note that the nodes follow the same color code as in Fig. 3.1.3. The agents in the initial swarm (blue) split into sub-swarm u_1 (blue) and u_2 (red). Then u_1 splits into two more sub-swarms u_3

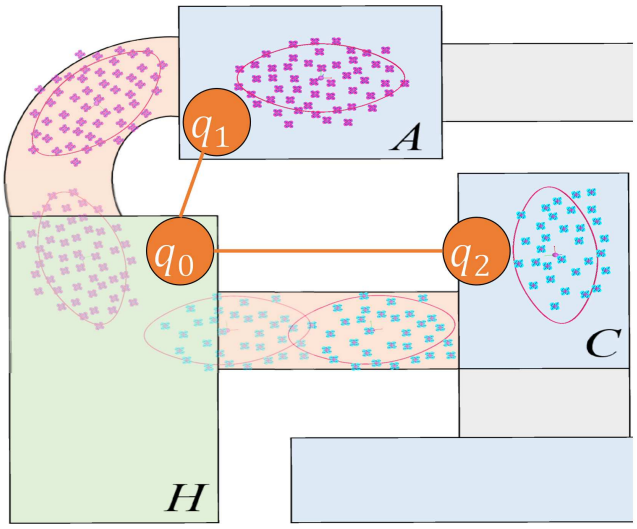


Figure 3.1.5: Solution performed in CoppeliaSim with actions by the swarms when traversing corridors connecting two states.

(blue) and u_4 (green). Lastly, all of them merge in u_5, the same as in Fig. 3.1.3. We show the splitting fraction (probabilities) of the agents going into the different swarms on the edges. The weights are computed to balance the number of agents in each sub-swarm by taking into account the subsequent merging and splitting actions. An extended example is shown in Case study 2.

Finally, we compute the abstract state commands, as shown in Fig. 3.1.5, where two swarms are traveling from q_0 to q_1 and q_2 respectively, representing the first command from the solution of the MILP. Note that one of the corridors is curved, requiring the swarm to rotate and scale along it. For the second swarm, it is asked to rotate once it reaches the state to be able to traverse towards q_3. Simulation is performed in CoppeliaSim.

Case Study 2

In this case study, we consider an extended example that showcases the functioning of the swarm splitting and merging algorithm. Consider the following mission specification $\phi = \square_{[1,2]}(q_1 \wedge q_2) \wedge \square_{[2,3]}(q_1 \wedge q_3 \wedge q_4 \wedge q_6) \wedge \square_{[3,4]}(q_5 \wedge q_3) \wedge \square_{[4,5]}(q_5 \wedge q_0 \wedge q_2)$, with maximum number of swarms $N_s = 4$ and initial state q_0. After computing the MILP solution, we build

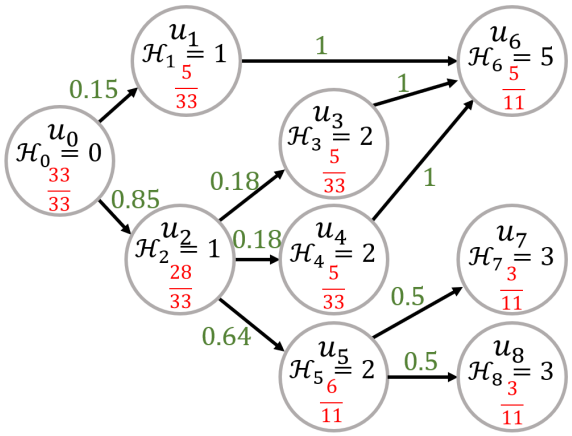


Figure 3.1.6: Split and merge generated DAG for case study 2.

the merging and splitting actions DAG shown in Fig. 3.1.6. Following Algorithm 1. the number of ancestors $|\mathcal{H}_i|$ for every node is computed. We sum the number of the ancestor overall leaf nodes $\mathcal{H}_{max} = \sum_{u_i} |\mathcal{H}_i| = 5 + 3 + 3 = 11$, and then the expected proportion of the leaves is obtained $p_u = \mathcal{H}_i/\mathcal{H}_{max}$ as $5/11, 3/11, 3/11$, respectively. Next, from the leaves, we sum the proportions backward from the leaves to the root (shown as red values in the nodes). Finally, we compute each edge's splitting fractions (probabilities) from the root by considering the expected proportions in every node (green values on the edges). In Fig. 3.1.7, we show the agent distribution among the states. The horizontal axis represents the state and the vertical axis the number of agents in the swarm. We ran the same DAG 1000 times and considered two different initial number of agents $|\mathcal{A}_s| = 50$ (blue) and $|\mathcal{A}_s| = 150$ (red). Note that the smaller the swarm, the more difficult it becomes to split it in balanced proportions. Moreover, in both cases, node u_2 has the most significant fraction of agents, which makes sense since this node requires the swarm to split again into three new sub-swarms, and one of them will be split into the other two. In contrast, u_1 does not require a higher fraction since later, it will only merge with other sub-swarms.

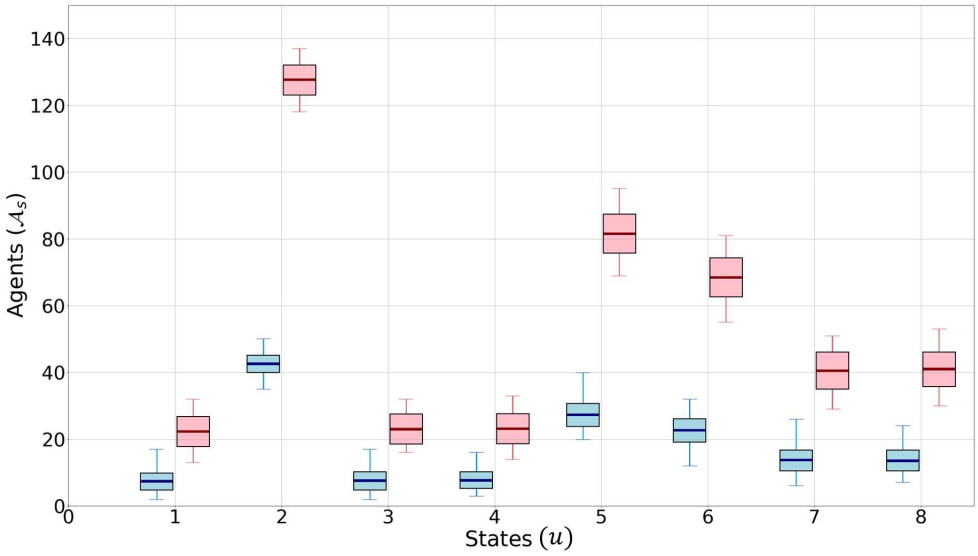


Figure 3.1.7: Agents distribution statistics through the states when run 1000 times for an initial number of agents of 150 and 50.

Case Study 3: (scalability and time performance)

Here, we gradually increase the size of the mission specification to show the run time performance. The mission specification we use is $\phi = \bigwedge_1^n \mathcal{T}_I(\mathcal{X} \otimes \mathcal{X})$, where $\mathcal{T} \in \{\Box, \Diamond\}$, $\otimes \in \{\wedge, \vee\}$, $\mathcal{X} \in \mathcal{Q}$ are variables randomly chosen, n is an iterator that grows from 1 to 200, and the time interval of the temporal operator is defined randomly as $I = [n + 4, \ldots, n + \text{rand}(1, 5)]$. The initial state is q_0, and the split bound $N_s = 10$. The results are shown in Fig. 3.1.8, where we can see that time grows linearly for a small value of n and then exponentially as n becomes larger. Note that our MILP implementation is agent agnostic, and the only variables that affect the performance are the size of the transition system and the mission specification.

3.1.5 Conclusions

We present a framework for controlling swarms subject to Metric Temporal Logic (MTL) constraints. We propose a MILP approach for computing the trajectories of sub-swarms in the environment, the splitting and merging actions if required, such that the MTL mis-

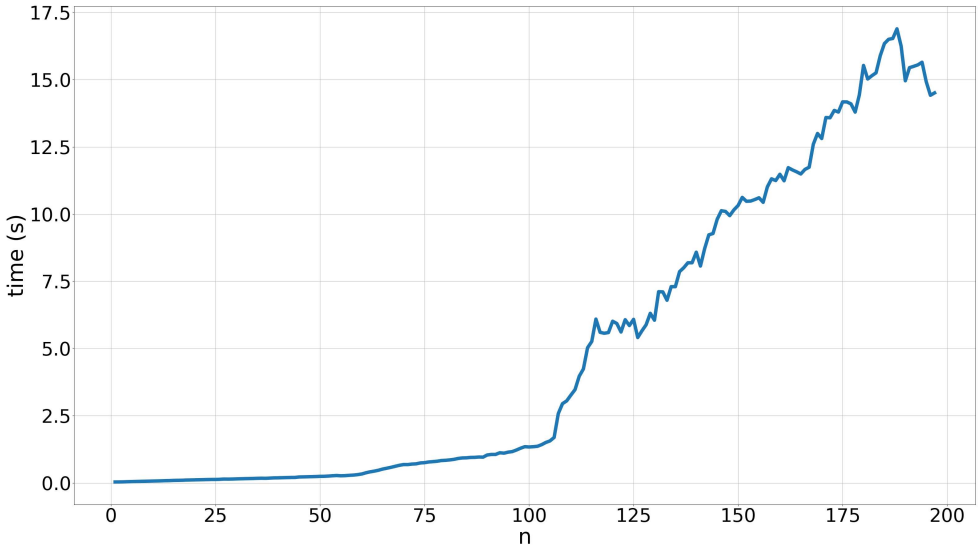


Figure 3.1.8: Time performance by increasing mission ϕ gradually.

sion is satisfied with minimal division of the swarm over the mission horizon. We modify standard flow dynamics constraints using inequalities to capture the swarm motion in the environment. The solution of the MILP is used to build a DAG that captures the splitting and merging actions and is used to compute splitting fractions for resulting sub-swarms. A distributed randomized protocol ensures that the agents choose sub-swarms close to the computed splitting fraction with no inter-agent communication.

3.2 Planning for Modular Aerial Robotic Tools with Temporal Logic Constraints

Modular robots are highly versatile because they can reconfigure and change their mechanical properties. This ability makes them optimal for scenarios that require different types of tasks. However, task allocation and cooperation become a combinatorial problem when the number of modules increases. To tackle this problem, we propose a high-level planner for reconfigurable robots with heterogeneous capabilities, e.g., aerial motion and tool operation. Modules can attach and detach to create configurations that manipulate tools, satisfying temporal and logic-constrained tasks. The mission is specified using Metric Temporal Logic (MTL), which offers the capacity to not only account for where and who needs to satisfy a task but also when and for how long. We model the problem using a Mixed Integer Linear Problem (MILP) approach, capturing cost for reconfiguration, satisfying a task, and motion in the environment in a specific configuration. Additionally, we consider that not all configurations can satisfy every task. We find trajectories for modular robots that guarantee mission satisfaction. Finally, we show the performance in simulations with multiple tasks and requirements in an environment.

3.2.1 Introduction

In recent years, the development of technology and the increase in computational capacity have implemented multi-robot systems possible. Multi-robot systems have been widely studied for their capacity to handle multiple tasks simultaneously, robustness, and resiliency in overcoming failures while guaranteeing task satisfaction. This type of system is a suitable platform for tasks ranging from perimeter surveillance, search and rescue missions, cargo delivery, and planetary exploration [48, 157, 23]. Nevertheless, for tasks requiring more capabilities, such as versatile locomotion and manipulation, modular robots can reconfigure and change both force capacity in the environment and control constraints [144].

Modular reconfigurable robots that change from one configuration to another offer flexibility to address a broader set of tasks than multiple single robots. Here, configuration is understood not only as the pose of the robot but also the module connectivity and shape [144]. Changing their capacities and the possible redundancy in the degree of freedom make modular robots versatile and robust [4]. However, there is an increase in the computational cost when coordinating the robots in both low-level and high-level control. Low-level control is required to guarantee the collision-free rearrangement of modular robots to reach the desired configuration (lattices or chains) and act [138, 102, 123, 101]. On the other hand, the high-level control takes a set of real-world tasks and generates or matches existing configurations to satisfy the mission specification [166].

Here, we focus on high-level planning, coordinating modular robots to satisfy different types of tasks in an environment. Multiple works have tackled this problem of coordinating and allocating a set of robots to satisfy tasks from a deterministic and stochastic approach [43, 12]. However, they consider agents homogeneous and do not consider temporal logic constraints in the specifications. Several other works have used Temporal logic for heterogeneous multi-robot systems working with Linear Temporal Logic (LTL) with automata theory [143, 88] or Signal Temporal Logic [95, 150]. Nevertheless, these papers do not consider physical connections or modularity, which adds to the problem of not only allocating some robots but also finding proper configurations that satisfy a particular task.

Our work focuses on generating an automatic controller for synthesizing module trajectories, allowing modular robots to find configurations that satisfy task specifications using formal languages, specifically Metric Temporal Logic (MTL). Few works have tackled modular robots using Temporal Logic approaches [74], [34]. Nonetheless, authors consider LTL and encode the system into an automaton, which might be computationally expensive when dealing with scalability. Instead, we consider MTL a rich temporal logic formalism. We propose a MILP approach to efficiently solve the problem while accounting for the cost of reconfiguration, motion, and satisfying a task in a specific configuration.

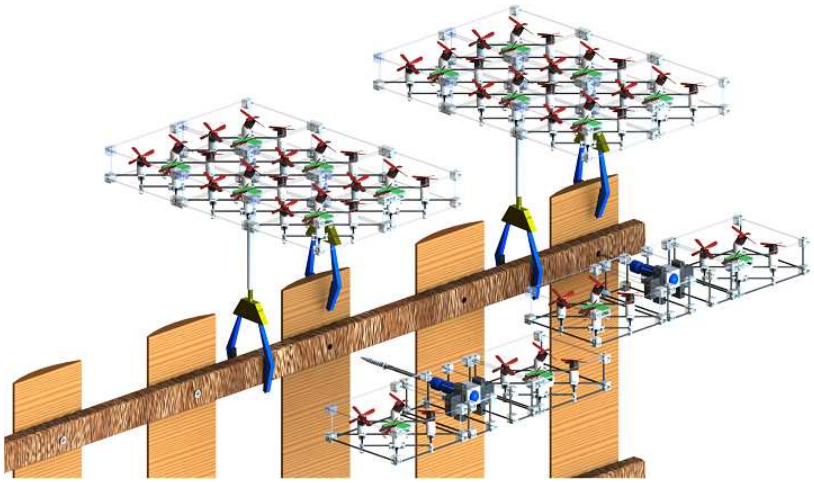


Figure 3.2.1: Planning for multiple modular robots that can reach configurations to manipulate tools such as a gripper to grasp wood and screwdrivers to build a wooden fence.

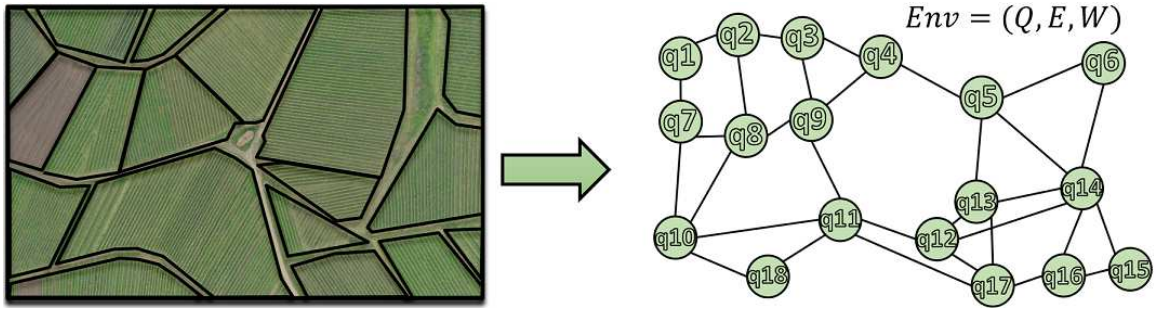


Figure 3.2.2: Example of a tessellated agriculture environment (left) that can be abstracted into a transition system (right).

The contributions of the paper are threefold. 1) We propose and formalize a planning problem for modular aerial robots with heterogeneous capabilities and configurations tasked with performing timed temporal logic missions. The missions involve tasks that can be performed only by a subset of robot configurations and may require robots to reconfigure during the mission. 2) We propose an efficient MILP approach that minimizes the total energy consumption while satisfying the MTL mission specification, robot motion, and reconfiguration constraints. 3) We show the performance of the proposed MILP method in two case studies.

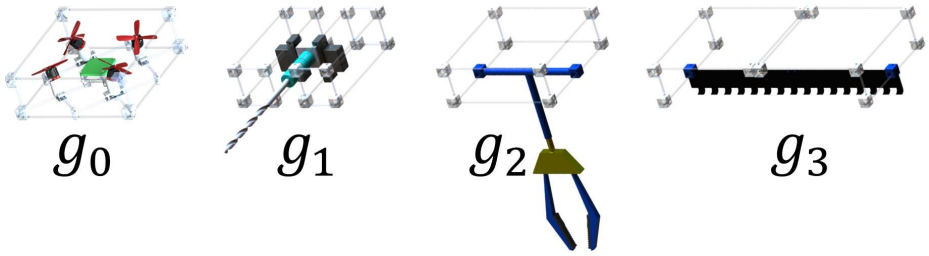


Figure 3.2.3: Example of module capabilities, form left to right g_{fly}, $g_{screwdriver}$, $g_{grasping}$, g_{sawing}.

3.2.2 Problem Formulation

In this section, we introduce the planning problem for teams of heterogeneous modular robots tasked with rich temporal logic tasks that require the use of specialized tools, e.g., a driller, screwdriver, and grasping. Robots can reconfigure during the mission to perform the various specification tasks. Thus, the number and configurations of modular robots active in the mission space may be time-varying during the mission. We introduce the models for the environment, modules, configurations, robots, and tasks that define the planning problem for modular robots. We are motivated by construction problems for agriculture. For instance, consider the construction of a wooden fence, as shown in Fig. 3.2.1. The task is described in the following Example 3.2.1.

Example 3.2.1. *Fence construction on crop q_1:*

1. *Always from deployment to the end of the mission, region q_1 must be video-monitored;*
2. *Within 10 minutes after deployment, the wood must be transported in q_1;*
3. *Within 11 to 60 minutes after deployment, the wood must be cut into smaller pieces at q_1;*
4. *Always within 60 to 90 minutes, the fence needs to be assembled at q_1.*

First, we define the *environment* that captures the motion of robots between locations of interest.

Definition 3.2.1 (Environment)**.** *The environment is a weighted transition system defined by the tuple* $Env = (\mathcal{Q}, \mathcal{E}, \mathcal{W}, \Pi, \mathcal{L})$*, where* $\mathcal{Q}$ *is a finite set of locations of interest (states);* $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ *is the set capturing the transitions of robots between locations in* $\mathcal{Q}$*;* $\mathcal{W} : \mathcal{E} \to \mathbb{Z}_{\geq 1}$ *maps each transition in* $\mathcal{E}$ *to its travel duration;* Π *is a set of atomic propositions that label the states* $\mathcal{Q}$*, and* $\mathcal{L} : \mathcal{Q} \to 2^{\Pi}$ *is the state-labeling function. A robot stationary at* $q \in \mathcal{Q}$ *is modeled as a unit-weight self-transition, i.e.,* $(q, q) \in \mathcal{E}$ *for all* $q \in \mathcal{Q}$*, and* $\mathcal{W}((q, q)) = 1$*.*

In Fig. 3.2.2, we illustrate a tessellated agricultural environment, which generates the abstracted environment Env describing the states $\mathcal{Q}$ and edges $\mathcal{E}$. Extending from Example 3.2.1, we consider modular robots with a flying capability and able to carry and manipulate modular tools.

Definition 3.2.2 (Module)**.** *A module is a cuboid with a capability* g *interacting in the environment.*

The set of all capabilities is $\mathcal{G} = \{g_0, g_1, g_2, g_3, \dots g_m\}$. At least one capability is the ability to fly. The capability of a module also defines the module's class. Some examples of these modules are shown in Fig. 3.2.3. Note that we do not differentiate between modules of the same class. Additionally, modules from the same or different classes can attach and detach each other to form modular robots. Configurations describe the types, numbers, and interconnections of modules that robots compose.

Formally, we have the following definition.

Definition 3.2.3 (Configuration)**.** *A robot configuration* $c = (M_c, \to_c, Cap_c)$ *is a labeled graph, where the nodes* M_c *are modules, the* $\to_c \subseteq M_c \times M_c$ *represent the physical connections between modules, and* $Cap_c : M_c \to \mathcal{G}$ *denotes the capability of each module.*

The number of modules with capability g in configuration c is denoted by $\Lambda(c, g)$. Formally, we have $\Lambda(c, g) = |\{m \in M_c \mid cap_c(m) = g\}|$. A configuration is defined not only by the number of modules and the class of modules but also by the arrangement. We show

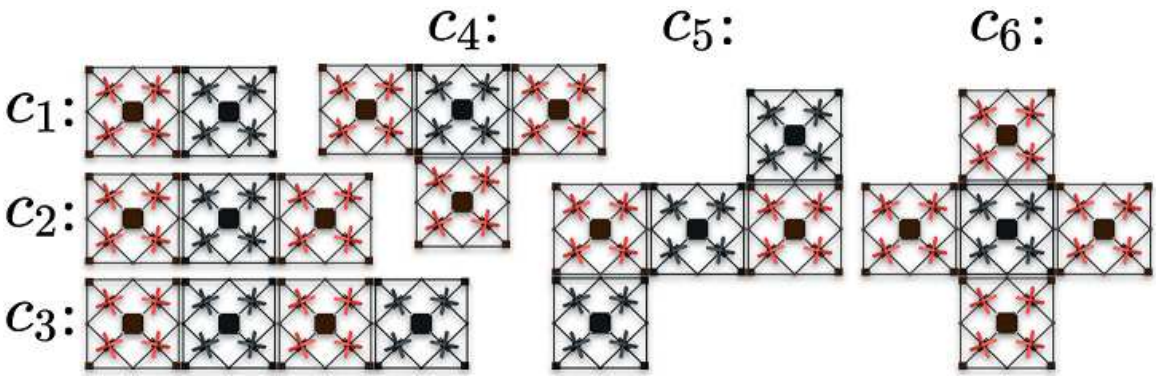


Figure 3.2.4: Examples of configurations of modules in different arrangements.

examples of different configurations in Fig. 3.2.4. Configurations c_3 and c_4 and configurations c_5 and c_6 have the same number of modules. However, the arrangement of modules is different. Lastly, a single module is a configuration with $|M_c| = 1$. Different configurations are advantageous for changing the robot's capabilities and performing different tasks. However, considering all possible configurations given a heterogeneous group of modules with capabilities $g \in \mathcal{G}$ leads to an intractable combinatorial problem, even for a few modules. The number of all possible configurations is in the order $\mathcal{O}(|\mathcal{G}|^{2\lambda})$, where λ is the number of modules available. Instead, we consider the following assumption.

Assumption 3.2.1. *The set of feasible robot configuration $\mathcal{C} = \{c_1, \ldots, c_n\}$ is given beforehand, and n is small.*

We are now ready to define *modular robot*, which are the actual agents moving and interacting with the environment to accomplish specified tasks.

Definition 3.2.4 (Modular Robot)**.** *A modular robot R_k at time $k \in \mathbb{Z}_{\geq 0}$ is defined by a configuration $c_{R_k} \in \mathcal{C}$ and a robot state $s_{R_k} \in \mathcal{Q} \cup \mathcal{E}$, where $s_{R_k} \in \mathcal{Q}$ denotes that R_k is at location s_{R_k} at time k, and $s_{R_k} \in \mathcal{E}$ denotes that R_k is moving along edge s_{R_k} at time k.*

Examples of modular robots are shown in Fig. 3.2.5. Robots R_1 and $R_2 \in \mathcal{R}$ use the same tool $g_{grasping} \in \mathcal{G}$, but differ in the number of quadrotors $g_{fly} \in \mathcal{G}$. The arrangement and, therefore, the configurations are different. Additionally, a robot with a saw tool

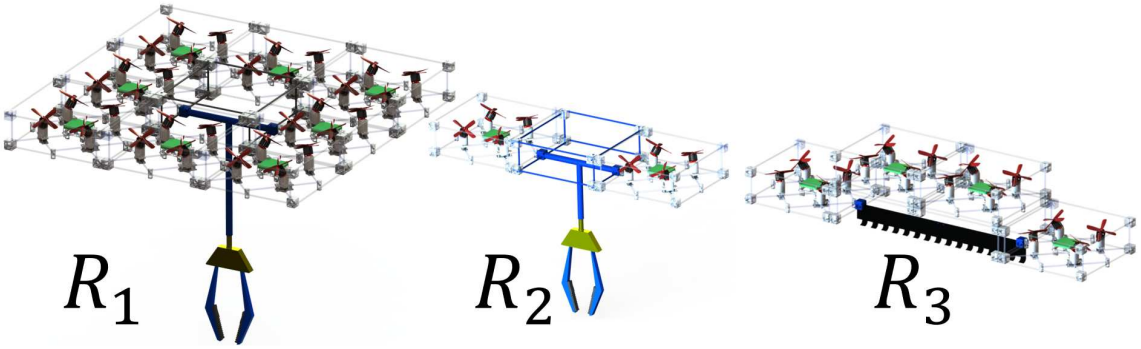


Figure 3.2.5: Examples of modular robots are composed of a set of modules with configurations that can manipulate tools such as grippers and saws.

$g_{sawing} \in \mathcal{G}$ and four quadrotors g_{fly} is shown. Note that robots are defined as depending on time. This is due to their fundamental ability to self-reconfigure, i.e., change their configurations $c_{R_k} \in \mathcal{C}$. Moreover, the number of robots each time k may differ. Modules from multiple robots may transform into a lesser number of robots, and vice versa. Thus, modular robots do not have a persistent identity and are considered a unit only while maintaining their configuration $c_{R_k} \in \mathcal{C}$. For simplicity, we allow robots to reconfigure only at states $q \in \mathcal{Q}$. We model the reconfiguration process as special self-loops $e^r = (q, q)^r$, for all $q \in \mathcal{Q}$. We abuse notation and consider $e^r \in \mathcal{E}$ in addition to normal self-loops (q, q) that capture stationarity[2]. Since we focus on high-level planning, we consider that low-level controllers for determining reconfiguration sequences of attaching and detaching of modules and their motion during the process are available [138, 101]. Thus, we consider reconfigurations at location $q \in \mathcal{Q}$ at time $k \in \mathbb{Z}_{\geq 0}$ *feasible* if the number of modules of each class $g \in \mathcal{G}$ in all robots involved remains unchanged. We assume that the upper bounds for the duration of any reconfiguration process at a state $q \in \mathcal{Q}$ are known and denoted by $W(e^r)$, where $e^r = (q, q)^r$. We denote the set of modular robots at time $k \in \mathbb{Z}_{\geq 0}$ by $\mathcal{R}_k$. We assume that the maximum number of robots is $p \in \mathbb{Z}_{\geq 1}$ known a priori at any moment.

Each robot $R_k \in \mathcal{R}_k$ with state $s_{R_k} = q \in \mathcal{Q}$ must take a control action $u_{R_k} \in \mathcal{E}$. Specifically, it either a) flies along a transition $(q, q') \in \mathcal{E}$, $q \neq q'$, b) remains stationary at q via

[2]Formally, Env takes the structure of a multi-graph with parallel self-loops in our case. For brevity, we define Env as a directed graph and denote reconfiguration self-loops with the superscript $()^r$ when needed.

self-loop transition $(q, q) \in \mathcal{E}$, or c) takes part in a reconfiguration process at q via self-loop transition $e^r = (q, q)^r \in \mathcal{E}$. We consider $u_{R_k} = \varnothing$ when the robots is in transition between locations, i.e., $s_{R_k} \in \mathcal{E}$.

Specification

The primitive units of our specifications are *tasks* that capture the required locations, duration, and robot capabilities.

Definition 3.2.5 (Task)**.** *A task is a tuple $T = (d, \pi, \mathcal{C}_T)$, where $d \in \mathbb{Z}_{\geq 0}$ is the duration of the task, $\pi \in \Pi$ is the label of all locations where the task needs to be performed, and $\mathcal{C}_T \subseteq \mathcal{C}$ is the subset of configurations that can perform the desired action, e.g., transporting, sawing, drilling.*

We define the configuration-task matrix $\Gamma(c, T) \in \mathbb{B}$ such that $\Gamma(c, T) = 1$ if $c \in \mathcal{C}_T$, and $\Gamma(c, T) = 0$ otherwise. We assume that configurations are characterized for tasks in the mission specification, and thus, Γ is available for planning. A robot must generate the desired wrench (forces and torques) to perform a task's action. The physical parameters, such as volume and weight, and robot configuration impact its performance in the tasks. In this section, we assume that all configurations $\mathcal{C}_T$ can perform task T within its duration d. We will address performance differences between configurations in future work. The duration of tasks may be expressed via the always operators in MTL. Formally, we have $T = \square_{[0,d]} \bigwedge_{q \in \mathcal{L}^{-1}(\pi)} \varpi_{q,T}$, where $\varpi_{q,T}$ denotes that there exists a robot at location $q \in \mathcal{Q}$ and configuration $c \in \mathcal{C}_T$ that can perform T. To capture the satisfaction of robot missions, we define the *joint output word* generated by the teams of robots as $\mathbf{o} = o_0 o_1 o_2 \ldots$, where $o_k = \{T = (d, \pi, \mathcal{C}_T) \mid \forall q \in \mathcal{L}^{-1}(\pi), \exists R_k \in \mathcal{R}_k \text{ s.t. } s_{R_k} = q, c_{R_k} \in \mathcal{C}_T\}$ captures the tasks performed at time k, disregarding their durations. Lastly, mission specifications are expressed as MTL formulas, with tasks taking the role of atomic propositions. As noted above, tasks are MTL formulas with additional semantics regarding their satisfaction by robots with

specific configurations.

Objective function

The motion, actions, and reconfiguration of robots' energy required to perform. Let $S_m = \{(e, R_k) \mid u_{R_k} = e = (q, q') \in \mathcal{E}, q \neq q'\}$ be the set of all robot motions, $S_r = \{(e, R_k) \mid u_{R_k} = e^r\}$ be the set of all robot reconfigurations, and $S_a = \{(e, R_k, T) \mid u_{R_k} = e = (q, q) \in \mathcal{R}, q \in \mathcal{L}^{-1}(\pi), c_{R_k} \in \mathcal{C}_T\}$ be the set of all robot actions. We define the cost function as

$$\begin{aligned} J &= J_r + J_m + J_a,\ J_r = \sum_{((q,q),R_k)\in S_r} \sum_{g\in\mathcal{G}} Y_{q,g}\,\Lambda(c_{R_k}, g), \\ J_m &= \sum_{(e,R_k)\in S_m} Y_{e,c_{R_k}},\ J_a = \sum_{((q,q)^r,R_k)\in S_a} Y_{q,T,c_{R_k}}, \end{aligned} \tag{3.8}$$

where $Y_{e,c} \in \mathbb{R}_{>0}$ is the energy required to traverse edge e with configuration c, $Y_{q,g} \in \mathbb{R}_{>0}$ is the per module energy cost for module class g at location q for reconfiguration, and $Y_{q,T,c} \in \mathbb{R}_{>0}$ is the energy required by a robot with configuration c to perform actions to satisfy task T at location q. Again, we assume that our configurations are characterized with respect to motion, reconfiguration, and tasks, and the energy costs $Y_{e,c}$, $Y_{q,g}$, and $Y_{q,T,c}$ are available for planning.

Now that we have defined all the framework components, we formally describe our problem as follows.

Problem 3.2.1. *Given an MTL mission specification ϕ and a set of modules with capabilities $\mathcal{G}$ deployed in Env that can assemble in a maximum number p of robots at each time $k \in \mathbb{Z}_{\geq 0}$ with configurations $\mathcal{C}$, find the set $\mathcal{R}_k$ and control actions u_{R_k} for all robots $R_k \in \mathcal{R}_k$ at each time $k \in [0 \,..\, \|\phi\|]$ such that $\mathbf{o} \vDash \phi$ and the cost J in* (3.8) *is minimized.*

3.2.3 Mixed Integer Linear Programming Approach

In this section, we formulate Problem 3.2.1 as a Mixed Integer Linear Program (MILP). In the following, we introduce the set of virtual robots $\mathcal{R} = \{R_1, \ldots, R_p\}$. Instead of keeping track of the set of robots $\mathcal{R}_k$ at each time $k \in \mathbb{Z}_{\geq 0}$, we consider the fixed set of virtual robots $\mathcal{R}$, where some of the robots are active, and some robots are inactive (e.g., do not exist physically). We want to emphasize that tracking each robot's identity from when it is first assembled until it is reconfigured is not required. The same virtual robot might be created, disappear, and then assembled again, potentially in a different configuration. The time horizon for the robots $\mathcal{R}$ solving specification mission ϕ is denoted as $K = \|\phi\|$.

Robot and module dynamics

For tracking modular robots navigating in Env, let us define the following binary variables $z_{q,R,c,k}$, $u_{e,R,c,k} \in \mathbb{B}$, which take value one if there is a modular robot R with configuration c, at time k, at state q or edge e, respectively. The initial position of modular robots and at an initial configuration can be encoded as the following equality constraint

$$z_{q,R,c,0} = I(q = q_0, R, c), \ \forall q \in \mathcal{Q}, R \in \mathcal{R}, c \in \mathcal{C} \tag{3.9}$$

where $I(\cdot)$ is an indicator function.

Then, we need to constraint that every robot has to be in a specific configuration or it does not exist as follows

$$\sum_{c \in \mathcal{C}} z_{q,R,c,k} \leq 1, \ \forall q \in \mathcal{Q}, \ R \in \mathcal{R}, \ k \in [0 \,..\, K], \tag{3.10}$$

and is at single location if it exists,

$$\sum_{q \in \mathcal{Q}} z_{q,R,c,k} \leq 1, \ \forall c \in \mathcal{C}, \ R \in \mathcal{R}, \ k \in [0 \,..\, K] \tag{3.11}$$

Remark 3.2.1. *Note that the number of modular robots is not conserved in the environment since they can reconfigure (merge or split into a configuration with a different number of robots). Nevertheless, the number of modules is conserved in the entire mission. Therefore, we can track individual modules g from every robot R in the environment Env.*

Let us define $z_{q,g,k}$ and $u_{e,g,k} \in \mathbb{Z}_{\geq 1}$ as the number of modules with capability g at time k at state q or edge e, respectively. We can recover the number of modules from the modular robot variables as

$$z_{q,g,k} = \sum_{R \in \mathcal{R}} \sum_{c \in \mathcal{C}} \Lambda(c, g)\, z_{q,R,c,k}, \tag{3.12}$$

$$u_{e,g,k} = \sum_{R \in \mathcal{R}} \sum_{c \in \mathcal{C}} \Lambda(c, g)\, u_{e,R,c,k}, \tag{3.13}$$

for all $q \in \mathcal{Q}$, $e \in \mathcal{E}$, $g \in \mathcal{G}$, where $\Lambda(c, g)$ is the number of modules with capability g used in configuration c. Using these module capabilities variables, we can define flow dynamic constraints in the environment as follows

$$z_{q,g,k} = \sum_{(q',q) \in \mathcal{E}} u_{e,g,k}, \tag{3.14}$$

$$\sum_{(q',q) \in \mathcal{E}} u_{e,g,k} = \sum_{(q,q') \in \mathcal{E}} u_{e,g,k+W(e)}. \tag{3.15}$$

The flow constraints enforce the conservation of modules of each type while robots fly, perform tasks, and reconfigure in the environment. Note that the sums in (3.14) and (3.15) also include the reconfiguration self-loops $e^r = (q, q)^r$.

Next, we constrain the robots not to change configurations unless a reconfiguration self-loop is used

$$z_{q,R,c,k} + z_{q',R,c,k+W(e)} \geq 2\, u_{(q,q'),R,c,k}, \tag{3.16}$$

for all $e = (q, q') \in \mathcal{E} \setminus \{e^r = (q, q)^r \mid q \in \mathcal{Q}\}$, $R \in \mathcal{R}$, $c \in \mathcal{C}$, $k \in [0 \,..\, K - W(e)]$. We do not enforce the constraint for reconfiguration self-loops e^r, and let the variables free to

change robots' configurations as required by tasks' satisfaction.

Task Satisfaction

For encoding task satisfaction, let us consider the following binary variables $z_k^T \in \mathbb{B}$, which takes value one if task T is satisfied. Thus, we have

$$z_k^T \leq \sum_{R \in \mathcal{R}} \sum_{c \in \mathcal{C}} z_{q,R,c,k} \, \Gamma(c, T), \tag{3.17}$$

for all $T = (d, \pi, \mathcal{C}_T)$ with $\pi \in AP$, $q \in \mathcal{L}^{-1}(\pi)$, $k \in [0 \,..\, K]$, which ensures that modular robot R with configuration c can perform task T at state q, and at time k. Then, the satisfaction of an MTL specification is captured via a recursive encoding using binary variables as $z_k^\varphi \in \mathbb{B}$ for a subformula φ at k. The variable z_k^φ takes value one if subformula φ holds at time k and is zero otherwise. The complete encoding is similar to the one in [133, 95], and we omit it for brevity.

Objective Function

Finally, the objective functions to capture the desired optimal behavior of the modular robots.

Reconfiguration cost function We define a cost to penalize the reconfiguration of the modular robots into new configurations. The cost becomes

$$J_r = \sum_{k \in [0..K]} \sum_{q \in \mathcal{Q}} \sum_{g \in \mathcal{G}} Y_{q,g} \cdot u_{e^r,g,k},$$

where $Y_{q,g}$ is the energy for reconfiguration per module of type g at state q, $e^r = (q, q)^r$.

Action cost function We define a cost for satisfying a task T at state q using a modular robot in a configuration c as

$$J_a = \sum_{k\in[0,..K]} \sum_{q\in\mathcal{Q}} \sum_{c\in\mathcal{C}} \sum_{R\in\mathcal{R}} Y_{q,T,c} \cdot z_{aux} \cdot \Gamma(c,T),$$

where $z_{aux} = z_k^T \cdot z_{q,R,c,k}$, and $Y_{q,T,c}$ is the per time unit cost of using configuration c at state q to satisfy task T. Note that since both variables are binary, the product can be reduced to $z_{aux} = \min\{z_k^T, z_{q,R,c,k}\}$, which can be encoded as mixed integer linear constraints.

Motion cost function We capture the cost for traversing the edges e with configuration c

$$J_m = \sum_{k\in[0,..K]} \sum_{e\in\mathcal{E}'} \sum_{c\in\mathcal{C}} \sum_{R\in\mathcal{R}} Y_{e,c} \cdot u_{e,R,c,k}, \tag{3.18}$$

where $\mathcal{E}' = \mathcal{E} \setminus \{(q,q),(q,q)^r \mid q \in \mathcal{Q}\}$, and $Y_{e,c}$ is the per time unit cost of using configuration c while traversing edge e.

Optimization problem Then, we can formulate Problem 3.2.1 as the following MILP optimization problem

$$\min_{u_{e,R,c,k}} J_r + J_a + J_m, \text{ s.t. } (3.9) - (3.16), (3.17), z_0^\phi = 1.$$

3.2.4 Case Studies

In this section, we perform multiple simulations to demonstrate the performance of the MILP formulation in terms of scalability and correctness. We are mainly motivated by case studies where we need to specify time and logically constrained tasks that describe a construction mission in agriculture environments. All computations of the following case studies were performed on a PC with four cores at 2.7 GHz with 32 GB of RAM. We used

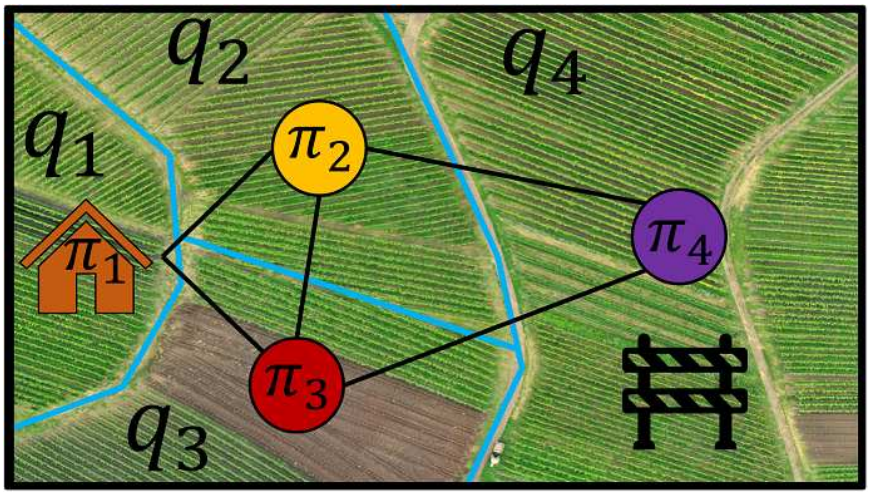


Figure 3.2.6: Case study 1

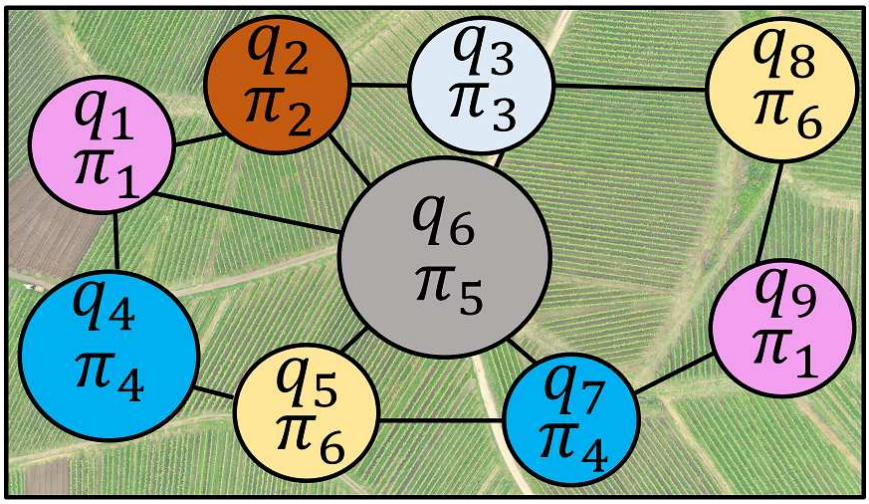


Figure 3.2.7: Case study 2

Gurobi [66] as the solver.

Case Study 1

Here, we describe a small construction mission for showcasing how the MILP planning algorithm encoded in Sec. 3.2.3 works. We consider and agricultural environment shown in Fig. 3.2.6 with four different states $\mathcal{Q} = \{q_1, q_2, q_3, q_4\}$ and atomic proposition set $\Pi = \{\pi_1, \pi_2, \pi_3, \pi_4\}$. The mission considers that at some time between deployment, wood must be cut at state q_4, and then the wood needs to be used to assemble a fence. The MTL mission specification is $\phi = \Diamond_{[0,3]} T(1, \pi_{purple}, g_3) \wedge \Diamond_{[4,6]}(T(1, \pi_{purple}, g_2) \wedge T(1, \pi_{purple}, g_1))$. We consider the following initial conditions, module capabilities set $\mathcal{G} = \{g_0, g_1, g_2, g_3\}$ whose meaning is shown in Fig. 3.2.3. We have $|\mathcal{C}| = 10$ predefined configurations $\Lambda(c, g) = \{c_0 : \{g_0 = 4, g_1 = 1, g_2 = 1\}, c_1 : \{g_0 = 8, g_1 = 1, g_2 = 1\}, c_2 : \{g_0 = 3, g_2 = 1\}, c_3 : \{g_0 = 9, g_2 = 1\}, c_4 : \{g_0 = 4, g_1 = 1\}, c_5 : \{g_0 = 7, g_2 = 1, g_3 = 1\}, c_6 : \{g_2 = 1\}, c_7 : \{g_1 = 1\}, c_8 : \{g_3 = 1\}, c_9 : \{g_0 = 1\}\}$. We consider twelve g_0 modules with flying capability and two of each g_1, g_2, g_3 at headquarters q_1. We constraint the maximum number of robots we can generate to be $|\mathcal{R}| = 13$. The binary matrix indicating whether or not a configuration can manipulate a tool $\Gamma(c, T)$ has value one if c contains g_τ (capability needed in the task T) and at least three g_0 and is zero otherwise. Motion, reconfiguration, and task energy consumption for using a specific configuration are generated so that $Y_{q,g}, Y_{qtc}$, and $Y_{e,c} \in [1 \ .. \ 10]$. Finally, the reconfiguration time $W(e^r)$ for going from any configuration to another feasible configuration is first considered as one. We use these initial parameters for four different cases.

Feasible Case The generated solution shows that there are six initial robots at time $k = 0$ at state q_1, robots are in the following initial configurations $R_0 : c_8$, $R_1 : c_4$, $R_2 : c_8$, $R_3 : c_1$, $R_4 : c_6$. Note that the rest of the robots do not appear since not all the robots must exist at every time; they may reconfigure, merge, and split as required. Sat-

Table 3.1: Case Study 2: Runtime and complexity performance, integer and binary variables while increasing the number of modules and module capabilities.

n	Modules	$\lvert\mathcal{C}\rvert$	**time**	**Int. Vars**	**Binary Vars**
1	30	5	46.26s	23799	18088
2	45	10	69.93s	25234	19391
3	60	15	168.52s	29321	20892
4	80	20	255.63s	32321	21865
5	100	25	451.31s	33321	25324

isfaction of the subformula $\varphi_1 = \Diamond_{[0,3]} T(1, \pi_{purple}, g_3)$ is achieved at time $k = 2$ where the robots have reconfigured and traveled as follows at state q_1. We have robots with the following configurations $R_5 : c_9$, $R_6 : c_6$, $R_9 : c_9$, $R_{10} : c_7$, $R_{11} : c_8$, $R_{12} : c_9$, at state q_4 time $k = 2$. There is one robot with configuration $R_4 : c_5$. The type of modules in configuration c_5 shows that the subformula φ_1 has been satisfied. Satisfaction of subformula $\varphi_2 = \Diamond_{[4,6]}(T(1, \pi_{purple}, g_2) \wedge T(1, \pi_{purple}, g_1))$ is achieved at time $k = 6$. The robots at state q_1 are the following $R_0 : c_4$, $R_7 : c_8$, $R_{11} : c_6$, at state q_4 robots $R_0 : c_9$, $R_1 : c_2$, $R_5 : c_8$, $R_{12} : c_4$. Here we can see that robots $R_1 : c_2$ and $R_{12} : c_4$ satisfy the subformula. The computation time for solving this case study is 1.30s. The objective values are reconfiguration cost $J_r = 4$, action cost $J_a = 6$, motion cost $J_m = 45$.

Infeasible Case Here, we change the reconfiguration time $W(e^r) = 3$, and the model becomes infeasible since robots need to enter into reconfiguration to satisfy tasks. Still, it causes a violation of the deadlines of the specification.

Feasible case with additional modules, configuration, and possible robots Here, we add twelve additional quadrotors and increase the set of configurations $|\mathcal{C}| = 20$, and robots to $|\mathcal{R}| = 23$. Initially, there are so many robots and configurations that reconfigurations are not necessary, and the chosen robots are the ones that reduce the motion and task cost. All changes are at the expense of increasing the computation since every time we increase the set of possible configurations, $\mathcal{C}$, and the set of possible robots $\mathcal{R}$, we include additional

binary variables. The solution took 5.86 seconds, and specification satisfaction $z_0^{\phi} = 1$ is achieved while the other objective values are reconfiguration cost $J_r = 0$, action cost $J_a = 4$, and motion cost $J_m = 29$.

Feasible with an increase in the number of states and edges Lastly, we augment the environment by adding six states $\mathcal{Q}$ and generating edges $\mathcal{E}$ according to proximity. The transition system is shown in Fig. 3.2.7. Here, the computation time increases to 27.06 seconds The specification is satisfied, $z_0^{\phi} = 1$, with reconfiguration cost $J_r = 0$, action cost $J_a = 4$, and motion cost $J_m = 33$. Additionally, there is a drastic increase in the number of generated variables, which shows that the algorithm is susceptible to the number of states and edges in terms of time and number of variables. This is not surprising since for every state and edge in Env, all the variables for modules and robots have to be created.

Case Study 2

We consider the environment shown in Fig. 3.2.7. The predefined configurations and number of robots are $|\mathcal{C}| = 20$ and $|\mathcal{R}| = 20$, and the mission specification $\phi_n = \wedge_{i=1}^{n}\big(\wedge_{j=1}^{3} \Diamond_{[0,20]} T(2, \pi_j, g_{\tau,j})\big)$, where π_j is a randomly picked state in the environment and $g_{\tau,j}$ the first three capabilities added every time that n increases. Table 3.1 summarizes the information at every iteration on how the runtime and number of variables increase when adding more modules and module capabilities.

Finally, we keep all other parameters constant and gradually increase the number of possible robots and configurations. All initial parameters are the same as the one considered for 3.2.4. In Fig. 3.2.8, we can see that, at first, increasing the number of variables does not affect the time performance dramatically. However, after fifteen configurations and fifteen possible robots, the combinatorial problem makes the runtime grow almost exponentially.

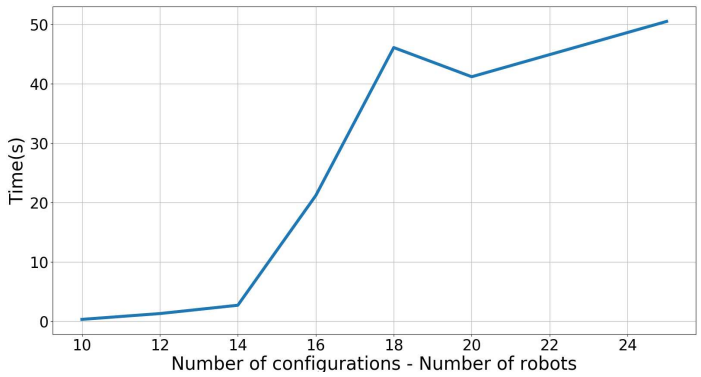


Figure 3.2.8: Runtime performance while increasing the number of possible configurations and robots.

3.2.5 Conclusions

This section proposes a MILP formulation for planning for heterogeneous modular robots that can form configurations to manipulate tools and satisfy tasks in an environment. We consider that not all configurations can satisfy every task, which creates the necessity for modular robots to reconfigure. We formulate tasks to account for logical and timing constraints using Metric Temporal Logic, where atomic propositions capture the location and the tool the task requires. We formulate costs for motion, reconfiguration, and actions for satisfaction in a specific configuration. We conclude that the time performance depends directly on the number of binary variables, which depend on all the possible configurations and robots that the system can generate, the size of the environment, and the specification size.

3.3 Planning for Heterogeneous Teams of Robots with Temporal Logic, Capability, and Resource Constraints

This section presents a comprehensive approach for planning for teams of heterogeneous robots with different capabilities and the transportation of resources. We use Capability Temporal Logic (CaTL), a formal language that helps express tasks involving robots with multiple capabilities with spatial, temporal, and logical constraints. We extend CaTL to also capture resource constraints, where resources can be divisible and indivisible, for instance, sand and bricks, respectively. Robots transport resources using various storage types, such as uniform (shared storage among resources) and compartmental (individual storage per resource). Robots' resource transportation capacity is defined based on resource type and robot class. Robot and resource dynamics and the CaTL mission are jointly encoded in a Mixed Integer Linear Programming (MILP), which maximizes disjoint robot and resource robustness while minimizing spurious movement of both. We propose a multi-robustness approach for Multi-Class Signal Temporal Logic (mcSTL), allowing for generalized quantitative semantics across multiple predicate classes. Thus, we compute availability robustness scores for robots and resources separately. Finally, we conduct multiple experiments demonstrating functionality and time performance by varying resources and storage types.

3.3.1 Introduction

Advancements in multi-robot platforms have enabled new applications that use multiple robots with different capabilities. Such applications include aerial surveillance, disaster response, and planetary exploration, which could involve aerial and ground robots working together [48, 157, 17, 23, 28]. Having a team of heterogeneous robots can be very useful when it comes to satisfying tasks. By using their capabilities, they can explore different solutions and make the mission more resilient and robust. However, finding the best so-

lution becomes difficult when the number of robots or classes increases. This is because considering all the different combinations and possibilities can be very complex and time-consuming. In fact, many existing planning algorithms become impractical or less effective when dealing with many robots and complex mission specifications.

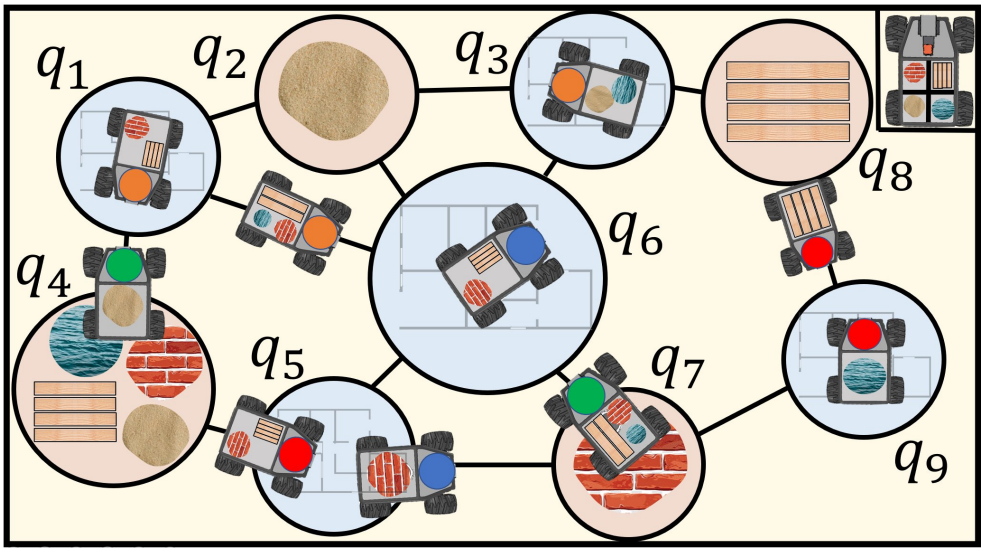


Figure 3.3.1: Schematic of a construction motion coordination problem. Connected circles correspond to construction areas (light blue) or warehouse storage (light brown). There are four resources, two indivisible (bricks and wooden beams) and two divisible (construction sand and water). The circle color on the robots indicates the agent class (set of capabilities each robot has). Uniform storage type is shown in the robots; however, a robot with compartmental storage capacities can be seen in the top-left corner.

Temporal logic has become increasingly popular for specifying tasks when synthesizing plans for large homogeneous and heterogeneous teams [63, 62, 79, 47, 122, 29, 27]. This tool is especially useful for expressing spatial and temporal requirements for complex missions, such as the type of robots needed in a particular area at a specific time. While Linear Temporal Logic (LTL) [13, 160, 80] is a commonly used temporal logic formalism in robotics, it only reasons over untimed sequences (e.g., "Visit region A before going to region B"), which may not be suitable for time-sensitive missions. To accommodate explicit timing constraints, various other temporal logics have been proposed, such as Signal Temporal Logic (STL) [108], weighted-STL (wSTL) [111, 24], Time Window Temporal Logic (TWTL) [162], Bounded Linear Temporal Logic (BLTL) [155], and Metric Tempo-

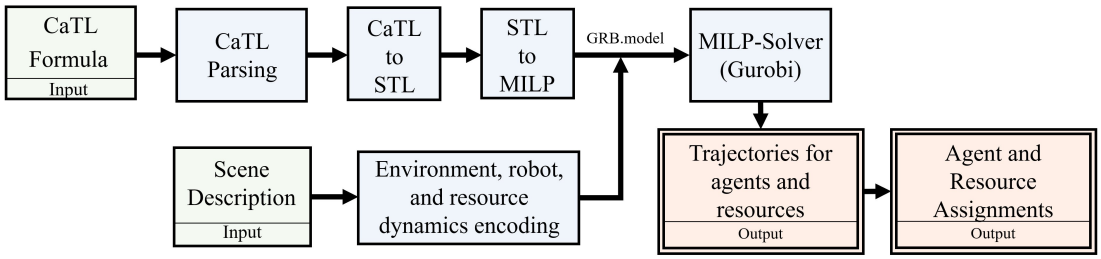


Figure 3.3.2: Schematic overview of CaTL with resource constraints.

ral Logic (MTL) [18].

Among the formalisms that capture explicit time, STL specifications have additional advantages by considering predicates that allow reasoning about signals with relation operators instead of atomic propositions. For instance, it can express whether a car's speed limit should remain within 70 miles per hour by expressing the predicate "$speed \leq 70$". It also can be applied to discrete-valued signals, such as indicating the number of robots per class required for heterogeneous multi-robot frameworks. Additionally, unlike LTL or MTL, which only provide feedback on task satisfaction, STL offers a margin of satisfaction, also known as *robustness* [57, 51], which indicates the extent to which the mission is satisfied or violated. For example, if a car maintains a speed of 65 miles per hour when the speed limit is 70 miles per hour, the mission is considered satisfied with a robustness of 5. Nevertheless, even when considering a complex STL specification or a multi-agent approach [150, 110, 29], robustness will be a single value computed by a recursive definition over the STL operators. The single robustness value lacks interpretability when considering an STL specification using different predicate classes, i.e., semantically different variables, e.g., reasoning about agents and resources. Instead, we propose a multi-robustness for Multi-Class Signal Temporal Logic (mcSTL), which generalizes the quantitative semantics for STL specifications with different predicate classes to compute disjoint robustness scores.

This section builds upon [75, 95], which introduces Capability Temporal Logic (CaTL), an STL fragment allowing a team of robots with varying sets of capabilities, to satisfy different requests. We extend CaTL to also account for resources required to perform

tasks. Tasks specify the type of robot capabilities and resources, the robot and resource quantities, the specific location in an environment, and the time robots need to be there. In the case of resources, they disappear once the satisfaction of the task starts. Satisfaction of a task is robot agnostic, i.e., there is no explicit allocation of a specific agent to the task; instead, any robot or team of robots with the required capabilities can complete the task. The resources are transported by robots with different storage types and capacities, such as uniform (i.e., regardless of the resource, all share the same space and robot's storage capacity) and compartmental (i.e., each resource has its own dedicated space and storage capacity). Agent and resource assignments and trajectories are created after computing a solution for agents' and resource flows. Additionally, multiple types of resources are considered, such as indivisible (e.g., bricks) or divisible (e.g., sand).

General Overview of the Approach

Here, we provide an overview of our approach to addressing route planning challenges in heterogeneous robots and resource transportation teams. Our method involves utilizing Capability Temporal Logic (CaTL [75, 95]), a fragment of STL, to specify mission objectives. We have extended the semantics of CaTL[3] to capture multiple resource transportation modes and capacities.

Our model incorporates the dynamics of agents and resources and formulates an optimal route planning problem under CaTL specification using a Mixed Integer Linear Programming (MILP) approach. Fig. 3.3.2 provides a schematic overview of the problem, where only the CaTL formula and the scene description are needed as input. The CaTL formula describes the desired tasks, including their duration, the number of agent capabilities required, and the amount of resources needed to complete the task. Logical and temporal constraints can also be included. For the scene description, the user specifies the

[3]We decided to retain the name CaTL name of the specification language that now considers resources required for tasks and not just agent capabilities as in [95]

environment as a transition system, indicating the states, transitions, durations, and initial distribution of agents and resources.

The CaTL formula is parsed, and its associated Abstract Syntax Tree (AST) [71] is constructed using an LL(*) parser [124]. We then translate the CaTL formula into an equivalent STL formula in the AST representation (refer to PyTeLo Cardona et al. (2023b) for more details). Using PyTeLo, we recursively translate the STL specification AST into a MILP, capturing the quantitative semantics of the specification. The environment, robot, and resource dynamics encoding are automatically generated as a MILP using the formulation in Sec. 3.3.7. Then, we utilize a MILP solver such as Gurobi [66] to solve the equivalent MILP, capturing the specification and dynamics and generating the solution as trajectories for agents and resources in the environment. The solution to the MILP provides trajectories for the flows of agents and resources in the environment. However, these trajectories do not assign specific robots to tasks or resources to be transported, as they are agent-agnostic. Instead, they rely on the number of agents per class and the amount of resources needed for satisfaction. Therefore, the final step is allocating agents and resources to flows (i.e., trips) according to the solution provided by the MILP. The overall result consists of trajectories for all agents and resources that they need to pick-up, transport, and drop-off over the mission horizon.

The main contributions of this section are

1. Extending CaTL Leahy et al. (2021) to include tasks that require both agents and resources.

2. Defining and modeling resource transportation that can accommodate various types of resources (divisible, indivisible, packets, etc.) and considers different storage types (uniform, compartmental) and capacities for agents.

3. Defining Multi-Class Temporal Logic (mcSTL), an extension of STL that considers predicates from multiple semantic classes. We also define a multi-robustness score

that enables quantification of satisfaction for each predicate class.

4. Applying the multi-robustness mcSTL in the context of CaTL to independently compute the robustness of robots and resources.

5. Proposing an efficient MILP-based planning approach that captures the CaTL specification, robots and resources dynamics, and transportation constraints. The MILP aims to maximize disjoint robots' and resources' robustness while minimizing spurious motion.

6. Evaluating the performance of the planning approach on construction scenarios to gain insights into the problem's characteristics and practical application.

3.3.2 Literature Review

Temporal logics have witnessed success in the domain of high-level planning for robotics, from single-agent systems [6, 10] and, increasingly, in multi-agent systems [62, 64, 47, 122], including scenarios involving heterogeneous teams [142]. One of the most predominant temporal logic formalisms is Linear Temporal Logic which traditionally embraced automata theory as the primary paradigm [37, 93, 6, 10, 58, 78, 5, 159, 118]. However, applying automata-based methodologies in multi-robot scenarios introduces substantial challenges related to scalability and computational complexity, primarily owing to the requisite computation of automata products. Other works have considered a decomposition of tasks to allocate to single agents so that, by the composition of all agents, the satisfaction of the mission is guaranteed [158, 176]. Also, there is promising work on avoiding the need for automata products by incorporating fixed Petri nets approaches [72, 92, 85, 105]. Nonetheless, handling large numbers of agents is still a challenge since the problem and its variants are NP-hard.

To alleviate these challenges, recent approaches have shifted their focus towards Mixed-Integer Linear Programs (MILP), benefitting from advancements in solution techniques and

leveraging optimized off-the-shelf tools like Gurobi [66]. This transition has proven effective, enabling MILP to handle a significantly larger number of variables and constraints with reasonable computational resources. Existing works adopting MILP strategies for multi-robot systems planning, particularly when subject to Signal Temporal Logic (STL) specifications, are evident in the literature [150, 30, 21, 104, 20, 177, 146, 32], one of their main advantage is that they are complete, i.e., if there exists a solution it will be found. Compared to Linear Temporal Logic (LTL), Signal Temporal Logic (STL) allows for the explicit definition of time, which enhances the expressiveness of the mission by enabling the inclusion of timing constraints. However, existing STL methods focus on computing a solution by maximizing a single robustness score, even when predicates may belong to semantically different classes. In contrast, our research proposes a new approach by using a multi-robustness strategy for multi-class Signal Temporal Logic (mcSTL). This unique method allows us to formulate a multi-predicate class within the CaTL language, where the robustness of both resources and robots is maximized simultaneously but independently.

In high-level planning for multi-robot systems, besides the mission specification, the abstraction of the environment is crucial for handling the complexity of the problem. Allocating heterogeneous agents and resources to tasks with temporal logic constraints is a difficult problem known to be NP-hard. Previous works have explored the use of Petri nets and graph-based models to tackle this problem [72, 92, 85, 105]. Other works have considered occupancy grids [151, 14], cell decomposition maps [11, 41], or Voronoi tessellation approaches [119, 60]. However, our approach incorporates dynamics efficiently by imposing flow dynamic constraints over a transition system and strategically abstracting agents and resources. We define the joint state of the team as the count of agents with each capability and resource in each region at a given time. This abstraction model, combined with the use of the robot and resource routing flows problem approach, enhances the efficiency and scalability of our approach.

Some of the works closely related to our approach are [136, 135], which have incor-

porated censusSTL [168] into (cLTL+) to enable agent or capability counting in abstracted time specifications. However, these works have not accounted for tasks involving robots and resource constraints. Related works considering capabilities and resource constraints in heterogeneous multi-robot systems are [142, 65]. More specifically, in [65], authors consider agents capable of gathering data, receiving, and uploading via buffers, defining in this way a resource transportation. Still, both use LTL language to express mission specifications, lacking the expressivity to account for time explicitly. The concept of resources is presented dually in the paper [142]. Firstly, it involves incorporating a proposition in LTL format, which is expressed as $(\varsigma > \mu)$, where ς denotes a specific resource, such as the battery power of a robot, while μ sets the threshold. Secondly, regions in the environment are labeled as either true or false, and their state is updated based on the actions taken in an automaton that captures resource constraints. In contrast, we expand on the notion of resource planning by considering divisible and indivisible types. Also, robots might have different storage types (compartmental and uniform) and adjustable storage capacity. Our proposed planning model does not include unnecessary binary variables for routing the resources while robots move in the environment.

3.3.3 Problem Formulation

Here, we define the route planning problem for heterogeneous teams of robots performing missions specified as Capability Temporal Logic formulae (CaTL) [75, 95]. We extend CaTL to account for the capabilities and resources required to fulfill tasks. Additionally, we extend the agent model with various resource transportation modalities and capacities. Finally, we conclude the section with the optimal route planning problem under capability temporal logic task and resource constraints.

The following example motivates the problem class we consider in this section, where resources are required at locations of interest in the environment to perform tasks by agents with heterogeneous task and transportation capabilities.

Example 3.3.1. *Consider an environment with areas q_1, q_3, q_5, q_6, q_9 under construction and warehouses at q_2, q_4, q_7, q_8. A fleet of ground-based robots with different capabilities, such as bricklaying, digging, sample extraction, video surveillance, and structure inspection, is deployed to assist in the construction areas. Each robot has a combination of task capabilities defining a robot class (e.g., a robot can have the capabilities of bricklaying and digging) and transportation. Each robot class has a storage capacity to transport resources such as sand, water, bricks, and wooden beams from warehouses to construction areas. As usual in a construction area, not all capabilities or resources are needed simultaneously and change daily depending on the progress of construction. An example of a list of tasks the robots need to perform during a workday is given in List 1.*

LIST 1 Example list of construction tasks requiring capabilities and resources.

1. From deployment to the end of the day, one video surveillance is required at every construction area.
2. Within 1 to 6 hours after deployment, digging is required for 2 hours and 2 wooden beams in every construction area.
3. Within 4 and 12 hours after deployment, structure inspection is required in every construction area for 1 hour. Afterward, 10 kg of sand and 10 liters of water are required for foundation-laying tasks.
4. Within 12 hours after deployment to the end of the day, 2 bricklaying and 200 bricks are required for 6 hours at every construction area.

Based on this example, we introduce the environment, capabilities, resources, and robot models.

Environment, Agent, and Resource Models

Consider a team of heterogeneous agents deployed in a bounded environment $Env = (\mathcal{Q}, \mathcal{E}, \mathcal{W})$, where $\mathcal{Q}$ is a finite set of locations of interest (states), $\mathcal{E} \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set capturing the possible transitions between locations, and $\mathcal{W} : \mathcal{E} \to \mathbb{Z}_{\geq 1}$ maps each transition

to its travel duration. We assume that time is discretized, and all weights $\mathcal{W}(\mathcal{E})$ are integer multiples of a discretization time $\Delta t > 0$. States are labeled with atomic propositions from a set $\mathcal{AP}$. The labeling function is denoted by $L : \mathcal{Q} \to 2^{\mathcal{AP}}$. An agent stationary at $q \in \mathcal{Q}$ is modeled as a unit-weight self-transition, i.e., $(q, q) \in \mathcal{E}$ for all $q \in \mathcal{Q}$, and $\mathcal{W}((q, q)) = 1$.

We consider heterogeneous agents with varying capabilities for performing tasks and transporting resources. The finite set of agents is denoted by $\mathcal{J}$ with tasks capabilities from the finite set $\mathcal{C}$. Continuing Example 3.3.1, the capabilities of a robot may include bricklaying, digging, sample extraction, video surveillance, and structure inspection using ultrasound sensors. The capability set of agent $j \in \mathcal{J}$ is $c_j \subseteq \mathcal{C}$, and defines the *agent's class* $g = c_j$. The set of all agent classes is $\mathcal{G} \subseteq 2^{\mathcal{C}}$.

We also consider that each agent is capable of transporting resources. Consider the finite set of resources $\mathcal{H}$. Resources can be divisible (e.g., sand, water, fuel) or indivisible (e.g., bricks, wooden beams, solar panels). We consider multiple resource storage modes for transportation, *uniform storage* and *compartmental storage*. In the uniform storage case, agents' storage can hold all resource types and is characterized by a maximum transportation capacity $\Omega_g > 0$ for each agent class $g \in \mathcal{G}$. For example, the trunk of a car can hold various resources such as bags, tools, and construction materials. In the compartmental storage case, each resource $h \in \mathcal{H}$ has its own compartment with a given maximum transportation capacity $\Omega_{h,g} \geq 0$ in the agents' storage, $g \in \mathcal{G}$. In this case, we denote the set of agent classes that can transport resource $h \in \mathcal{H}$ by $\mathcal{G}_h = \{g \in \mathcal{G} \mid \Omega_{h,g} > 0\}$. For example, the reservoir of a car can only hold fuel; Robots for science missions have limited unique storage slots for samples collected from the environment.

Remark 3.3.1. *The proposed modeling framework and solution can accommodate a mixture of storage modes. For simplicity, we only consider the extreme cases of uniform (single storage for all resources), and compartmental (specialized storage for each resource) storage in the problem formulation and solution description. In Sec. 3.3.8, we show how to generalize and adapt the algorithms for custom storage configurations. Moreover, we*

consider storage defined by agent classes for performance reasons. However, we can accommodate cases of agents with the same capability sets but distinct storage, as discussed in Sec. 3.3.8.

Each agent, $j \in \mathcal{J}$, is characterized by its initial state $q_{0,j} \in \mathcal{Q}$, and capability set c_j gives its class. The trajectory of an agent j in environment Env is defined as $s_j : \mathbb{Z}_{\geq 0} \to \mathcal{Q} \cup \mathcal{E}$ such that $s_j(t)$ is the state occupied or transition traversed by agent j at time $t \in \mathbb{Z}_{\geq 0}$, and every agent starts at the initial state $s_j(0) = q_{0,j}$. The synchronous trajectory of a set of agents $\mathcal{J}$ is $s_{\mathcal{J}} : \mathbb{Z}_{\geq 0} \to (\mathcal{Q} \cup \mathcal{E})^{|\mathcal{J}|}$. The number of agents at state $q \in \mathcal{Q}$ with capability $c \in \mathcal{C}$ at time $t \in \mathbb{Z}_{\geq 0}$ is

$$n_{q,c}(t) = |\{j \in \mathcal{J} \mid q = s_j(t), c = c_j\}|.$$

Similarly, the trajectory (distribution) of resource $h \in \mathcal{H}$ over the environment is $b_h : \mathbb{Z}_{\geq 0} \times (\mathcal{Q} \cup \mathcal{E}) \to \mathbb{R}_{\geq 0}$. Note that for indivisible resources $b_h \in \mathbb{Z}_{\geq 0}$. The initial distribution of resources, $b_h(0, \mathcal{Q})$ for all $h \in \mathcal{H}$, is given, and we assume that all resources are at states, i.e., $b_h(0, e) = 0$ for all $e \in \mathcal{E}$. The trajectory of all resources is denoted by $b_{\mathcal{H}}$.

For simplicity, we enforce that agents pick up, drop off, and transfer resources between themselves only at states $q \in \mathcal{Q}$. A *transportation plan* for agents $\mathcal{J}$ with respect to their trajectory $s_{\mathcal{J}}$ is a set of assignments $m_j : \mathbb{Z}_{\geq 0} \times \mathcal{H} \to \mathbb{R}_{\geq 0}$ that define the quantity of resource $h \in \mathcal{H}$ associated with agent $j \in \mathcal{J}$ at time $t \in \mathbb{Z}_{\geq 0}$. We impose $m_j(t, h) = 0$ for all $h \in \mathcal{H}$ whenever $s_j(t) \in \mathcal{Q}$ which captures pick-up, drop-off, and transfer of resources in a unified way, and may not correspond to physical operations (e.g., a robot that passes through a state $q \in \mathcal{Q}$ does not need to drop off and pick up the resources it is carrying). A transportation plan is *feasible* if the transportation capacity of agent j is satisfied at all times,

$$m_j(t, h) \leq \Omega_{h,g}, \forall h \in \mathcal{H}, \ \forall t \geq 0 \text{ (compartmental)},$$

$$\sum_{h \in \mathcal{H}} m_j(t, h) \leq \Omega_g, \ \forall t \geq 0 \text{ (uniform)}.$$

Now that we have introduced the environment, agents, capabilities, and resources models, we are ready to introduce CaTL with task and resource constraints.

Capability Temporal Logic with Resource Constraints

We extend the Capability Temporal Logic (CaTL) [4] Specification language from [75] to account for resource constraints when performing tasks. The core units of CaTL are *tasks* that capture the required number of agents with each capability required in a location of interest. We extend the task definition to specify required resource quantities as follows

Definition 3.3.1. *A* task *is a tuple* $T = (d, \pi, cp, rs)$*, where* $d \in \mathbb{Z}_{\geq 1}$ *is a discrete duration of time (i.e., multiple time steps* Δt*),* $\pi \in \mathcal{AP}$ *is an atomic proposition,* $cp : \mathcal{C} \to \mathbb{Z}_{\geq 0}$ *and* $rs : \mathcal{H} \to \mathbb{R}_{\geq 0}$ *are counting maps specifying how many agents with each capability and how many resources of each type should be in each region labeled* π*, respectively. Note that for divisible resources* $rs(h) \in \mathbb{R}_{\geq 0}$*, while for indivisible ones* $rs(h) \in \mathbb{Z}_{\geq 0}$*.*

A capability c not required to perform task T is defined by $cp(c) = 0$. Similarly, $rs(h) = 0$ indicates that h is not required to perform T. We denote the set of capabilities and resources necessary for a task T by $cp_T = \{c \in Cap \mid cp(c) > 0\}$, and $rs_T = \{h \in \mathcal{H} \mid rs(h) > 0\}$, respectively.

Definition 3.3.2. *The* syntax *of CaTL [75] is a fragment of STL described in* (2.2)*. The CaTL syntax in Backus-Naur form is*

$$\phi ::= T \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \mathcal{U}_I \phi_2 \mid \Diamond_I \phi \mid \Box_I \phi$$

where ϕ *is a CaTL formula,* T *is a* task containing capability and resource constraints*,* $\wedge$ *and* $\vee$ *are the Boolean conjunction and disjunction operators,* $\mathcal{U}_I$*,* $\Diamond_I$*, and* $\Box_I$ *are the time-bounded until, eventually, and always operators, respectively.*

[4] We have decided not to introduce a new name for the extension of CaTL with resources. Instead, throughout the paper, when we refer to CaTL, we mean the version defined in this section.

Example 3.3.2 ((Continuation of Example 3.3.1)). *Thus, the mission described in List 1 can now be expressed as*

$$\phi = \Box_{[0,24]} T_1 \wedge \Diamond_{[1,6]} T_2 \wedge \Diamond_{[4,12]} T_3 \wedge \Diamond_{[12,24]} T_4,$$

with the set of tasks defined as follows

$$\begin{aligned} T_1 =& (1, \pi_{cons}, \{(Struct.Insp., 1)\}, \varnothing), \\ T_2 =& (2, \pi_{cons}, \{(Digging, 1)\}, \{(woodbeams, 2)\}), \\ T_3 =& (1, \pi_{cons}, \{(Struct.Insp., 1)\}, \{(sand, 10), (water, 10)\}), \\ T_4 =& (6, \pi_{cons}, \{(bricklaying, 2)\}, \{(bricks, 200)\}), \end{aligned}$$

where $\pi_{cons} \in \mathcal{AP}$, captures all regions under construction $L^{-1}(\pi_{cons}) = q_1, q_3, q_5, q_6, q_9$.

Remark 3.3.2. *As in [75], CaTL does not include negation since the negation of tasks does not have a well-defined meaning. This is feasible because any STL formula can be put in its positive normal form [133], and CaTL is a fragment of STL.*

Definition 3.3.3. *The Boolean (qualitative) semantics of CaTL are defined over trajectories $s_{\mathcal{J}}$ of agents and $b_{\mathcal{H}}$ of resources. At time t, satisfaction of a task T is defined by*

$$\begin{aligned} (s_{\mathcal{J}}, b_{\mathcal{H}}, t) \vDash T \Leftrightarrow \forall \tau \in [t \,..\, t+d], \forall q \in L^{-1}(\pi), \forall c \in cp_T, \forall h \in rs_T, \\ n_{q,c}(\tau) \geq cp(c) \wedge b_h(t, q) \geq rs(h) \,. \end{aligned}$$

The Boolean and temporal operators' semantics are the same as for STL, (2.3). *A pair of team and resource trajectories satisfy a CaTL formula ϕ, denoted $(s_{\mathcal{J}}, b_{\mathcal{H}}, t) \vDash \phi$ if $(s_{\mathcal{J}}, b_{\mathcal{H}}, 0) \vDash \phi$.*

Tasks require resources only at the time of satisfaction, unlike agents required throughout their duration. Resources associated with a task T are consumed when T is satisfied

and disappear from the environment in the next time step in the amount $rs(h)$ required by T, for all $h \in rs_T$. However, we must ensure that concurrent tasks using the same resource type at overlapping locations do not exceed the total resources available at these locations. In such a case, not all tasks requiring the same resource can be satisfied simultaneously. We refer to this property as the *cross-consumption constraint*.

Next, we review the *agent availability robustness* from [75], and propose a counterpart for resources.

Definition 3.3.4 (Agent Availability Robustness [75])**.** *The agent availability robustness of a task is defined as*

$$\rho_a(s_{\mathcal{J}}, T, t) = \min_{c \in cp_T} \min_{t' \in [t..t+d]} \min_{q \in L^{-1}(\pi)} n_{q,c}(t') - cp(c). \tag{3.19}$$

For Boolean and temporal operators, robustness is defined recursively as for STL, (2.4)*.*

The agent availability robustness of a task captures the deviation from the number of agents needed to perform the task over its duration, capabilities involved, and locations spanned. Positive values indicate the maximum number of arbitrary agents whose failure leads to failing the task. We compute ρ_a based only on the agents' trajectory $s_{\mathcal{J}}$.

We propose *resource availability* defined similarly.

Definition 3.3.5 (Resource Availability Robustness)**.** *The resource availability robustness of a task is defined as*

$$\rho_h(b_{\mathcal{H}}, T, t) = \min_{h \in rs_T} \min_{q \in L^{-1}(\pi)} b_h(t, q) - rs(h). \tag{3.20}$$

Again, for Boolean and temporal operators, robustness is defined recursively as for STL, (2.4)*.*

Resource robustness captures the surplus and shortage of resources required to satisfy a task and is computed only from the resource trajectory $b_{\mathcal{H}}$.

Problem Statement

Before we state the problem, let us explain how the agents' trajectories with resource trajectories are linked. We say that a resource trajectory $b_{\mathcal{H}}$ is consistent with agents' trajectory $s_{\mathcal{J}}$ if a feasible transportation plan exists for all agents $j \in \mathcal{J}$ with respect to $s_{\mathcal{J}}$ that induces $b_{\mathcal{H}}$. Specifically, resource quantities at states and transition at each time step result from agents carrying them via some transportation plan. We can now formally introduce the problem as follows.

Problem 3.3.1. *Given a set of agents $\mathcal{J}$ with initial states $q_{0,j}$, capabilities $c_j \subseteq \mathcal{C}$ deployed in environment Env with initial resources distribution $b_{\mathcal{H}}(0, \mathcal{Q})$, and CaTL specification ϕ, find trajectories s_j and b_h for agents and resources such that $(s_{\mathcal{J}}, b_{\mathcal{H}}) \vDash \phi$, $b_{\mathcal{H}}$ is consistent with $s_{\mathcal{J}}$, $b_{\mathcal{H}}$ satisfies the cross-consumption constraint at all times, and the robustness score*

$$F = \rho_a(s_{\mathcal{J}}, \phi, 0) + \gamma \cdot \rho_h(b_{\mathcal{H}}, \phi, 0), \tag{3.21}$$

is maximized, where $\gamma \in \mathbb{R}_{>0}$ is a positive constant value.

Pb. 3.3.1 captures the goal of using a fleet of heterogeneous robots (agent classes defined by the set of capabilities) to satisfy CaTL tasks considering capabilities, resources, and logical and temporal constraints. Note that instead of routing resources and enforcing correspondence between agents and resources, we want to exploit the constraint that the transportation plan (flows of resources) has to follow agents' trajectory (flows of agents). Thus, we avoid introducing complex agent-resource assignment constraints. The robustness score F corresponds to finding a plan that maximizes the number of robots with specific capabilities and the number of resources available at locations asked in a specification.

Pb. 3.3.1 is solved by encoding the models as a Mixed Integer Linear Program (MILP). This allows us to use off-the-shelf solvers that are orders of magnitude faster than standard automata-based approaches. We show the encodings in Sec. 3.3.7.

Note that for STL specifications, a positive robustness guarantees Boolean satisfaction (Soundness, Thm. 2.1.1). However, here we work with multiple robustness variables, such as agent and resource availability, which means that a positive result to the weighted addition of ρ_a and ρ_h in (3.21) does not guarantee the satisfaction of a CaTL specification formula (an example and formal results are presented in Sec. 3.3.9). Furthermore, in the syntax of STL presented in Sec. 2.1, there is no distinction when computing the robustness of the STL specification formula that uses multiple predicate classes. To address this issue, in the next section, we introduce a semantic extension of STL called Multi-class STL (mcSTL). mcSTL can handle multiple predicate classes and compute their respective robustness scores.

3.3.4 Multi-robustness for STL Specifications with Disjoint Predicate Classes

The standard robustness score (2.4) for STL specifications considers that all predicates ($\varsigma = s_i \sim \mu$) are the same, belong to the same class [133, 91, 110, 130]. Thus, even when signal components have different meanings, the robustness score (2.4) treats them all the same and combines them into a single number. However, application settings may consider multiple robustness scores from the same specification, e.g., in Pb. 3.3.1, we consider capability and resource predicates stemming from semantically disjoint variables. To capture this, we introduce the *multi-robustness* semantics of an STL specification, which generalizes the standard quantitative semantics to multiple predicate classes. The definition follows for CaTL since it is a fragment of STL. Throughout the paper, we assume that STL formulas are in PNF form.

Definition 3.3.6 (Multi-robustness for Multi-Class STL)**.** *A Multi-Class STL (mcSTL) formula is a tuple $\psi = (\phi, \Sigma, \mathcal{L})$, where ϕ is an STL formula, Σ is a set of predicate classes, $\mathcal{L} : \mathfrak{P}(\phi) \to \Sigma$ is a function that maps each predicate $\varsigma \in \mathfrak{P}(\phi)$ to its predicate class $\sigma \in \Sigma$,*

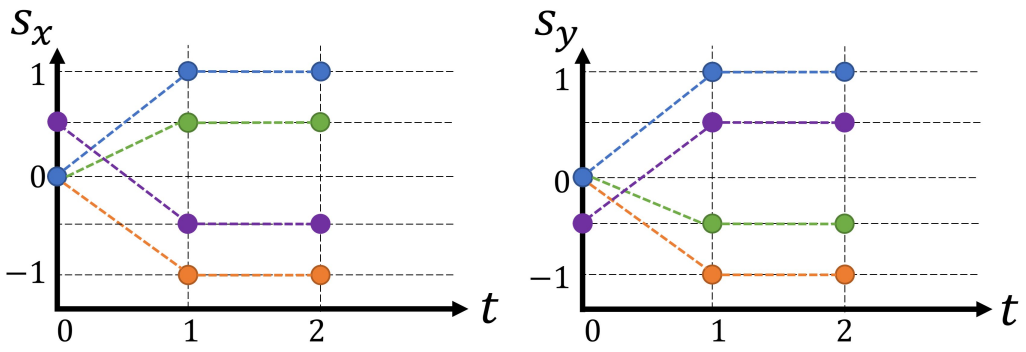


Figure 3.3.3: Ex. 3.3.3: Four different solution trajectories for ϕ_1^{ex}, ϕ_3^{ex}, ϕ_3^{ex}, and ϕ_4^{ex}.

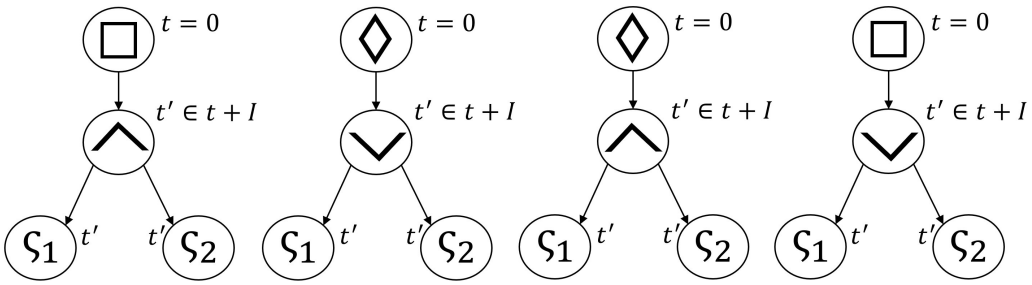


Figure 3.3.4: Ex. 3.3.3: AST for ϕ_1^{ex}, ϕ_2^{ex}, ϕ_3^{ex}, and ϕ_4^{ex}.

and $\mathfrak{P}(\phi)$ *is the set of predicates of* ϕ*. The multi-robustness score* ρ_σ *with respect to class* σ *of a predicate* $\varsigma = s_i \geq 0$ *is*

$$\rho_\sigma(s, \phi, t) ::= \begin{cases} \varsigma(s_i(t)), & \phi = \varsigma,\ \mathcal{L}(\varsigma) = \sigma, \\ \varnothing, & \phi = \varsigma,\ \mathcal{L}(\varsigma) \neq \sigma, \end{cases} \tag{3.22}$$

where s *is a signal. The robustness of temporal and logical operators* [5] *are computed in the same manner as in* (2.4)*. When* $\varnothing$ *is encountered in a computation, the operand is ignored, i.e.,* $a \otimes \varnothing = a$*,* $\varnothing \otimes \varnothing = \varnothing$*, and* $-\varnothing = \varnothing$ *where* $a \in \mathbb{R}$ *and* $\otimes \in \{\min, \max\}$*.*

Note that $\rho_\sigma(s, \phi, 0) = \varnothing$ if and only if predicates from the class σ do not appear in ϕ, i.e., $\mathcal{L}^{-1}(\sigma) = \varnothing$.

In the following, we show that the classes' robustness are novel scores ρ_σ, $\sigma \in \Sigma$, that capture different aspects of satisfaction and violation than the overall robustness.

[5] For brevity, we identify the mcSTL formula ψ with its STL formula ϕ when the classes and labeling functions are clear from context.

Example 3.3.3. *Consider the following STL specifications* $\phi_1^{ex} = \Box_{[1,2]}(\varsigma_1 \wedge \varsigma_2)$, $\phi_2^{ex} = \Diamond_{[1,2]}(\varsigma_1 \vee \varsigma_2)$, $\phi_3^{ex} = \Diamond_{[0,1]}(\varsigma_1 \wedge \varsigma_2)$, *and* $\phi_4^{ex} = \Box_{[0,1]}(\varsigma_1 \vee \varsigma_2)$. *with* $\varsigma_1 = s_x > 0$, $\varsigma_2 = s_y > 0$, $\varsigma_1 \in \sigma_1$, $\varsigma_2 \in \sigma_2$ *and* $\Sigma = \{\sigma_1, \sigma_2\}$. *The three specifications' abstract syntax trees (AST) are shown in Fig. 3.3.4. The following formulas give the robustness scores for the three specifications and a signal* s*:*

$$\rho(s, \phi_1^{ex}, 0) = \min\{\min\{s_x(1), s_y(1)\}, \min\{s_x(2), s_y(2)\}\},$$
$$\rho(s, \phi_2^{ex}, 0) = \max\{\max\{s_x(1), s_y(1)\}, \max\{s_x(2), s_y(2)\}\},$$
$$\rho(s, \phi_3^{ex}, 0) = \max\{\min\{s_x(0), s_y(0)\}, \min\{s_x(1), s_y(1)\}\},$$
$$\rho(s, \phi_4^{ex}, 0) = \min\{\max\{s_x(0), s_y(0)\}, \max\{s_x(1), s_y(1)\}\}.$$

We obtain the robustness scores for σ_1 *and* σ_2 *by replacing* $s_x(t)$ *and* $s_y(t)$ *for* $t \in 1, 2$ *with* $\varnothing$*, respectively. For example,* $\rho_{\sigma_1}(s, \phi_1^{ex}, 0) = \min\{\min\{s_x(1), \varnothing\}, \min\{s_x(2), \varnothing\}\} = \min\{s_x(1), s_x(2)\}$.

Consider the blue, green, and orange trajectories in Fig. 3.3.3. The overall and class robustness scores computed using (2.4) *and* (3.22)*, respectively, are given in Table 3.2.*

Table 3.2: STL robustness scores for the overall formula and for classes σ_1 and σ_2.

Signal	ϕ	ρ	ρ_{σ_1}	ρ_{σ_2}
s^{blue}	ϕ_1^{ex}	1	1	1
s^{green}	ϕ_1^{ex}	-0.5	0.5	-0.5
s^{orange}	ϕ_1^{ex}	-1	-1	-1
s^{blue}	ϕ_2^{ex}	1	1	1
s^{green}	ϕ_2^{ex}	0.5	0.5	-0.5
s^{orange}	ϕ_2^{ex}	-1	-1	-1
s^{purple}	ϕ_3^{ex}	-0.5	1	0.5
$-s^{purple}$	ϕ_4^{ex}	0.5	-1	-0.5

Positive overall robustness does not imply that any class robustness score needs to be positive. The counter-examples at lines 5 and 8 of Table 3.2 illustrate cases with 1 and 2 negative class robustness scores and positive overall robustness, respectively. Similarly, negative overall robustness provides no information on the class scores. Counter-examples

at lines 2, 3, and 7 of Table 3.2 have negative overall robustness, but 1, 2, and 0 have negative class robustness scores, respectively.

As shown in Ex. 3.3.3, there is no relation between the signs of the multi-robustness values for mcSTL and the sign of the overall robustness. Thus, there is no direct soundness property stemming from multi-robustness. The combination of min and max operators from Boolean and temporal operators hinders the consistency when predicate classes are modulated. To address this inconsistency, we introduce a fragment of STL called class-complete mcSTL (mcSTL*), defined as follows.

Definition 3.3.7 (Class-complete mcSTL)**.** *A class-complete mcSTL (mcSTL*) is an mcSTL formula $\psi = (\phi, \Sigma, \mathcal{L})$ that has the following properties. For all $\varsigma \in \mathfrak{P}(\phi)$*

$$(1)\, pa_\phi(\varsigma) = \wedge,$$
$$(2)\, \forall \sigma' \in \Sigma \setminus \{\mathcal{L}(\varsigma)\},\ \exists \varsigma' \text{ such that, } \mathcal{L}(\varsigma') = \sigma', \text{ and } \varsigma' \in ch_\phi(pa_\phi(\varsigma)),$$

where $pa_\phi(\phi')$ and $ch_\phi(\phi')$ return the parent and children formulas of ϕ' based on the AST of ϕ, respectively.

The two properties of mcSTL* mean that every predicate $\varsigma \in \mathfrak{P}(\phi)$ has a conjunction operator as its parent, and has sibling predicates from all predicate classes. When computing (3.22) over an mcSTL* specification ψ, a consistency property exists that relates class robustness to the overall specification robustness.

Theorem 3.3.1 (mcSTL* Multi-robustness consistency)**.** *Let s be a signal and $\psi = (\phi, \Sigma, \mathcal{L})$ an mcSTL* formula with predicate set $\mathfrak{P}(\phi) \neq \varnothing$ from classes Σ. The following property holds for all $t \geq 0$ and $\sigma \in \Sigma$*

$$\rho_\sigma(s, \phi, t) \geq \rho(s, \phi, t).$$

The proof of Thm. 3.3.1 is in Appendix 3.3.6.

Corollary 3.3.1. *Under the same setting as in Thm. 3.3.1 and* $\rho(s, \phi, t) > 0$*, it follows that* $\rho_\sigma(s, \phi, t) > 0$ *for all* $\sigma \in \Sigma$.

Cor. 3.3.1 implies the necessary condition that all robustness class values $\rho_\sigma(s, \phi, 0)$ are positive for the overall robustness $\rho(s, \phi, 0)$ to be positive. However, it is not sufficient.

3.3.5 mcSTL and mcSTL* Discussion about Limitations and Connections

Here, we discuss the connections and limitations of all temporal logic formalisms introduced in this section: STL, CaTL, mcSTL, and mcSTL*. STL is a powerful language that enables users to reason about continuous signals while abiding by logical and temporal constraints. It allows for incorporating multiple signal components and the computation of both qualitative and quantitative semantics. The qualitative semantics defined in (2.3) determine whether operators are satisfied or violated, while the quantitative semantics determine the degree of satisfaction as defined in (2.4). Furthermore, STL has been shown to possess soundness properties about its quantitative semantics, as demonstrated in Thm. 2.1.1. This theorem establishes a connection between an STL formula's satisfaction and its robustness score being greater than zero. However, when considering semantically unrelated predicates on a single specification, it amalgamates all predicate categories to calculate a single robustness score, which may not be ideal. For instance, consider predicates that pertain to the position and velocity of a vehicle and establish a specification that encompasses the car's safety regulations. It becomes imperative to determine if the vehicle's position is in proximity to violating safety standards and colliding with other vehicles, regardless of whether it abides by speed rules. If the velocity margin satisfaction is significantly larger, it could mask the potential danger of violating position rules when calculating the overall robustness score.

Motivated by computing robustness scores over a single specification that contains

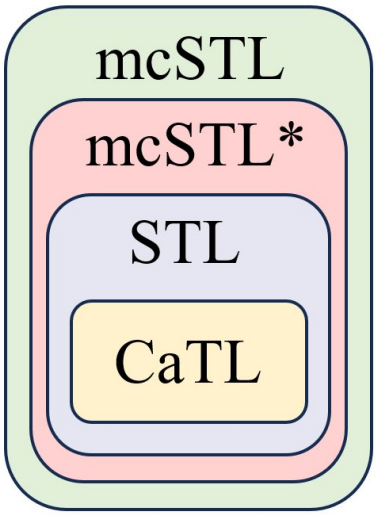


Figure 3.3.5: Set of expressivity definition of STL, CaTL, mcSTL, and mcSTL*.

multiple classes of predicates that are semantically disjoint between them, we have defined a semantic extension called multi-class Signal Temporal Logic (mcSTL), therefore, $STL \subseteq mcSTL$. Nevertheless, even though we can now define multiple types of predicates $\sigma \in \Sigma$ over the same specification, their individual robustness class $\rho_\sigma(s, \phi, t)$ and overall robustness scores $\rho(s, \phi, t)$ have not an evident correlation in their signs as shown in Table 3.2. The combination of $\min$ and $\max$ operators hinders the consistency of the signs when signals are modulated, making it difficult to determine the overall satisfaction of the specification. This poses a considerable limitation of this semantic extension on determining satisfaction without extensive bookkeeping of variables.

We define class-complete multi-class Signal Temporal Logic (mcSTL*) to address this inconsistency with $mcSTL^* \subseteq mcSTL$. mcSTL* requires that every predicate in the specification has as parent node a conjunction operator and all other classes existing as siblings. Consider, as an example, specifications $\phi_1^{ex} = \Box_{[1,2]}(\varsigma_1 \wedge \varsigma_2)$ and $\phi_3^{ex} = \Diamond_{[0,1]}(\varsigma_1 \wedge \varsigma_2)$ in Ex. 3.3.3, where $\Sigma = \{\sigma_1, \sigma_2\}$ with $\sigma_1 = \{\varsigma_1\}$ and $\sigma_2 = \{\varsigma_2\}$. For this type of specification, we have that all robustness classes have to be positive for the overall robustness to be positive, implying that the specification will be satisfied only if all individual classes are satisfied, as shown in Th. 3.3.1 and Cor. 3.3.1. Nonetheless, mcSTL*'s most signif-

icant limitation is its definition, requiring applications involving all predicate classes in conjunction whenever we want to define predicate properties in the specification.

In contrast to all other temporal logic formalisms discussed, CaTL allows us to give meaning to predicates for the number of agents and amount of resources. Note that the resource and agent availability problem studied in this section is just a particular case of the multi-robustness for mcSTL*. The definition of a CaTL task in Def. 3.3.3 includes the predicate classes in conjunction, in accordance with the mcSTL* properties. We have $\Sigma = \{\sigma_a, \sigma_h\}$, where σ_a defines the agents' class of predicates and σ_h the resources class of predicates. Allowing the linear combination computation of the robustness of different predicate classes such as agents' availability $\rho_a(s_{\mathcal{J}}, \phi, 0)$ and resources availability $\rho_h(b_{\mathcal{H}}, \phi, 0)$ shown in (3.21).

Therefore, some of the relations and properties of CaTL, STL, mcSTL*, and mcSTL are shown in Fig. 3.3.5 and Table 3.3.

Table 3.3: Properties hold for every semantic extension or fragment of STL discussed in this section.

	CaTL	STL	mcSTL*	mcSTL
Logical operators	✓	✓	✓	✓
Temporal operators	✓	✓	✓	✓
Robustness	✓	✓	✓	✓
Multi-class	✓	✗	✓	✓
Tasks	✓	✗	✗	✗
Multi-robustness consistency	✓	NA	✓	✗

We have defined a framework that allows us to handle specifications with multiple predicate classes and have mission satisfaction guarantees on their individual robustness computation being greater than zero. In the next section, we translate the problem formulation described in Sec. 3.3.3 into its corresponding MILP encoding. This enables us to compute agent and resource trajectories that satisfy the CaTL specification and dynamics constraints.

3.3.6 Proof of Theorem 3.3.1

Before we start the main argument of the proof, we show a general lemma on the monotonicity of evaluating mcSTL* formulas. For this, we define an *evaluation tree* based on the AST. Formally, an evaluation tree (ET) is a tuple $\mathcal{T} = (op, Ch = \{\mathcal{T}_{ch}\}_{ch})$, where $op \in \{\min, \max, \text{nop}\}$, nop denotes no operation, and Ch is a list of children's evaluation trees. A leaf node is one where $Ch = \varnothing$ and $op = \text{nop}$, while an intermediate node has children ETs ($Ch \neq \varnothing$) and $op \in \{\min, \max\}$. Let $\mathcal{S}(\mathcal{T})$ denote the list of leafs of $\mathcal{T}$.

Given an AST of a mcSTL* formula ψ, we construct the ET $\mathcal{T}$ as follows: (a) a terminal conjunction node, which has only predicates as operands, is translated to a leaf node, (b) non-terminal conjunction and disjunction nodes are translated to intermediate nodes with children ETs corresponding to its operands and $op = \min$ and $op = \max$, respectively, (c) always and eventually, nodes are translated to nodes with children for each time point in their associated time window and $op = \min$ and $op = \max$, respectively, and (d) until operators are translated as a combination of the previous two rules. Essentially, ETs capture the recursive evaluation process of the robustness scores in (2.4) and Def. 3.3.6.

Let $\nu : \mathcal{S}(\mathcal{T}) \to \mathbb{R}$ be an valuation for the leafs of ET $\mathcal{T}$. We define the tree evaluation $\eta(\mathcal{T}, \nu)$ by

$$\eta(\mathcal{T}, \nu) = \begin{cases} op(\mathcal{T}_1, \ldots, \mathcal{T}_{|Ch|}), & \mathcal{T}_\ell \in Ch \neq \varnothing, \\ \nu(\mathcal{T}), & \text{otherwise.} \end{cases} \tag{3.23}$$

Lemma 3.3.1. *Let $\mathcal{T}$ be an ET and ν and ν' two leaf valuations. If $\nu'(\varsigma) \geq \nu(\varsigma)$ for all $\varsigma \in \mathcal{S}(\mathcal{T})$, then $\eta(\mathcal{T}, \nu') \geq \eta(\mathcal{T}, \nu)$.*

Proof. The proof follows by structural induction over ET $\mathcal{T}$.

Base case: Let $\mathcal{T} = (\text{nop}, \varnothing)$ be a leaf node. We have $\eta(\mathcal{T}, \nu') = \nu'(\mathcal{T}) \geq \nu(\mathcal{T}) = \eta(\mathcal{T}, \nu)$.

Induction step: Let $\mathcal{T} = (op, Ch)$ be a tree with children, i.e., $Ch \neq \varnothing$. In case $op = \min$, we have that $\eta(\mathcal{T}, \nu) = \min_{\mathcal{T}_{ch} \in Ch}\{\eta(\mathcal{T}_{ch}, \nu)\}$. By the induction hypothesis, it follows that $\eta(\mathcal{T}_{ch}, \nu') \geq \eta(\mathcal{T}_{ch}, \nu)$ for all $\mathcal{T}_{ch} \in Ch$. Finally, we have $\eta(\mathcal{T}, \nu') = \min_{\mathcal{T}_{ch} \in Ch}\{\eta(\mathcal{T}_{ch}, \nu')\} \geq$

$\min_{\mathcal{T}_{ch} \in Ch}\{\eta(\mathcal{T}_{ch}, \nu)\} = \eta(\mathcal{T}, \nu)$. The case $op = \max$ follows similarly. □

Proof of Theorem 3.3.1.

Proof. Let $\mathcal{T}_\phi$ be the ET of mcSTL* formula ϕ, and s be a signal such that $\rho(s, \phi, t) > 0$.

We define ET valuation $\nu_\phi : \mathcal{S}(\mathcal{T}_\phi) \to \mathbb{R}$ such that $\nu_\phi(\mathcal{T}) = \min_{\varsigma \in ch_\phi(\phi_\mathcal{T})} \rho(s, \varsigma, t_\mathcal{T})$, where $(t_\mathcal{T}, \phi_\mathcal{T})$ is the time-formula pair associated with ET leaf node $\mathcal{T}$. By construction, each $\mathcal{T} \in \mathcal{S}(\phi)$ corresponds to a terminal conjunction whose operands are predicates covering all predicate classes Σ. Thus, the tree evaluation captures the robustness computation for signal s with respect to ϕ at time t. Formally $\eta(\mathcal{T}_\phi, \nu) = \rho(s, \phi, t)$. Similarly, we define ET valuation $\nu_{\phi,\sigma} : \mathcal{S}(\mathcal{T}_\phi) \to \mathbb{R}$ such that $\nu_\phi(\mathcal{T}) = \min_{\varsigma \in ch_\phi(\phi_\mathcal{T})} \rho_\sigma(s, \varsigma, t_\mathcal{T})$. The tree evaluation using $\nu_{\phi,\sigma}$ captures the robustness score with respect to predicate class $\sigma \in \Sigma$. Formally, $\eta(\mathcal{T}_\phi, \nu_{\phi,\sigma}) = \rho_\sigma(s, \phi, t), \forall \sigma \in \Sigma$.

Since all terminal conjunctions have predicates from all classes as operands, their robustness with respect to any class is defined, i.e., not equal to $\varnothing$. It follows that $\nu_{\phi,\sigma}(\mathcal{T}) = \min_{\varsigma \in ch^\sigma_\phi(\phi_\mathcal{T})} \rho(s, \varsigma, t_\mathcal{T}) \geq \min_{\varsigma \in ch_\phi(\phi_\mathcal{T})} \rho(s, \varsigma, t_\mathcal{T}) = \nu_\phi(\mathcal{T})$ since $ch^\sigma_\phi(\mathcal{T}) \subset ch_\phi(\mathcal{T})$ for all $\mathcal{T} \in \mathcal{S}(\mathcal{T}_\phi)$ and $\sigma \in \Sigma$, where $ch^\sigma_\phi(\mathcal{T}) = \{\varsigma \in ch_\phi(\mathcal{T}) \mid \mathcal{L}(\varsigma) = \sigma\}$. Thus, by Lemma 3.3.1 we have $\rho_\sigma(s, \phi, t) = \eta(\mathcal{T}_\phi, \nu_{\phi,\sigma}) \geq \eta(\mathcal{T}_\phi, \nu_\phi) = \rho(s, \phi, t)$ which concludes the proof. □

3.3.7 Mixed Integer Linear Programming Encoding

This section describes the encoding of agents' and resource dynamics, CaTL specifications defined in Sec. 3.3.3 as a MILP. We formulate Pb. 3.3.1 using these variables and define an objective function that captures the desired behavior of the robot fleet, maximizing the robot availability and resource availability robustness while minimizing the total travel time of robots and resources.

Agents' Dynamics Encoding

We introduce agent class decision variables for all states and edges in the environment Env over the planning time horizon $K = \|\phi\|$ as in (2.5) to encode agent dynamics.

Let $z_{q,g,k} \in \mathbb{Z}_{\geq 0}$ represent the number of agents of class $g \in \mathcal{G}$ at time $k = \Delta t$ at state $q \in \mathcal{Q}$, and $u_{e,g,k} \in \mathbb{Z}_{\geq 0}$ the number of agents of class $g \in \mathcal{G}$ entering edge $e \in \mathcal{E}$ at time k, for all $k \in [0..K]$. Then the agents' dynamics [75] for all $q \in \mathcal{Q}$, $g \in \mathcal{G}$, $k \in [0..K]$ is captured as follows

$$z_{q,g,0} = |\{j \in \mathcal{J} \mid q_{0,j} = q, c_j = g\}|, \tag{3.24}$$

$$z_{q,g,k} = \sum_{(q',q)\in\mathcal{E}} u_{(q',q),g,k-\mathcal{W}((q',q))}, \tag{3.25}$$

$$\sum_{(q,q')\in\mathcal{E}} u_{(q,q'),g,k} = \sum_{(q',q)\in\mathcal{E}} u_{(q',q),g,k-\mathcal{W}((q',q))}, \tag{3.26}$$

where (3.24) captures the initial agent distribution over the environment; specifically, the number of agents $j \in \mathcal{J}$ with a capability set c_j related to class g at $k = 0$ present at region $q \in \mathcal{Q}$. Note that the same agent cannot be at two places simultaneously and that no agent is allowed to be initially at edges $\mathcal{E}$. On the other hand, (3.25) and (3.26) capture the agent distribution at every time k such that the flow of agents is conserved over the environment Env at all times. Thus, the number of agents entering a node $(q', q) \in \mathcal{E}$ must equal the number of agents going out $(q, q') \in \mathcal{E}$. This covers the case of agents staying at state q modeled as taking the self-loop $(q, q) \in \mathcal{E}$ for a one-time unit.

Resource Dynamics Encoding

Similarly, as for agents' dynamics, we encode resource dynamics by imposing flow constraints using the following variables. Let $y_{q,h,k} \in \mathbb{H}$ represents the quantity of resource $h \in \mathcal{H}$ at time $k \in [0..K]$ at state $q \in \mathcal{Q}$, and $v_{e,h,k} \in \mathbb{H}$ represents the quantity of resource $h \in \mathcal{H}$ entering edge $e \in \mathcal{E}$ at time $k \in [0..K]$. Note that resources can be divisible $\mathbb{H} = \mathbb{R}_{\geq 0}$,

indivisible $\mathbb{H} = \mathbb{Z}_{\geq 0}$, or binary $\mathbb{H} = \mathbb{B}$. Lastly, let $C_{q,h,k}$ be the cross-consumption constraint, which ensures that the same resource is not used more than once in case different tasks ask for the same resource simultaneously and at the same location. Then resource dynamics for all $q \in \mathcal{Q}$, $h \in \mathcal{H}$, and $k \in [0..K]$, is encoded as follows

$$y_{q,h,0} = b_{\mathcal{H}}(0, q), \tag{3.27}$$

$$y_{q,h,k} = \sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}((q',q))}, \tag{3.28}$$

$$C_{q,h,k} + \sum_{(q,q')\in\mathcal{E}} v_{(q,q'),h,k} = \sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}((q',q))}, \tag{3.29}$$

where (3.27) captures how the resources $h \in \mathcal{H}$ are distributed over the environment Env at time $k = 0$. Similarly, (3.28) and (3.29) represent the resource distribution at every time step $k \in [0..K]$, and impose the conservation of resource flows while considering cross-consumption constraint (3.38) defined below.

We can now impose the transportation constraints that resources must be carried by agents using the agent and resource flow variables. These constraints depend on the transportation type, either uniform or compartmental, as described in Sec. 3.3.3. The motion for all of the resources $h \in \mathcal{H}$, over every edge $e = (q, q') \in \mathcal{E}$ for all $k \in [0..K]$, is

$$v_{e,h,k} \leq \sum_{g\in\mathcal{G}_h} \Omega_{h,g} u_{e,g,k}, \qquad \text{(compartmental)}, \tag{3.30}$$

$$\sum_{h\in\mathcal{H}} v_{e,h,k} \leq \sum_{g\in\mathcal{G}} \Omega_g u_{e,g,k}, \qquad \text{(uniform)}, \tag{3.31}$$

where $u_{e,g,k}$ is the number of agents of class $g \in \mathcal{G}$ entering edge $e \in \mathcal{E}$ at time k defined in Sec. 3.3.7 and $\Omega_{h,g}$, Ωg are the transportation capacities for the compartmental and uniform cases, respectively. Thus, (3.30) and (3.31) guarantee the total amount of resource $h \in \mathcal{H}$ transported by agents do not exceed the maximum capacity that agents can transport based

on storage capacity and type. Variable $v_{e,h,k}$ is bounded by

$$\overline{v_{e,h,k}} = \min\{|\mathcal{J}| \cdot \max_{g \in \mathcal{G}} \Omega_{g,h}, \sum_{q \in \mathcal{Q}} b_h(0,q)\},$$

for compartmental transportation, for all $h \in \mathcal{H}$, and

$$\overline{v_{e,h,k}} = \min\{|\mathcal{J}| \cdot \max_{g \in \mathcal{G}} \Omega_{g}, \sum_{q \in \mathcal{Q}} \sum_{h \in \mathcal{H}} b_h(0,q)\},$$

for uniform transportation. In the case of package transportation (binary resource), we set $y_{q,h,k}$ and $v_{e,h,k}$ as binary variables.

Remark 3.3.3. *The main insight of the encoding is that we do not need to assign resources to agents explicitly. We only need to ensure that agents can transport resources along their routes. Thus, we extract the resource trajectories from resource flows similar to the agent case, see Sec. 3.3.7 for details. Moreover, no binary and integer auxiliary variables are required to enforce a satisfiable solution.*

CaTL Specification Encoding

We translate the CaTL specification ϕ to be captured into the MILP in three steps.

First, we introduce the variables $z_{\pi,q,c,k} \in \mathbb{R}_{\geq 0}$, $y_{\pi,q,h,k} \in \mathbb{R}_{\geq 0}$ that capture the amount of agents with capability $c \in \mathcal{C}$ and resources $h \in \mathcal{H}$ at time step $k \in [0..K]$ at every state $q \in \mathcal{Q}$ using label $\pi \in \mathcal{AP}$. We couple them with the system variables for agents and resources, $z_{q,g,k}$ and $y_{q,h,k}$, respectively. The following constraints ensure that capabilities and resources are not counted more than once in all regions with atomic proposition π

labeled with $L(q)$

$$\sum_{\pi \in L(q)} z_{\pi,q,c,k} = \sum_{g:c \in G} z_{q,g,k}, \tag{3.32}$$

$$\sum_{\pi \in L(q)} y_{\pi,q,h,k} = y_{q,h,k}. \tag{3.33}$$

Second, we introduce variables $z_{\varsigma(\pi,c),k} \in \mathbb{R}$ for agents and $y_{\varsigma(\pi,h),k} \in \mathbb{R}$ for resources, coupled with the counting proposition variables (see (3.36) and (3.37) for definitions, respectively). We use these variables to guarantee that the number of agent capabilities and resources in the regions is at least the amount requested in the counting propositions. We have

$$z_{\varsigma(\pi,c),k} \leq z_{\pi,q,c,k}, \tag{3.34}$$

$$y_{\varsigma(\pi,h),k} \leq y_{\pi,q,h,k}, \tag{3.35}$$

for all $q \in L^{-1}(\pi)$, $c \in \mathcal{C}$, $h \in \mathcal{H}$, $\pi \in \mathcal{AP}$, and $k \in [0..K]$.

Lastly, we encode the CaTL specification for the satisfaction of tasks by translating it into an STL specification. This is possible since CaTL is a fragment of STL. CaTL allows intuitive, compact encodings of tasks, and like STL, it can be efficiently encoded as a MILP [95, 22, 103, 30] by recursively encoding the specification based on the robustness definition (2.4). A task $T = (d, \pi, cp, rs)$ defined as in Def. 3.3.1 is semantically equivalent to the following STL specification formula

$$\phi_T = \bigwedge_{h \in \mathcal{H}} \varsigma(\pi, h) \wedge \square_{[0,d]} \bigwedge_{c \in \mathcal{C}} \varsigma(\pi, c),$$

where $\varsigma(\pi, c)$ and $\varsigma(\pi, h)$ are

$$\varsigma(\pi, c) = \min_{q \in L^{-1}(\pi)} \{n_{q,c}\} \geq cp(c), \tag{3.36}$$

$$\varsigma(\pi, h) = \min_{q \in L^{-1}(\pi)} \{b_h(\cdot, q)\} \geq rs(h). \tag{3.37}$$

Therefore, CaTL specification can be fully translated into an STL specification and further added into the MILP by encoding the operators using (3.22). Satisfaction can be guaranteed via the binary variable associated with the satisfaction of the overall STL specification [130, 133, 106].

Resources Cross-consumption Encoding

Before introducing the objective function, we define the cross-consumption constraint which ensures that a resource is used once and immediately consumed in the required amount at the start of a task's satisfaction. Let $\Lambda(q) = \{T = (d, \pi, cp, rs) \mid \pi \in L(q)\}$ be the set that captures all the tasks that are going to be satisfied at the same location. Resource cross-consumption is given by

$$C_{q,h,k} = \sum_{T \in \Lambda(q)} rs(h) \cdot x_{\phi_T,k}, \tag{3.38}$$

where $x_{\phi_T,k} \in \mathbb{B}$, capture whether or not the task T in the set $\Lambda(q)$ is satisfied. Thus, the cross-consumption constraint is

$$C_{q,h,k} \leq y_{q,h,k}, \tag{3.39}$$

for all $q \in \mathcal{Q}$, $h \in \mathcal{H}$, $k \in [0..K]$. In other words, (3.39) ensures that the total amount of resources needed to satisfy multiple tasks in overlapping regions is less or equal to the number of resources at that location.

Objective Definition

Here, we formulate Pb. 3.3.1 as an optimization problem (MILP), which can be solved using any off-the-shelf software tool. In addition to the objective in Pb. 3.3.1, we want to account for agents' and resources' travel time and eliminate inefficient behavior, i.e., spurious motion. For instance, agents taking longer paths or transporting resources even when they are not required to satisfy tasks wastes time and energy. We use regularization terms based on the weighted total travel times for agents and resources

$$\tau_u = \sum_{k=0}^{K} \sum_{g \in \mathcal{G}} \sum_{e=(q,q') \in \mathcal{E}, q \neq q'} u_{e,g,k}, \tag{3.40}$$

$$\gamma_u = \frac{\alpha_u}{K \cdot |\mathcal{J}|}, \tag{3.41}$$

$$\tau_v = \sum_{k=0}^{K} \sum_{h \in \mathcal{H}} \sum_{e=(q,q') \in \mathcal{E}, q \neq q'} v_{e,h,k}, \tag{3.42}$$

$$\gamma_v = \frac{\alpha_v}{K \cdot \max_h \sum_{q \in \mathcal{Q}} b_h(0,q)}, \tag{3.43}$$

where $\alpha_v, \alpha_u \in [0,1]$ and τ_u, τ_v are the total travel times of the agents and resources moving along edges in the environment. The weights γ_u, γ_v scale the regularization terms to ensure that agents and resources robustness have higher priority; agent and resource regularization does not come at the expense of maximizing either agents or resources robustness.

Finally, the overall optimization problem of maximizing agents and resources' robust-

ness while minimizing agent and resource traveling time is

$$
\begin{aligned}
&\max_{z,u,y,v} \rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v, \\
&\quad \text{s.t. } (s_{\mathcal{J}}, b_{\mathcal{H}}) \vDash \phi, \\
&\quad (3.24), (3.25), (3.26), \quad \text{(Agent dynamics)}, \\
&\quad (3.27), (3.28), (3.29), \quad \text{(Resource dynamics)}, \\
&\quad (3.30) \text{ or } (3.31), \quad \text{(Agent resource capacity)}, \\
&\quad (3.32) - (3.35), \quad \text{(Task satisfaction)}, \\
&\quad (3.38) \text{ and } (3.39), \quad \text{(Resources cross-consumption)}.
\end{aligned}
\tag{3.44}
$$

Note that the solution to the problem provides us with the counts of agents and amounts of resources required to meet the mission specification and dynamics constraints. However, it does not explicitly assign agents for transporting and fulfilling the tasks. To solve this issue, the next section introduces an algorithm that can be used for agent and resource assignments.

Agent and Resource Assignments

The outcome of solving (3.44) are $z_{q,g,k}$, $y_{q,h,k}$, $u_{(q,q'),g,k}$, and $v_{(q,q'),h,k}$ capturing the number of robots of each class $g \in \mathcal{G}$ and the amount of resource $h \in \mathcal{H}$ at time $k \in [0..K]$ at either state $q \in \mathcal{Q}$ or traversing edge $(q, q') \in \mathcal{E}$, respectively. However, these values do not provide explicit information about the trajectories of individual robots or the assignment of resources to robots. Therefore, we extend the approach presented in [95] for extracting individual robot trajectories and assigning resource transportation to robots. The method is described in the Alg. 4.

The sets $\mathcal{U}_{\mathcal{E},\mathcal{G},K} = \{u_{e,g,k} \mid e \in \mathcal{E}, g \in \mathcal{G}, k \in [0..K]\}$, and $\mathcal{V}_{\mathcal{E},\mathcal{H},K} = \{v_{e,h,k} \mid e \in \mathcal{E}, h \in \mathcal{H}, k \in [0..K]\}$ representing the MILP solution are the inputs of the method. Alg. 4 is executed for each time step, generating a trajectory for each agent and assigning resources

to agents that fulfill the mission specification ϕ. The trajectory of a robot j is represented as a sequence of states and edges $s_j(k) = q_{0,j} \dots (q, q') \dots q_{K,j}$, where $k \in [0..k]$ starting at the initial position $s_j(0) = q_{0,j}$ of agent $j \in \mathcal{J}$, see Sec. 3.3.3. Let $\mathcal{J}_{q,g,k}$ be the set of agents of class $g \in \mathcal{G}$, at state $q \in \mathcal{Q}$, at time $k \in [0..K]$. The transportation assignment of resource h to agent j at time k is denoted by $\mathcal{B}_{\mathcal{H}}(j, h, k)$. The procedure initializes each agent j's trajectory with its initial state $q_{0,j}$, and the sets of agents at each state q of class g at time 0, lines 1-2. For each time step, the algorithm propagates agents from states onto outgoing transitions (line 4), iteratively computes the agents trajectories (lines 5-7), recomputes agents at states (line 9), and allocates resources to agents for transportation (line 10-11). In the first step of the loop, the set of agents at each state q from each class g is partitioned over the outgoing transitions $\mathcal{E}^+(q) = \{(q, q') \in \mathcal{E}\}$ using function $Part()$ based on the agent numbers from the solution $\mathcal{U}_{\mathcal{E}^+(q),g,k}$, line 4. The trajectories of agents $\mathcal{J}_{e,g,k}$ departing on each outgoing transition $e = (q, q')$ are extended over the transition's duration $\mathcal{W}(e)$, lines 5-7. The third step, computes the agents $\mathcal{J}_{q,g,k+1}$ from each class g arriving at each state q at time $k + 1$ as the union over incoming transitions $\mathcal{E}^-(q) = \{(q', q) \in \mathcal{E}\}$, line 9.

Finally, we use the $Alloc(\bigcup_{g \in \mathcal{G}} \mathcal{J}_{e,g,k}, (v_{e,h,k})_{h \in \mathcal{H}})$ function to assign resources that must be transported by agents traveling along edges $e \in \mathcal{E}$ according to their corresponding storage type and capacity, lines 10-11, described in Alg. 5.

It formulates a Linear Programming (LP) problem to allocate amount $\xi_{j,h} \in \mathbb{R}_{\geq 0}$ of resources h to be transported by agent $j \in J$ according to its storage type and capacity. The LP's variables are $\xi_{j,h}$ and are created for all agents in the input set J, line 2. Next, we create the constraints set $\mathcal{CS}$ for resource allocation. The first constraint requires the exact allocation of the amount $v(h)$ for each resource $h \in \mathcal{H}$ among agents in j, line 3. If the storage type is uniform, we add the total transportation capacity bound Ω_{c_j} for each agent j, lines 4-5. Conversely, if the storage type is compartmental, we add the capacity bounds Ω_{h,c_j} for each resource h and each agent j, lines 6-7. Finally, we solve the LP and allocate

Algorithm 4 Robot trajectories and resource transportation assignments.

Require: $\mathcal{U}_{\mathcal{E},\mathcal{G},K},\ \mathcal{V}_{\mathcal{E},\mathcal{H},K}$
Ensure: $\mathcal{S}_{\mathcal{J}},\ \mathcal{B}_{\mathcal{H}}$
1: $s_j(0) \leftarrow q_{0,j}, \quad \forall j \in \mathcal{J}$
2: $\mathcal{J}_{q,g,0} \leftarrow \{\, j \in \mathcal{J} \mid q = q_{0,j},\ g = c_j \,\}$
3: **for** $k \leftarrow 0$ **to** $K-1$ **do**
4: $\quad \mathcal{J}_{\mathcal{E}^+(q),g,k} \leftarrow Part\Big(\mathcal{J}_{q,g,k}, \mathcal{U}_{\mathcal{E}^+(q),g,k}\Big), \quad \forall q \in \mathcal{Q},\ \forall g \in \mathcal{G}$
5: $\quad$ **for** each $j \in \mathcal{J}_{e,g,k}$, for all $q \in \mathcal{Q}$ and $g \in \mathcal{G}$, where $e = (q, q')$ **do**
6: $\quad\quad s_j((k+1)\dots(k+\mathcal{W}(e)-1)) \leftarrow e$
7: $\quad\quad s_j(k+\mathcal{W}(e)) \leftarrow q'$
8: $\quad$ **end for**
9: $\quad \mathcal{J}_{q,g,k+1} \leftarrow \bigcup_{e \in \mathcal{E}^-(q)} \mathcal{J}_{e,g,k+1-\mathcal{W}(e)}, \quad \forall q \in \mathcal{Q},\ \forall g \in \mathcal{G}$
10: $\quad$ **for** each $e \in \mathcal{E}$, and each $k' \in \{k, k+1, \dots, k+W(e)\}$ **do**
11: $\quad\quad \mathcal{B}_{\mathcal{H}}(\cdot,\cdot,k') \leftarrow Alloc\Big(\bigcup_{g \in \mathcal{G}} \mathcal{J}_{e,g,k},\ (v_{e,h,k})_{h \in \mathcal{H}}\Big)$
12: $\quad$ **end for**
13: **end for**
14: **return** $\mathcal{S}_{\mathcal{J}},\ \mathcal{B}_{\mathcal{H}}$

the amounts of resource $\xi^*_{j,h}$ to agent j, line 9.

Algorithm 5 Resource Assignments – $Alloc()$

Require: $J \subseteq \mathcal{J}, V : \mathcal{H} \to \mathbb{R}_{\geq 0}$
1: **Param:** Ω_g or $\Omega_{h,g}$, $\forall g \in \mathcal{G},\ h \in \mathcal{H}$, `type`
Ensure: $(\xi^*_{j,h})_{j \in J, h \in \mathcal{H}}$ ▷ Amount of h transported by agent j
2: Create $\xi_{j,h} \in \mathbb{R}_{\geq 0}$, $\forall j \in J,\ h \in \mathcal{H}$
3: $\mathcal{CS} \leftarrow \{\sum_{j \in J} \xi_{j,h} = V(h) \mid h \in \mathcal{H}\}$
4: **if** `type = 'uniform'` **then**
5: $\quad \mathcal{CS} \leftarrow \mathcal{CS} \cup \{\sum_{h \in \mathcal{H}} \xi_{j,h} \leq \Omega_{c_j} \mid \forall j \in J\}$
6: **else if** `type = 'compartmental'` **then**
7: $\quad \mathcal{CS} \leftarrow \mathcal{CS} \cup \{\xi_{j,h} \leq \Omega_{h,c_j} \mid \forall j \in J,\ h \in \mathcal{H}\}$
8: **end if**
9: Solve LP: $\xi^*_{j,h} = \arg\min_{\xi_{j,h}} 1$ s.t. $\mathcal{CS}$
10: **return** $(\xi^*_{j,h})_{j \in J, h \in \mathcal{H}}$

3.3.8 Encoding Generalization

In this section, we show extensions to the solution presented in Sec. 3.3.7, that handle generalized storage configurations and external resource dynamics, i.e., resource creation

and destruction during deployment. Moreover, we present an encoding for gradual resource creation during a task's satisfaction.

Generalized storage configurations

In Sec. 3.3.7, we showed uniform and compartmental storage setups encodings. However, robots may have a complex combination of these storage types. See the example in Fig. 3.3.6.

Let $\mathcal{H}_a \subseteq \mathcal{H}$ define the storage compartment $a \in \mathcal{A}$ as the set of resources that it can store. All resources must be handled, i.e., $\mathcal{H} = \bigcup_{a \in \mathcal{A}} \mathcal{H}_a$. Compartmental storage requires $|\mathcal{H}_a| = 1$ for all $a \in \mathcal{A}$, while uniform storage corresponds to $|\mathcal{A}| = 1$. Let $\Omega_{a,g}$ be the total transportation capacity of compartment a on an agent from class $g \in \mathcal{G}$ for all resources $\mathcal{H}_a$. Next, we consider first the case where resources are stored only in one compartment and then generalize it to the case of multiple compartments.

Case 1 (Mutually Exclusive): In this case, each resource can be stored only in a single compartment. Thus, the compartments are mutually exclusive and partition the set of resources. Formally, $\mathcal{H}_a \cap \mathcal{H}_{a'} = \varnothing$ for all $a \neq a'$.

The resource transportation capacity constraints become

$$\sum_{h \in \mathcal{H}_a} v_{e,h,k} \leq \sum_{g \in \mathcal{G}_a} \Omega_{a,g} u_{e,g,k}, \tag{3.45}$$

for all $e \in \mathcal{E}$, $k \in [0..K]$, $a \in \mathcal{A}$, where $\mathcal{G}_a = \{g \mid \Omega_{a,g} > 0\}$ is the set of agent classes that have compartment $a \in \mathcal{A}$. The constraints in (3.30) and (3.31) are special cases of (3.45).

Case 2 (Overlapping): When resources can be stored in multiple compartments, we must track the amount of each resource in each compartment. Let $\mathcal{A}(h) = \{a \in \mathcal{A} \mid h \in \mathcal{H}_a\} \neq \varnothing$ be the set of compartments that can store resource $h \in \mathcal{H}$.

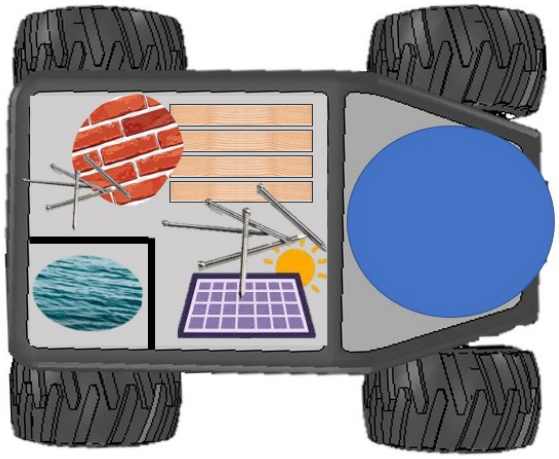


Figure 3.3.6: Example of generalization of storage type: a combination of compartmental and uniform storage.

We introduce variables $v_{e,h'_a,k}$ that indicate the amount of resource h in compartment a, and the constraints

$$v_{e,h,k} = \sum_{a \in \mathcal{A}(h)} v_{e,h'_a,k} \tag{3.46}$$

$$\sum_{h \in \mathcal{H}_a} v_{e,h'_a,k} \leq \sum_{g \in \mathcal{G}_a} \Omega_{a,g} u_{e,g,k} \tag{3.47}$$

for all $e \in \mathcal{E}$, $k \in [0..K]$, $a \in \mathcal{A}$. When $|\mathcal{A}(h)| = 1$, we have the previous case. Specifically, (3.46) and (3.47) become $v_{e,h,k} = v_{e,h'_a,k}$ and (3.45), respectively.

Generalized creation, destruction, and gradual consumption of resources

The encoding discussed in Sec. 3.3.7 assumes predetermined total amounts for each resource. However, resources may be added or removed from the environment by external processes (e.g., delivery and material loss) or by agents' actions (e.g., mining). Additionally, when the resources are used to satisfy a task, we consider these to disappear from the environment at the beginning of a task and are no longer available. However, commonly resources are consumed gradually by tasks.

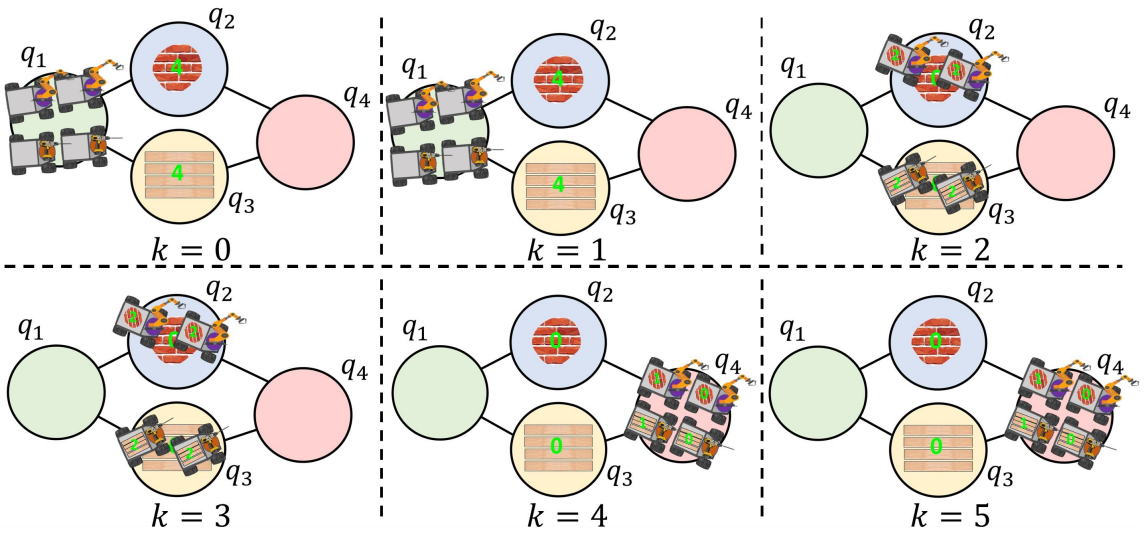


Figure 3.3.7: Solution computed for basic example in Sec. 3.3.9.

Creation and destruction of resources: We consider two cases of creation and destruction of resources: (a) robot independent that captures external processes, and (b) robot dependent that requires agent action.

Robot independent: Let us introduce the rates $C^+_{q,h} \geq 0$ and $C^-_{q,h} \geq 0$ of creation and destruction for all resources $h \in \mathcal{H}$ at each state $q \in \mathcal{Q}$, respectively. These constants define external, independent processes that govern resources in the environment. We adjust the resource flow dynamics in (3.28) to account for the net resource change $C^+_{q,h} - C^-_{q,h}$ per time step

$$y_{q,h,k} = \max\{\sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}((q',q))} + C^+_{q,h} - C^-_{q,h}, 0\},$$

where the $\max$ ensures resource amounts remain non-negative. The bound on variables $v_{e,h,k}$ becomes $\overline{v_{e,h,k}}^* = \overline{v_{e,h,k}} + k \cdot \sum_{q\in\mathcal{Q}} C^+_{q,h}$. Lastly, the RHS of (3.29) is set to the new $y_{q,g,k}$.

Robot dependent: Each robot class g has a rate of creation $C^+_{h,g} > 0$ and destruction $C^-_{h,g} > 0$ of resources $h \in \mathcal{H}$. We denote the sets of classes that can create and destroy resource h by $\mathcal{G}^+_h$ and $\mathcal{G}^-_h$, respectively. We modify the resource flow constraints (3.28) to

account for agents' resource creation and destruction work while stationary at states q

$$y_{q,h,k} = \max\{\sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}} + \sum_{g\in\mathcal{G}_h^+} C_{h,g}^+ u_{(q,q),g,k} - \sum_{g\in\mathcal{G}_h^-} C_{h,g}^- u_{(q,q),g,k}, 0\},$$

where, again, we use the $\max$ operator to disallow negative resource amounts. The bound of $v_{e,h,k}$ becomes $\overline{v_{e,h,k}}^* = \overline{v_{e,h,k}} + k \cdot |\mathcal{J}| \cdot \sum_{g\in\mathcal{G}_h^+} C_{h,g}^+$. As before, the RHS of (3.29) is set to the new expression of $y_{q,g,k}$.

Creation of resources on demand: Previously, we have explored situations where resources were either produced or depleted based on the environment or robot. However, resources may be generated only when and as much as needed for specific tasks. Consider $\iota_{q,k}^+ \in \mathbb{H}^+$ as a resource creation on demand variable at state $q \in \mathcal{Q}$ and at time $k \in [0..K]$, where $\mathbb{H}^+ = \mathbb{B}$ if the resource is binary, $\mathbb{H}^+ = [0..C_{q,h,k}^+]$ if it is indivisible, and $\mathbb{H}^+ = [0, C_{q,h,k}^+]$ if it is divisible. The updated resource flow constraint is

$$y_{q,h,k} = \sum_{(q',q)\in\mathcal{E}} v_{(q',q),h,k-\mathcal{W}} + \iota_{(q,q),h,k}^+,$$

To avoid the unrestricted creation of resources that may lead to trivial solutions and numerical issues, we incorporate a penalty for creating on-demand resources in the objective function. The modified optimization function is

$$J = \rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v - \gamma_c \sum_q \sum_k P_{q,k} \cdot \iota_{q,k}^+,$$

where $P_{q,k} > 0$ is the cost for creating a resource at state $q \in \mathcal{Q}$ at time $k \in [0..K]$, and $\gamma_c > 0$ is a scaling weight.

Gradual consumption of resources: In Sec. 3.3.7, we assumed that resources are consumed when the task starts. However, in some applications, this assumption may not hold.

For example, resources may be consumed as they are created, which can not be accommodated with advanced reservation of resources at the beginning of tasks' satisfaction. To address this, we modify the cross-consumption constraint to enable gradual resource consumption during tasks

$$C_{q,h,k} = \sum_{T \in \Lambda(q)} \sum_{\kappa=0}^{d_T} x_{\phi_T,(k-\kappa)} \frac{rs(h)}{d_T},$$

$$\sum_{\kappa=0}^{d_T} x_{\phi_T,(k-\kappa)} \leq 1, \ \forall k \in [0, K - d_T],$$

where d_T is the task's duration, and κ is an iterator index going from the beginning of the task to its end.

3.3.9 Analysis and Results

In this section, we examine several case studies that showcase the functionality and performance of the framework. First, a simple example illustrates the fundamental functionality of the MILP encoding for CaTL planning with resources. Following this, we explore four case studies highlighting performance differences between various resource types (divisible and indivisible) and transportation storage types (compartmental and uniform) under the same specification. Additionally, we provide examples that underscore how positive values of the objective function do not guarantee specification satisfaction and showcase some robustness properties for CaTL. Finally, we analyze runtime performance for an increasing number of agents and growing specification size.

All computations for the case studies were conducted on a computer featuring 20 cores at 3.7 GHz, with 64 GB of RAM. To solve the MILP, we utilized Gurobi [66]. For the MILP encoding of CaTL specifications we used PyTeLo [26] and ANTLRv4 [124], LOMAP [164] and networkx [67] for transition system models of environments.

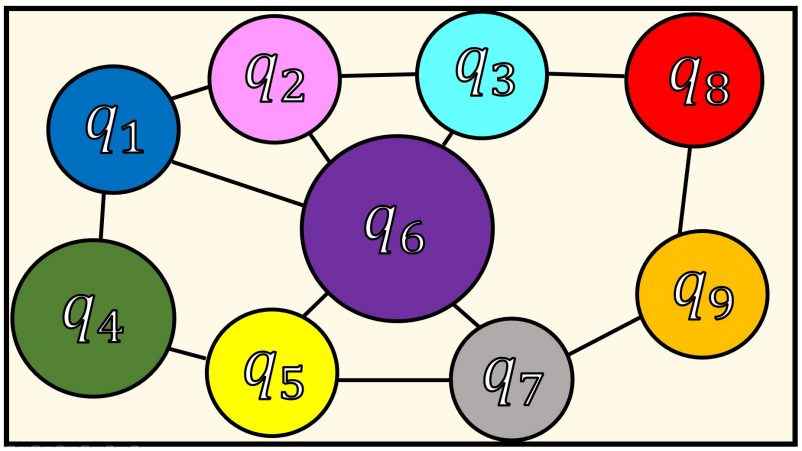


Figure 3.3.8: Abstracted labeled transition system from environment used for case studies.

Basic Example

Consider the small case study in Fig. 3.3.7. The environment has four regions with transitions between them of duration one. Two robots from each of the classes $c_1 = drilling$ and $c_2 = arm$ start at the state q_1. The scenario also includes two resources, $r_1 = bricks$ and $r_2 = wooden\,beams$, which are indivisible and initially distributed as $b_{\mathcal{H}}(0, \mathcal{Q}) = \{r_1 : \{q_2 : 4\}, r_2 : \{q_3 : 4\}\}$. The storage type is uniform, and the storage capacity $\Omega_g = 2$, $\forall g \in \mathcal{G}$. The mission specification is

$$\phi = \Diamond_{[3,4]}(T(1, \pi_{red}, (arm, 1), (brick, 3)) \wedge T(1, \pi_{red}, (drill, 1), (wooden\,beam, 3))).$$

In the given scenario, four robots travel to pick up resources from states q_2 and q_3 and then move towards q_4 to satisfy the specification. The trajectories of these robots are computed and displayed in Fig. 3.3.7. The resources are consumed upon the start of the tasks $k = 4$, and the task is fully satisfied after a one-time unit at $k = 5$. Both agents and resources have a robustness of one in this solution, as there is one additional agent and resource for each agent and resource class, respectively.

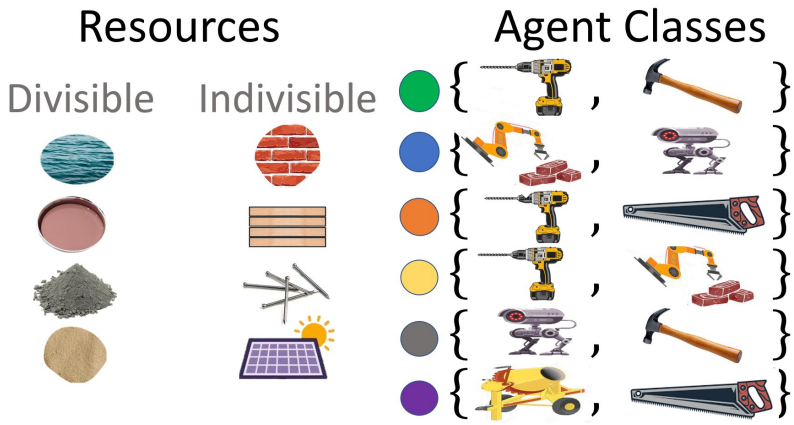


Figure 3.3.9: Resource types: divisible (water, paint, cement, sand) and indivisible (bricks, wooden beams, steel nails, solar panels). Colored circles on the right indicate a robot class with its corresponding set of capabilities.

Resource storage and types

In this section, we show the functionality and performance of various resources and storage types described in Sec. 3.3.3. In all cases, we consider a construction environment $Env = (\mathcal{Q}, \mathcal{E}, \mathcal{W})$ illustrated in Fig. 3.3.1. The environment is abstracted as the transition system shown in Fig. 3.3.8. The set $\mathcal{H}$ contains divisible resources such as water, paint, cement, and sand and indivisible resources such as bricks, wooden beams, steel nails, and solar panels. Fig. 3.3.9 shows agent classes with their corresponding set of capabilities, such as drilling, hammering, bricklaying, sawing, monitoring, and cement mixing. Lastly, transportation storage types (i.e., compartmental and uniform) are shown in Fig. 3.3.10. We design the construction mission to be feasible and given as a CaTL specification

$$\phi = \Diamond_{I_1} T_1 \wedge \Box_{I_2} T_2 \wedge \Diamond_{I_3} T_3 \wedge \Diamond_{I_4} T_4 \wedge \Box_{I_5} T_5 \wedge \Box_{I_6} T_6, \tag{3.48}$$

where $I_1 = [0, 5]$, $I_2 = [10, 12]$, $I_3 = [10, 14]$, $I_4 = [10, 14]$, $I_5 = [20, 22]$, $I_6 = [20, 22]$, tasks T_i with $i \in [1..5]$ are described in Table 3.4. The counting resource $rs(h)$ column and its amount are not explicitly defined since they vary depending on whether the resource is divisible or indivisible, so they are described later in each case study. We consider $|\mathcal{J}| = 18$

Table 3.4: List of tasks considered for case studies comparing resources and storage types.

Name	**Duration** (d)	**Region** $(\pi \in \mathcal{AP})$	**Counting proposition** $(cp(c))$	**Counting resource** $(rs(h))$
T_1	1	π_{Cyan}	$\{(drill, 2), (hammer, 2)\}$	$\{(r_1, x), (r_2, x)\}$
T_2	1	π_{Yellow}	$\{(mixer, 2), (sawing, 1)\}$	$\{(r_3, y), (r_2, x)\}$
T_3	1	π_{Purple}	$\{(drill, 2), (sawing, 1)\}$	$\{(r_2, x), (r_3, x)\}$
T_4	1	π_{Orange}	$\{(monitor, 3), (hammer, 2)\}$	$\{(r_4, y), (r_3, x)\}$
T_5	1	π_{Gray}	$\{(bricklaying, 1), (monitor, 2)\}$	$\{(r_1, y), (r_3, y)\}$
T_6	1	π_{Pink}	$\{(mixer, 2), (sawing, 1)\}$	$\{(r_1, y), (r_2, y)\}$

for the mission, with agent classes shown in Fig. 3.3.9 and the following quantities per class ● = 2, ● = 4, ● = 2, ● = 2, ● = 3, ● = 5, all with initial location q_1. With all this information, we define the initial agent distribution $s_{\mathcal{J}}(0, \mathcal{Q})$.

Case study 1 – compartmental storage with divisible resources We consider the abstracted Env shown in Fig. 3.3.8, mission specification (3.48), and initial agent distribution $s_{\mathcal{J}}(0, \mathcal{Q})$ described before. The initial resource distribution is

$$\begin{aligned} b_{\mathcal{H}}(0, \mathcal{Q}) = \ & \{r_1 : \{q_1 : 10, q_5 : 10\},\ r_2 : \{q_2 : 10, q_6 : 10\}, \\ & r_3 : \{q_3 : 10, q_7 : 10\},\ r_4 : \{q_4 : 10, q_8 : 10\}\}, \end{aligned} \tag{3.49}$$

where r_1, r_2, r_3, and r_4 correspond to *water, paint, cement,* and *sand*, respectively. The amount of resources required are $x = 1.4$ and $y = 0.7$ as indicated in Table 3.4. The storage capacities of each resource per agent class are

$$\begin{aligned} \Omega_{h,g} = \{ & ● : \{r_1 : 4.2, r_2 : 4.3, r_3 : 4.1, r_4 : 3.2\}, \\ & ● : \{r_1 : 2.1, r_2 : 2.2, r_3 : 2.3, r_4 : 2.4\}, \\ & ● : \{r_1 : 2.2, r_2 : 2.3, r_3 : 2.4, r_4 : 2.1\}, \\ & ● : \{r_1 : 2.1, r_2 : 2.2, r_3 : 2.3, r_4 : 2.2\}, \\ & ● : \{r_1 : 2.2, r_2 : 2.1, r_3 : 2.1, r_4 : 2.2\}, \\ & ● : \{r_1 : 2.3, r_2 : 2.2, r_3 : 2.1, r_4 : 2.3\}\}. \end{aligned}$$

The weights in (3.41), (3.43), and (3.44) are $\alpha_u = 0.2$, $\alpha_v = 0.2$, and $\gamma = 0.7$, respec-

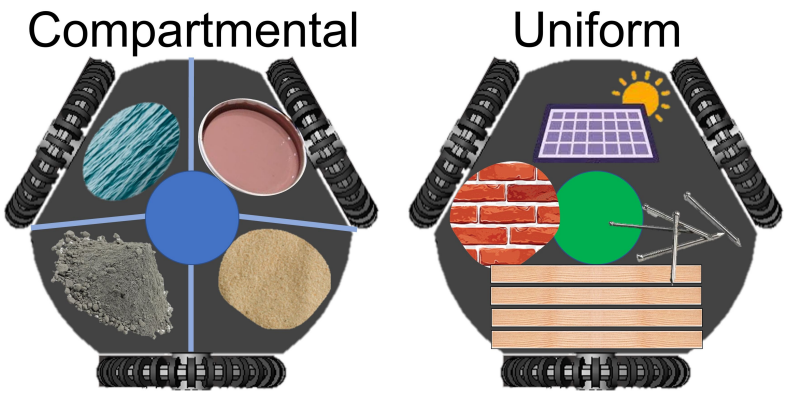


Figure 3.3.10: The transportation type examined in case studies can be compartmental (left) or uniform (right). In the former, each resource has its own compartment, while in the latter, all resources share the same storage capacity.

tively. With these weights, the objective prioritizes agent robustness, followed by resource robustness. Although the normalized travel time for agents and resources are also objectives, they have less priority. The time horizon of the given specification is $\|\phi\| = K = 22$(s).

The mission specification is feasible; see Sec. 3.3.9 for an infeasible case. Upon computing the solution to the planning problem, the availability robustness for agents and resources are $\rho_a = 3$ and $\rho_h = 5.89$. The normalized travel time for agents and resources of $\tau_u = 0.18$, and $\tau_v = 0.25$, indicating there was more flow of resources in the environment than agents. Note that both agent and resource robustness classes are positive. Thus, the satisfaction of the mission is guaranteed by Thm. 3.3.1. Moreover, the overall optimization value is $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 3 + 0.7 \cdot 5.89 - 0.2 \cdot 0.18 - 0.2 \cdot 0.25 = 7.03$.

Case study 2 – compartmental storage with indivisible resources Since we consider indivisible resources, see Fig. 3.3.9, resource decision variables are integer, rather than real as in the previous case. The mission specification is defined by equation (3.48), where r_1 = bricks, r_2 = wooden beams, r_3 = nails, and r_4 = solar panels. For tasks in Table 3.4, we set $x = 1$ and $y = 2$. The initial resource distribution $b_{\mathcal{H}}(0, \mathcal{Q})$ is the same (3.49). The resource capacities per agent class are

$$\Omega_{h,g} = \{\bullet : \{r_1 : 4, r_2 : 4, r_3 : 4, r_4 : 3\}, \quad \bullet : \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\},$$
$$\bullet : \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\}, \quad \bullet : \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\},$$
$$\bullet : \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\}, \quad \bullet : \{r_1 : 2, r_2 : 2, r_3 : 2, r_4 : 2\}\}.$$

As both robustness classes consider integer predicates, we have integer scores of availability for agents $\rho_a = 3$ and resources $\rho_h = 5$. Thm. 3.3.1 guarantees the mission's satisfaction, as both robustness classes are positive. Thus, the overall robustness is positive. The normalized travel time for agents and resources are $\tau_u = 0.18$ and $\tau_v = 0.26$, respectively, indicating a greater resource flow in the environment. Interestingly, the values for both cases are close, which suggests that the trajectories taken by agents to satisfy the specification are similar or identical. The overall objective value is $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 3 + 0.7 \cdot 5 - 0.2 \cdot 0.18 - 0.2 \cdot 0.26 = 6.41$.

Case study 3 – uniform resource storage with divisible resource type For this case study, we consider $s_{\mathcal{J}}(0, \mathcal{Q})$, $b_{\mathcal{H}}(0, \mathcal{Q})$, resources ($r_1$, r_2, r_3, and r_4), environment Env, and mission specification ϕ as in case study 1. However, the storage type for agents is uniform. The storage capacity for each agent class is

$$\Omega_g = \{\bullet : 8.2, \bullet : 6.1, \bullet : 6.3, \bullet : 5.4, \bullet : 4.5, \bullet : 5.6\}.$$

The robustness values for agents' and resource availability are $\rho_a = 3$ and $\rho_h = 5.89$, respectively. The latter value is not an integer, as resources are considered divisible. Since both are positive, the mission is satisfied via Thm. 3.3.1 The normalized agent and resource total travel times are $\tau_u = 0.17$ and $\tau_v = 0.24$, respectively. The overall objective value is $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 3 + 0.7 \cdot 5.89 - 0.2 \cdot 0.17 - 0.2 \cdot 0.24 = 7.04$.

Case study 4 – uniform resource storage with indivisible resource type Let us consider $s_{\mathcal{J}}(0, \mathcal{Q})$, $b_{\mathcal{H}}(0, \mathcal{Q})$, resources ($r_1$, r_2, r_3, and r_4), environment Env, and mission specification ϕ identical to case study 2. The transportation storage type is uniform (right side of Fig. 3.3.10). The storage class capacity per agent class is

$$\Omega_g = \{ : 8, : 6, : 6, : 5, : 4, : 5\}.$$

The robustness values for the availability of agents and resources are $\rho_a = 3$ and $\rho_h = 5$. The normalized travel time of agents and resources are $\tau_u = 0.14$ and $\tau_v = 0.25$. Note that the travel of agents score is slightly lower than in the previous case since using a uniform storage type provides a higher capacity to transport a specific resource, requiring fewer agents to move in the environment. The overall objective value is $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 1 + 0.7 \cdot 8 + 0.3 \cdot 0.918 + 0.3 \cdot 3.301 = 7.8657$ Similar to previous cases, both robustness class scores are positive, ensuring the satisfaction of the mission specification.

Time performance comparison: Table 3.5 contains the information on the time performance, objective value, and continuous, integer, and binary variables of the four case studies. All of the objective values are similar. However, their time performance differs due to the varying number of variables required to capture the problem in a MILP. Although continuous variables can affect the time performance, it is usually mainly influenced by the integer and binary variables. Notably, solving the fourth case study with more integer and binary variables takes the longest. As four cases work with the same specification, number of agents, and resources, it is plausible to conclude that a problem with the compartmental storage type is faster than the uniform storage type and that divisible resources yield better runtimes than indivisible resources. However, the quantity of integer and binary variables depends on the specifications, environment size, number of agent classes, number of resource types, and storage type. For further examples of increasing specification complexity

and the number of agents, see Sec. 3.3.9.

Table 3.5: Number of continuous, integer, and binary variables, runtime, and objective values for the four case studies.

Case study	Time	Objective Values	Continuous Vars.	Integer Vars.	Binary Vars.
1	12.05(s)	7.03	2859	5281	115
2	26.39(s)	6.41	64	8083	119
3	28.86(s)	7.04	2859	5274	115
4	45.18(s)	7.86	68	8375	248

Robustness and satisfaction comparison for CaTL

Here, we delve into a discussion on how robustness scores of both the agents and resources impact the fulfillment of the mission specification as a whole. To this end, we use the setup from case study 1, wherein we modify task T_6 so that instead of requesting two mixers, we now require six units. The solution for this case study differs slightly from the one in case study 1. The main difference is that in this case, the agent availability robustness score is $\rho_a = -1$, indicating that we do not have enough agents with the necessary capabilities to complete the mission. However, the overall objective function takes a value of $\rho_a + \gamma\rho_h - \gamma_u\tau_u - \gamma_v\tau_v = 3.037$, which does not guarantee the satisfaction of the mission even though it is positive. Instead, to guarantee satisfaction, we make use of Cor. 3.3.1, which implies that if $\rho_a(s_{\mathcal{J}}, \phi, 0) > 0 \wedge \rho_h(b_{\mathcal{H}}, \phi, 0) > 0 \Rightarrow \rho((s_{\mathcal{J}}, b_{\mathcal{H}}), \phi, 0) > 0$. Hence, using Thm. 2.1.1, we can infer that mission satisfaction is guaranteed only if both the availability robustness of agents and resources are positive.

Although we may encounter situations where the specifications suggest an absence of agents, our MILP encoding still produces a solution that violates the specification to a minimum extent. In the case of the resources, this is impossible since, by definition, the resource variables in Sec. 3.3.7 are defined as positive variables. The rationale behind this approach is to simplify the establishment of cross-consumption and resource dynamics constraints. In situations with a shortage of resources, the problem becomes infeasible by

definition, which could be addressed by introducing a slack variable. However, this would necessitate more complex bookkeeping. All things considered, satisfaction with the CaTL mission relies on the availability of agents to fulfill specifications and feasible trajectories for agents to pick and drop resources.

Runtime performance comparison

This section presents a runtime performance comparison of four different types of problems, namely, compartmental storage with divisible and indivisible resources, uniform storage with divisible and indivisible resources, and the CaTL baseline [95]. The evaluation consists of two experiments: one increases the number of agents, and the other grows the specification size. For the first case, we consider the Env shown in Fig. 3.3.1. We use the same number of robot classes and divisible resource types shown in Fig. 3.3.9, compartmental resource storage type, and storage capacities defined for case study 1. The mission specification considered is $\phi = \Diamond_{[0,4]} T_1 \wedge \Box_{[18,22]}(T_3 \wedge T_4)$, where T_1, T_3 and T_4 are given in Table 3.4. Resource distribution is changed such that we have 2 units of each resource in q_2 and q_4. Five agents are added per iteration with random robot classes, and all start at state q_1. The results are presented in Table 3.6. Note that increasing the number of agents reduces the time for the four types of problems. Most of them compute a solution in around 0.7 seconds, which is still slightly longer than the CaTL baseline, which is expected since the latter does not plan for resources.

In the second scenario, we use the same initial conditions. However, rather than increasing the number of agents, we spawn 18 agents that are randomly generated and proceed to augment the specification in the following manner

$$\phi = \bigwedge_{n=1}^{N} \mathcal{X}_{I_n} \mathfrak{T},$$

where N is increased by two in every step. The temporal operators $\mathcal{X} \in \{\Box, \Diamond\}$ are chosen

Table 3.6: Runtime performance comparison for an incrementing number of agents.

Agents	**Compartmental**		**Uniform**		**CaTL Baseline** [75]
	Divisible	**Indivisible**	**Divisible**	**Indivisible**	(Resources not considered)
10	1.32 (s)	2.73 (s)	0.96 (s)	1.12 (s)	0.22 (s)
15	2.18 (s)	2.01 (s)	1.90 (s)	3.09 (s)	0.35 (s)
20	0.96 (s)	2.26 (s)	1.18 (s)	0.68 (s)	0.33 (s)
25	0.76 (s)	1.87 (s)	0.71 (s)	0.76 (s)	0.33 (s)
30	0.79 (s)	0.84 (s)	0.73 (s)	0.70 (s)	0.32 (s)

Table 3.7: Runtime performance comparison for an incrementing specification ϕ.

ϕ	**Compartmental**		**Uniform**		**CaTL Baseline** [75]
	Divisible	**Indivisible**	**Divisible**	**Indivisible**	(Resources not considered)
2	0.90 (s)	0.79 (s)	1.02 (s)	0.70 (s)	0.26 (s)
4	1.87 (s)	1.75 (s)	3.79 (s)	1.50 (s)	0.35 (s)
6	3.13 (s)	7.06 (s)	2.47 (s)	8.35 (s)	0.63 (s)
8	4.12 (s)	9.23 (s)	17.50 (s)	12.38 (s)	0.68 (s)
10	12.32 (s)	9.39 (s)	17.96 (s)	61.11 (s)	1.17 (s)

randomly with time intervals $I_n = [5(n-1), 5(n-1) + \text{rand}(1,5)]$. Tasks are randomly taken from Table 3.4 such that $\mathfrak{T} \in \{T_1, \ldots, T_6\}$. Table 3.7 shows the computation time for a solution when growing the specification size for the four cases. Note that the case of uniform storage and indivisible resources type takes the longest. This is mainly because finding the optimal combination of resources that each robot should carry to maximize the robustness scores becomes a combinatorial problem when considering integer values of resources. In general, computing a solution for compartmental is faster than uniform storage types, and divisible resources are faster than indivisible.

3.3.10 Conclusions and Future Work

This section presents an extension of CaTL that considers agent capabilities and resource constraints. The extension is achieved by enhancing the MILP encoding to capture different types of resources, including divisible and indivisible, and accommodating different agent storage types for resources, such as compartmental and uniform. Moreover, an encoding is

developed to extend STL to accept multiple predicate classes and compute robustness separately. The paper demonstrates the effectiveness of computing the multi-robustness score for an STL specification with multiple predicate classes and explores how satisfaction is related to the robustness scores. The proposed framework is applied to CaTL to capture agent and resource robustness scores. When these scores are positive, they guarantee satisfaction with the mission specification. Furthermore, the paper comprehensively evaluates the framework's runtime performance when considering uniform or compartmental storage types with divisible or indivisible resources. In future work, we will consider performing mission specifications on real robots and explore uncertainty in the availability of resources and agents in the environment.

Chapter 4

Handling Infeasibility in Temporal Logic: Partial Satisfaction of Specifications

4.1 Partial Satisfaction in Control Synthesis from Temporal Logic Specifications

4.1.1 Introduction

In recent years, mission specifications for describing system behaviors have increasingly been expressed using formal languages such as Signal Temporal Logic (STL) [108, 133]. These formalisms enable a precise representation of complex temporal, spatial, and logical constraints, which aids in synthesizing controllers and planners that ensure the correct operation of systems [8, 89]. Existing approaches for satisfying these specifications typically fall into two categories: automata-based methods and optimization-based methods. Automata-based approaches construct product automata to capture all possible task executions and identify optimal solutions [6, 15]. However, their applicability is often limited

due to the exponential growth of the state space, making them computationally infeasible for large-scale systems. In contrast, optimization-based methods, particularly those that utilize Mixed-Integer Linear Programming (MILP), offer better scalability [150, 133, 130]. The development of advanced off-the-shelf solvers has significantly enhanced the ability to handle large numbers of variables, making MILP-based approaches increasingly viable for complex mission planning and control tasks [66].

Traditional Signal Temporal Logic (STL) provides a robust quantitative measure for assessing how well a signal meets or violates a specification [108]. However, in its standard form, STL treats all subformulas with equal importance and lacks a mechanism for prioritizing certain tasks over others. Extensions like Weighted Signal Temporal Logic (wSTL) have been developed to address this limitation, where weights reflect user-defined preferences and priorities [111]. Building on this concept, we introduce an extended formalism called wSTL+ that includes both inclusive (soft) and exclusive (hard) operators[32]. This enhancement allows for a clearer expression of scenarios in which some aspects of a mission must be fully enforced or completely disregarded. For instance, in cases where multiple objectives conflict, exclusive conjunction and exclusive always ensure that all relevant subformulas are either entirely satisfied or ignored, preventing partial compromises. Similarly, exclusive disjunction and exclusive eventually enforce that only one subformula holds at a specific point in time, making them well-suited for scenarios involving mutually exclusive choices. For instance, when a quadrotor is required to monitor two regions simultaneously. Standard STL or even wSTL could lead to a compromise solution that does not adequately differentiate which region is more critical. In contrast, wSTL+ enables designers to enforce that one region must be fully monitored (exclusive satisfaction) while the other can be addressed with a softer preference (inclusive satisfaction). This approach guides the system in prioritizing the more important objective.

While achieving satisfaction of all mission requirements is ideal, real-world constraints often make this infeasible. Factors such as conflicting sub-tasks, resource limitations, con-

strained configuration spaces, and control saturation can force the system to compromise, rendering a simple binary notion of satisfaction insufficient. Although STL and wSTL provide robustness measures that identify minimally violated solutions in the presence of conflicting or infeasible specifications, they cannot intentionally allocate partial satisfaction across competing subformulae. This limitation has driven the development of partial satisfaction (PS) frameworks [30, 32], which allow a system to fulfill as much of the mission as possible while prioritizing critical objectives. For example, consider a team of robots assigned to monitor multiple regions for surveillance while avoiding obstacles. If resource limitations prevent full coverage of all areas, a PS framework ensures that the most crucial locations, such as those with higher security risks, are fully monitored, while lower-priority areas receive only partial attention. By explicitly reasoning about trade-offs in mission satisfaction, PS frameworks provide a structured and interpretable way to handle infeasible or conflicting specifications, ensuring that essential tasks are met while optimizing overall mission success.

Our work enhances the traditional satisfaction metrics of STL (Signal Temporal Logic) and wSTL (Weighted STL) by quantifying the satisfaction of a subformula as a fraction rather than a simple binary decision. This nuanced metric is especially useful when specifications are inherently conflicting, such as when a mission requires mutually exclusive behaviors to occur within the same time interval. We propose two different approaches to define priorities within a specification. The first approach involves assigning weights that adjust the importance of Boolean and temporal operators. The second approach incorporates a preference structure that reflects the depth of subformulae in the specification's Abstract Syntax Tree (AST). Consequently, we formulate the control synthesis problem as a bilevel optimization problem: the inner Mixed Integer Linear Program (MILP) aims to maximize the number of satisfied subformulae (weighted according to their priority), while the outer Linear Program (LP) focuses on maximizing the robustness of these subformulae. [33] [32]

4.1.2 Weighted Signal Temporal Logic (wSTL+)

Here, we describe the semantics of Weighted Signal Temporal Logic (wSTL+ [32]), an extension of wSTL [111] that allows specifications to capture preferences, priorities, and importance inclusively and exclusively through the Boolean and temporal operators.

Definition 4.1.1 (Weighted Signal Temporal Logic (wSTL+))**.** *The syntax of wSTL+ in Backus-Naur form over linear predicates is*

$$\begin{aligned}\varphi ::= &\top \mid \bot \mid \mu \mid \neg\varphi \mid \bigwedge_{i\in[1..N]}{}^{p}\varphi_i \mid \bigvee_{i\in[1..N]}{}^{p}\varphi_i \mid \Diamond_I^w\varphi \mid \Box_I^w\varphi \mid \dot{\Diamond}_I^w\,\varphi \mid \\ &\dot{\bigvee}_{i\in[1..N]}{}^{p}\varphi_i \mid \dot{\bigwedge}_{i\in[1..N]}{}^{p}\varphi_i \mid \boxdot_I^w\varphi \mid \varphi_1\mathcal{U}_I^w\varphi_2 \mid \varphi_1\dot{\mathcal{U}}_I^w\varphi_2,\end{aligned} \tag{4.1}$$

where $\top$ and $\bot$ are the logical *True* and *False*; μ is a linear *predicate* of the form $s_i \geq \pi$ with threshold π over the i-th component of a discrete-time signal with values in the compact set $\mathbb{M} \subseteq \mathbb{R}^N$ such that $s : \mathbb{Z}_{\geq 0} \to \mathbb{M}$; $\neg$, $\wedge$, and $\vee$ are the Boolean *negation*, *inclusive conjunction* and *inclusive disjunction* operators, respectively. $\Diamond$ (*inclusive eventually*) and $\Box$ (*inclusive always*) are temporal operators with time bound in the range I with the same definitions and semantics as in STL [108]. Weight functions assign the positive weights over inclusive Boolean operators' conjunction and disjunction formulae $p : [1..N] \to \mathbb{R}_{>0}$, where N is the number of sub-formulae under the operator. For sub-formulae in inclusive conjunction, denoted as $(\wedge^p(\varphi_1, \varphi_2, \ldots, \varphi_N))$, the weights capture the importance of parallel tasks, whereas for the sub-formulae in inclusive disjunction, denoted as $(\vee^p(\varphi_1, \varphi_2, \ldots, \varphi_N))$, the weights indicate priorities for alternatives. The positive weight functions $w : I \to \mathbb{R}_{>0}$ capture user preferences for satisfaction times for the *inclusive eventually* operator, the importance of satisfaction times in the case of the *inclusive always* operator, and highlight the most preferred times at which φ_2 is satisfied for the *inclusive until* operator. For exclusive operators, consider a signal s at time k as $s(k)$. To simplify, let p and w be equal to one for all operators. Then $\dot{\wedge}_i^p\varphi_i$ with $i \in [1..N]$ is the *exclusive conjunction*, implying that a signal $s(k)$ has to satisfy every subformula φ_i or not at all. Formally, $s(k) \vDash \varphi \equiv$

$(s(k) \vDash \varphi_i, \forall i \in [1..N]) \vee (s(k) \nvDash \varphi_i,\ \forall i \in [1..N])$. $\veebar_i^p$ with $i \in [1..N]$ is the *exclusive disjunction*, capturing that a signal $s(k)$ has to satisfy one subformula φ_i and not any other subformulae (mutually exclusive satisfaction). Formally, $s(k) \vDash \varphi \equiv\ s(k) \vDash \varphi_i \wedge s(k) \nvDash \varphi_j$, such that $\exists i \in [1..N]\ \wedge \forall j \in [1..N], i \neq j$. $\dot{\Diamond}_I^w\ \varphi$ is the *exclusive eventually* that captures that a formula φ has to be satisfied at only one-time step and not anymore during the time interval I, formally, $s(k) \vDash \varphi \equiv\ s(k') \vDash \varphi \wedge s(k'') \nvDash \varphi, k' \neq k'', \forall k', k'' \in I$. $\dot{\boxdot}_I^w \varphi$ is the *exclusive always* capturing that formula φ has to be satisfied during the whole interval or not at all. Formally $s(k) \vDash \varphi \equiv\ (s(k') \vDash \varphi, \forall k' \in I) \vee (s(k') \nvDash \varphi,\ \forall k' \in I)$. Finally, $\varphi_1 \dot{\mathcal{U}}_I^w \varphi_2$ is the *exclusive until* operator captures that Capturing that φ_2 becomes $\top$ at exactly a one-time point within interval $k + I$ and φ_1 holds continuously until that time. Formally, $s(k) \vDash \varphi \equiv s(k^*) \vDash \varphi_2 \wedge s(k'') \vDash \varphi_1 \wedge s(k') \nvDash \varphi_2,\ \exists k^* \in k + I,\ \forall k' \in k + I,\ k' \neq k^*,\ k'' \in [k..k']$.

Remark 4.1.1. *A wSTL+ specification φ with only inclusive operators is equivalent to a wSTL specification. And if, in addition, all weights p and w are equal to one, φ is equivalent to an STL specification ϕ*[1].

Definition 4.1.2 (Weighted Traditional Robustness [111, 24])**.** *Given a wSTL+ specification φ and a signal s, the weighted robustness score $\tilde{\rho}(\varphi, s, k)$ at time k is recursively*

[1]From now on, we will refer to an STL specification as ϕ and a wSTL+ specification as φ

defined as follows

$$
\begin{aligned}
\tilde{\rho}(\mu, s, k) &:= s_i(k) - \pi, \\
\tilde{\rho}\left(\neg\varphi, s, k\right) &:= -\tilde{\rho}(\varphi, s, k), \\
\tilde{\rho}\left(\bigwedge_i^p \varphi_i, s, k\right) &:= \min_i \left\{p_i^{\wedge} \cdot \tilde{\rho}(\varphi_i, s, k)\right\}, \\
\tilde{\rho}\left(\dot{\bigwedge_i}^p \varphi_i, s, k\right) &:= \max\left\{\min_i \left\{p_i^{\wedge} \cdot \tilde{\rho}(\varphi_i, s, k)\right\}, \right. \\
&\qquad \left. \min_i \left\{p_i^{\wedge} \cdot \tilde{\rho}(\neg\varphi_i, s, k)\right\}\right\}, \\
\tilde{\rho}\left(\bigvee_i^p \varphi_i, s, k\right) &:= \max_i \left\{p_i^{\vee} \cdot \tilde{\rho}(\varphi_i, s, k)\right\}, \\
\tilde{\rho}\left(\dot{\bigvee_i}^p \varphi_i, s, k\right) &:= \min\left\{\max_i \left\{p_i^{\vee} \cdot \tilde{\rho}(\varphi_i, s, k)\right\}, \right. \\
&\qquad \left. \max_i \left\{p_i^{\vee} \cdot \tilde{\rho}(\neg\varphi_i, s, k)\right\}\right\}, \\
\tilde{\rho}\left(\Diamond_I^w \varphi, s, k\right) &:= \max_{k' \in k+I} \left\{w^{\Diamond}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}, \\
\tilde{\rho}\left(\dot{\Diamond}_I^w\ \varphi, s, k\right) &:= \min\left\{\max_{k' \in k+I} \left\{w^{\Diamond}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}, \right. \\
&\qquad \left. \max_{k' \in k+I} \left\{w^{\Diamond}(k'-k) \cdot \tilde{\rho}(\neg\varphi, s, k')\right\}\right\}, \\
\tilde{\rho}\left(\Box_I^w \varphi, s, k\right) &:= \min_{k' \in k+I} \left\{w^{\Box}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}, \\
\tilde{\rho}\left(\boxdot_I^w \varphi, s, k\right) &:= \max\left\{\min_{k' \in k+I} \left\{w^{\Box}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}, \right. \\
&\qquad \left. \min_{k' \in k+I} \left\{w^{\Box}(k'-k) \cdot \tilde{\rho}(\varphi, s, k')\right\}\right\}, \\
\tilde{\rho}\left(\varphi_1 \mathcal{U}_I^w \varphi_2, s, k\right) &:= \max_{k' \in k+I} \left\{\min\left\{w^{\mathcal{U}}(k'-k) \cdot \tilde{\rho}(\varphi_2, s, k'), \right.\right. \\
&\qquad \left.\left. \min_{k'' \in I'} \left\{\tilde{\rho}(\varphi_1, s, k'')\right\}\right\}\right\}, \\
\tilde{\rho}\left(\varphi_1 \dot{\mathcal{U}}_I^w \varphi_2, s, k\right) &:= \max_{k' \in k+I} \left\{\min\left\{w^{\mathcal{U}}(k'-k) \cdot \tilde{\rho}(\dot{\Diamond}_I^w\ \varphi_2, s, k'), \right.\right. \\
&\qquad \left.\left. \min_{k'' \in I'} \left\{\tilde{\rho}(\varphi_1, s, k'')\right\}\right\}\right\},
\end{aligned}
\tag{4.2}
$$

where $I' = [k..k']$ and for Boolean operators, we have

$$p_i^{\wedge}(r_i) = \left(\frac{1}{2} - \bar{p}_i\right)\mathrm{sign}(r_i) + \frac{1}{2}, \qquad p_i^{\vee}(r_i) = 1 - p_i^{\wedge}(r_i),$$

where $r_i = \tilde{\rho}(\varphi_i, s, k)$ is the weighted robustness of the subformula φ_i, and $\bar{p}_i = \frac{p_i}{\mathbf{1}^T p}$ is its normalized weight. Similarly, for the unary temporal operators, we have

$$w^{\Box}(r_{k'}) = \left(\frac{1}{2} - \bar{w}(k - k')\right)\mathrm{sign}(r_{k'}) + \frac{1}{2},\ w^{\Diamond}(r_{k'}) = 1 - w^{\Box}(r_{k'}),$$

where $r_{k'} = \tilde{\rho}(\varphi, s, k')$, $\bar{w}(\mathtt{t}) = \frac{w(\mathtt{t})}{W}$ and $W = \sum_{\mathtt{t}=0}^{k'-k} w(\mathtt{t})$. For the binary temporal operator, until we define

$$w^{\mathcal{U}}(r_{k'}) = w^{\Diamond}(r_{k'}),$$

where $r_{k'} = \tilde{\rho}(\varphi_2, s, k')$. The weights $p^{\wedge}$, $p^{\vee}$, $w^{\Box}$, and $w^{\Diamond}$ are defined as DeMorgan aggregation functions [111].

The *robustness* (quantitative semantics) of a wSTL+ specification φ captures the margin of satisfaction or violation of a signal s over φ modulated by the specified weights for all inclusive operators. In the case of exclusive (consensus) operators, *robustness* indicates how far the signal is from disrupting consensus among the sub-formulae and, therefore, violating or satisfying the exclusive operator.

Another important property of STL is soundness, which is also inherited by wSTL+ and is defined as follows.

Theorem 4.1.1 (wSTL+ Soundness [111])**.** *The robustness score of wSTL+ $\tilde{\rho}(\varphi, s, k)$ is sound iff*

$$\tilde{\rho}(\varphi, s, k) > 0 \iff \rho(\phi, s, k) > 0 \to s \vDash \varphi,$$
$$\tilde{\rho}(\varphi, s, k) < 0 \iff \rho(\phi, s, k) < 0 \to s \nvDash \varphi,$$

where $\rho(\phi, s, k)$[2] is the robustness of the equivalent STL specification.

Note that the robustness score of a wSTL+ specification φ is positive if and only if the equivalent STL specification ϕ is also positive. Therefore, the signal s satisfies both specifications.

A wSTL+ or STL formula is said to be in *positive normal form* (PNF) [133] if it satisfies two conditions. First, all its predicates are of the $s_i \geq \mu$ form. Second, it does not contain the negation operator.

Any STL or wSTL+ formula can be represented as an Abstract Syntax Tree, where internal nodes capture the logical and temporal operators, and leaves represent atomic predicates, defined as follows.

Definition 4.1.3 (Abstract Syntax Tree (AST) [71])**.** *An* abstract syntax tree *for a STL formula ϕ is a tuple $\mathcal{A}^{\varphi} = (\mathcal{V}, \mathcal{E}, \ell)$ where i) $\mathcal{V}$ is a finite set of nodes; ii) $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of directed edges, so that if $(v, v') \in \mathcal{E}$, then v' is considered a child of v; iii) $\ell : \mathcal{V} \to \Sigma$ is a labeling function, with Σ defined as $\Sigma := \mathcal{P} \cup \mathcal{L} \cup \mathcal{T}$, where $\mathcal{P}$ is the set of atomic predicates,$\mathcal{L}$ is the set of Boolean operators, $\mathcal{T}$ is the set of temporal operators. In the case of a wSTL+ formula, each edge $e \in \mathcal{E}$ is further annotated with a positive weight.*

We say φ' is a *proper subformula* of φ if for a node $v \in V$ representing φ' we write $\varphi' \sqsubset \varphi$ if and only if v lies on the unique path from the root $r \in V$ (which represents φ) to v and $v \neq r$. We further define the reflexive closure of $\sqsubset$ by writing $\varphi' \sqsubseteq \varphi \iff (v = r) \vee (\varphi' \sqsubset \varphi)$.

Definition 4.1.4 (Depth of a Subformula)**.** *Given an AST $\mathcal{A}^{\varphi} = (\mathcal{V}, \mathcal{E}, \ell)$ abstracting wSTL+ formula φ with root node $r \in \mathcal{V}$, the* depth *of a node $v \in \mathcal{V}$, denoted $depth_{\varphi}(v)$, is defined as the length of the unique path from r to v. If φ' captured by node v' is not a subformula of φ then $depth_{\varphi}(v') = \infty$.*

Definition 4.1.5 (Partial Order over Signals with Respect to φ)**.** *Let s and s' be two signals and let us define the set of satisfied subformulae-time pairs as $\Phi_d^{\varphi}(s) = \{(\varphi', k) \mid$*

[2]Through the paper we use $\bar{\rho}$ and ρ to refer to wSTL+ and STL robustness, respectively.

depth$_\varphi(\varphi') = d, (s,k) \vDash \varphi'\}$, *We say that* s *is* less than s' *with respect to* φ*, denoted* $s \prec_\varphi s'$*, if there exists a depth* $d \geq 0$ *such that:*

1. $|\Phi_d^\varphi(s)| < |\Phi_d^\varphi(s')|$*, and*

2. *for all depths* $d' < d$*, it holds that* $|\Phi_{d'}^\phi(s)| = |\Phi_{d'}^\phi(s')|$.

This partial order is used in our partial satisfaction framework to compare signals based on the number of lower-depth subformulae that they satisfy.

Similarities and limitations of STL, wSTL, and wSTL+.

Here, we discuss the connections and limitations of STL, wSTL, and wSTL+. STL (Signal Temporal Logic) introduced in [108] is a powerful language that allows users to reason about continuous signals while adhering to logical and temporal constraints. It enables the incorporation of multiple signal components and the computation of both qualitative and quantitative semantics. Qualitative semantics assess whether operators are satisfied or violated, while quantitative semantics computes a margin of satisfaction. STL also has soundness properties regarding its quantitative semantics, establishing a link between an STL formula's satisfaction and its robustness score greater than zero. It is important to note that all options available for satisfying an STL formula hold the same significance for both Boolean and temporal operators. Even though operators like disjunction and eventually explicitly offer choices, a unique solution is typically found by identifying the choice that maximizes robustness. However, this approach may not be ideal when users have feasible preferred choices. To address this, the authors in [111] proposed wSTL, which incorporates weights on Boolean and temporal operators to adjust robustness and reflect user preferences and time importance. When all considered weights are positive, the soundness property remains intact.

In [32], we introduced wSTL+, a syntactic sugar extension of wSTL that includes exclusive operators. For conjunctive operators (such as conjunction and always), these exclusive

operators allow for either full satisfaction or none at all. In the case of disjunctive operators (like disjunction and eventually), they impose a mutually exclusive constraint.

Robustness Degree and Perturbation The robustness degree [57] of an STL formula $\rho(\phi, s, k)$ represents the maximal perturbation that s can endure without altering the Boolean truth value of ϕ. However, this property does not hold in the same way for wSTL+, which scales the robustness of STL. To clarify this, we will first define the robustness degree for STL and show how wSTL+ handles the robustness degree.

Definition 4.1.6 (STL Robustness Degree)**.** *Let ϕ be an STL formula and s be a signal. The* robustness degree $\rho(\phi, s, k)$ *of ϕ at time $k \in \|\phi\|$ is defined recursively and has the following property:* $\forall\, s, s', \quad$ *if* $0 \leq \|s - s'\|_\infty \leq |\rho(\phi, s, k)|$, *then,* $s \vDash \phi \implies s' \vDash \phi$, *and* $s \nvDash \phi \implies s' \nvDash \phi$, *and* $\operatorname{sign}(\rho(\phi, s, k)) = sign(\rho(\phi, s', k))$.

Note that s' captures the maximal perturbation that the signal s can have without changing its Boolean truth value.

Let us define a function that considers the maximal scaling factor that the wSTL+ formula can perform to its equivalent STL formula.

Definition 4.1.7 (Maximal Scaling Factor in wSTL+)**.** *Let φ be a wSTL+ formula, we define the maximal scaling factor $\bar{W}_\varphi$ recursively by:*

$$\bar{W}_\varphi = \begin{cases} \max\limits_{i \in [1..N]} \{p_i \cdot \bar{W}_{\varphi_i}\}, & \text{if } \varphi \in \{\wedge^p \varphi_i, \barwedge^p \varphi_i, \vee^p \varphi_i, \barvee^p \varphi_i\}, \\ \max\limits_{k \in I} \{w(k) \cdot \bar{W}_{\varphi'}\}, & \text{if } \varphi \in \{\Diamond_I \varphi', \Box_I \varphi', \barDiamond_I\, \varphi', \barBox_I \varphi'\}, \\ \max\limits_{k \in I} \{w(k) \cdot \bar{W}_{\varphi_2}\}, & \text{if } \varphi \in \{\phi_1 \mathcal{U}_I^w \phi_2, \phi_1 \dot{\mathcal{U}}_I^w \phi_2\}, \\ 1, & \text{if } \varphi \text{ is an atomic predicate } \mu. \end{cases} \tag{4.3}$$

Intuitively, $\bar{W}_\varphi$ represents the maximum factor by which the original STL robustness is scaled in the corresponding wSTL+ formula.

Note that as every Boolean weight $p \in \mathbb{R}_{>0}$ or temporal weight $w \in \mathbb{R}_{>0}$ are positive therefore the recursion of $\bar{W}_\phi \in \mathbb{R}_{>0}$ is also positive.

Proposition 4.1.1 (Scaling of wSTL+ Robustness)**.** *Let φ be a wSTL+ formula and consider its corresponding STL formula ϕ. Then, for every time $k \in \|\varphi\|$ and every signal s,*

$$\bar{\rho}(\varphi, s, k) \leq \bar{W}_\varphi \cdot \rho(\phi, s, k).$$

In particular, if $\bar{W}_\varphi > 1$, then the wSTL+ robustness does not represent the maximal perturbation that s can bear without changing the Boolean truth value of φ.

Proof scketch. We prove the claim by structural induction on φ. *Base Case:* For an atomic predicate μ, we have $\rho(\mu, s, k) = s_i(k) - \mu$ and, by definition, $\bar{W}_\mu = 1$. Thus, $\bar{\rho}(\mu, s, k) = \rho(\mu, s, k)$ and the claim holds. *Inductive Step:* Assume the claim holds for all subformulae of φ. For *Boolean Operators:* For example, if $\varphi = \wedge^p_{i\in[1..N]}\varphi_i$, then the wSTL+ semantics yield $\bar{\rho}(\varphi, s, k) = \min_{i=1,\dots,N}\{p_i \cdot \bar{\rho}(\varphi_i, s, k)\}$. By the inductive hypothesis, for each $i \in [1..N]$, we have $\bar{\rho}(\varphi_i, s, k) \leq \bar{W}_{\varphi_i} \cdot \rho(\phi_i, s, k)$. Thus, $\bar{\rho}(\varphi, s, k) \leq \min_{i=1,\dots,N}\{p_i \cdot \bar{W}_{\varphi_i} \cdot \rho(\phi_i, s, k)\}$. Since, by definition, $\bar{W}_\phi = \max_{i\in\{1,\dots,N\}}\{p_i \cdot \bar{W}_{\varphi_i}\}$, and the STL robustness for conjunction is given by $\rho(\phi, s, k) = \min_{i=1,\dots,N}\{\rho(\phi_i, s, k)\}$, we obtain $\bar{\rho}(\varphi, s, k) \leq \bar{W}_\varphi \cdot \rho(\phi, s, k)$. A similar logic applies to the remaining Boolean and temporal operators. Thus, the inequality holds for all $\varphi' \sqsubseteq \varphi$ by induction. □

The robustness degree of STL, denoted as $\rho(\phi, s, k)$, effectively quantifies the maximum permissible perturbation of the signal s without altering the Boolean truth value of ϕ. However, when weights are incorporated, as seen in wSTL+, the resulting robustness measure $\bar{\rho}(\varphi, s, k)$ can be scaled by the maximum factor $\bar{W}_\varphi$. If $\bar{W}_\varphi > 1$, it implies that while a signal s may withstand a perturbation of up to $\rho(\phi, s, k)$ in the context of STL, the associated wSTL+ robustness is greater. This indicates that the wSTL+ robustness does not accurately represent the *maximum* allowable disturbance required to maintain the Boolean value. Let us clarify this in the following example:

Example 4.1.1. *Consider the STL formula $\phi = \mu_1 \wedge \mu_2$, with $\mu_1 : s \geq 0$ and $\mu_2 : s \leq 5$, then for a signal $s(0) = 1$, the STL robustness of the atomic predicates $\rho(\mu_1, s, 0) = s(0) - 0 = 1$ and $\rho(\mu_2, s, 0) = 5 - s(0) = 4$.Thus, the STL robustness of ϕ is $\rho(\phi, s, 0) = \min\{1, 4\} = 1$. This robustness indicates that any perturbation s' such that $\|s - s'\|_\infty \leq 1$ will not change the Boolean truth of ϕ, while a perturbation of size $s' > 1$ may change it. Now, consider the corresponding wSTL+ formula $\varphi = \wedge^p(\mu_1, \mu_2)$ with weight assignments $p_1 = 2$ for μ_1 and $p_2 = 1$ for μ_2. The wSTL+ robustness is $\bar{\rho}(\varphi, s, 0) = \min\{p_1 \cdot \rho(\mu_1, s, 0), p_2 \cdot \rho(\mu_2, s, 0)\} = \min\{2, 4\} = 2$. Compute the maximal scaling factor for φ as $\bar{W}_\varphi = \max\{p_1, p_2\} = 2$. Notice that although $\bar{\rho}(\varphi, s, 0) = 2$, the STL robustness still indicates that the signal can only tolerate a perturbation of up to 1 without changing its Boolean value. In other words, $\bar{\rho}(\varphi, s, 0) = 2 = \bar{W}_\varphi \cdot \rho(\phi, s, 0)$, so the wSTL+ robustness is scaled by $\bar{W}_\varphi$, but the true perturbation threshold remains $\rho(\phi, s, 0) = 1$.*

Robustness and soundness for wSTL+ Let us clarify how the robustness and soundness work for inclusive and exclusive operators for wSTL+ in the following example.

Example 4.1.2. *Let $\varphi_{ex} = \mathcal{X}^p(\varphi_1, \varphi_2)$ be a wSTL+ specification, with $\mathcal{X} \in \{\wedge, \vee, \barwedge, \barvee\}$ a inclusive or exclusive boolean operator, $p = \mathbf{1}_2$, and φ_1 and φ_2 two wSTL+ subformulae. To ease the notation, we drop the signal s and time $k = 0$ from all robustness definitions in this example. we evaluate the robustness of subformulae φ_1 and φ_2 over signal s as $\tilde{\rho}(\varphi_1)$ and $\tilde{\rho}(\varphi_2)$, respectively. We denote the robustness of the formula φ_{ex} with inclusive conjunction ($\mathcal{X} = \wedge$) as $\tilde{\rho}(\varphi_\wedge)$, for inclusive disjunction ($\mathcal{X} = \vee$) as $\tilde{\rho}(\varphi_\vee)$, for exclusive conjunction ($\mathcal{X} = \barwedge$) as $\tilde{\rho}(\varphi_\barwedge)$, and for exclusive disjunction ($\mathcal{X} = \barvee$) as $\tilde{\rho}(\varphi_\barvee)$. In Table. 4.1 we show four different cases of robustness values for subformulae φ_1 and φ_2 and their corresponding robustness scores for the formula φ_{ex} depending on the operator considered by using definition in* (4.2)*. Note that for the case of inclusive operators, the robustness score captures precisely the degree of robustness of the formula, e.g., in the first case $\tilde{\rho}(\varphi_\wedge) = 1$ which is the value of the subformulae closer to be violated and $\tilde{\rho}(\varphi_\vee) = 2$ indicates the value of*

the subformulae farther to be violated and therefore violate the parent operator. On the other hand, for exclusive operators, it indicates the margin for disrupting the consensus of the subformulae. For $\tilde{\rho}(\varphi_{\veebar})$, the operator is satisfied in the first and last case where both subformulae either satisfy or violate and violated in the second and third case where one of the subformulae violates, and the other satisfies, with values 1 and -1, respectively. Note that in all cases, the values for subformulae are at 1 unit to change the consensus to either satisfying or violating in the first and last cases and to both satisfying or violating in the second and third cases. For $\tilde{\rho}(\varphi_{\barwedge})$, it is satisfied for the second and third cases since only one subformulae is satisfied and violated for the first and last cases since both subformulae are satisfied or violated with values 1 and -1, respectively. Again, there is a distance of 1 for making a subformula to change the consensus of the subformulae.

Table 4.1: Robustness comparison for inclusive and exclusive Boolean operators. We drop the signal and time on all robustness for the sake of space.

$\tilde{\rho}(\varphi_1)$	$\tilde{\rho}(\varphi_2)$	$\tilde{\rho}(\varphi_{\wedge},)$	$\tilde{\rho}(\varphi_{\vee})$	$\tilde{\rho}(\varphi_{\veebar})$	$\tilde{\rho}(\varphi_{\barwedge})$
1	2	1	2	1	-1
-1	2	-1	2	-1	1
1	-2	-2	1	-1	1
-1	-2	-2	-1	1	-1

Some of the properties of STL, wSTL, and wSTL+ are summarized in the accompanying Table 4.2.

Table 4.2: Properties hold for STL, wSTL, and wSTL+.

	STL	wSTL	wSTL+
Robustness	✓	✓	✓
Soundness	✓	✓	✓
Robustness Degree	✓	✗	✗
Preferences	✗	✓	✓
Exclusive Operators	✗	✗	✓

4.1.3 Problem Formulation

In various practical applications, mission specifications are often expressed in STL and its weighted extension wSTL+ to modulate the control synthesis. However, due to limitations in control inputs or conflicting requirements, it is frequently impossible to fully satisfy every subformula of a specification. In such cases, achieving a level of *partial satisfaction* becomes desirable.

Here, we investigate two formulations of the partial satisfaction control synthesis problem: i) In [33], we addressed the partial satisfaction problem related to STL specifications by introducing a partial order over the specification's AST. This partial order, defined by the depth of the AST subformulas, prioritizes subformulas closer to the root, ensuring maximal partial satisfaction. However, this approach does not explicitly allow designers to indicate which constraints are more critical than others. ii) Motivated by this in [32], we considered the partial satisfaction related to wSTL+, in addition to maximizing the satisfaction of lower-depth subformulas, user-defined weights p and w are incorporated to reflect preferences among the subformulas. These weights help modulate the tie-breaking rules in the event of conflicting requirements.

System dynamics

The system dynamics are captured using linear difference equations as follows.

$$s(k+1) = As(k) + Bu(k) + D, \quad s(0) = s_\circ, \tag{4.4}$$

where $s(k) \in S \subseteq \mathbb{R}^n$ is the state variable at time $k \in \mathbb{Z}_{\geq 0}$, S is the state space of the signals, $u(k) \in \mathbf{U} \subseteq \mathbb{R}^m$ is the control input, A and B are the state transition and input matrices of appropriate sizes, and D is the exogenous inputs or additive disturbances.

Control Synthesis with Partial Satisfaction Problem

We define the *partial satisfaction robustness [33]*,

$$\varrho(s,\varphi) = \min_{(\varphi_i,k)\in F_\varphi(s)} \tilde{\rho}(\varphi_i,s,k), \tag{4.5}$$

where $F_\varphi(s) = \{(\varphi_i,k) \mid \nexists(\varphi_i',k') \text{ s.t. } \varphi_i \sqsubset \varphi_i', (s,k)\vDash\varphi_i, (s,k')\vDash\varphi_i'\}$ is the set of lowest-depth subformulae satisfied by s. Note that the partial satisfaction robustness holds for the equivalent STL formula ϕ.

Problem 4.1.1 (Partial Satisfaction)**.** *Given a discrete linear system dynamics* (4.4)*, and either an STL ϕ or wSTL+ φ specification, find input signal $u(k)$ such that the generated state trajectory $s(k)$ satisfies φ as much as possible while considering the user preferences and maximizing the partial satisfaction robustness* (4.5) *over all time-horizon $\|\varphi\|$* (2.5)*. Formally, we have a bi-level optimization problem*

$$\begin{aligned} \max_{\mathbf{u}} \quad & \varrho(s,\varphi) \\ \text{s.t.} \quad & \mathbf{u} \textit{ induces } s \\ & s \in \max_{\mathbf{u}'}{}^{\varphi}\{s'\} \text{ s.t. } \mathbf{u}' \textit{ induces } s' \end{aligned} \tag{4.6}$$

where $\max^\varphi\{s'\}$ denotes the trajectory s' that maximizes the satisfaction of φ (i.e., formula with minimal depth that can be satisfied while capturing user preferences).

Note that the exact problem definition holds for the equivalent STL formula ϕ, but in this case, $\max^\phi$ denotes maximization with respect to partial order $\prec_\phi$ defined above, meaning it finds the maximal elements in the induce lattice [44].

4.1.4 Control Synthesis with Partial Satisfaction Encoding

In this section, we formulate Problem 4.1.1 as an optimization problem and introduce a Mixed Integer Linear Program (MILP) encoding for STL and wSTL+. We consider the

following assumptions throughout the rest of the paper.

Assumption 4.1.1. *STL and wSTL+ specifications are over linear predicates.*

While STL and wSTL+ formulae can be defined over general, non-linear predicates $l_\mu(s(k)) \geq \pi$ for some function $l : \mathbb{R}^n \to \mathbb{R}$, we limit them to simple linear functions $s_i(k) \geq \pi$ (or $s_i(k) \leq \pi$). Our MILP encoding can still be employed by introducing output variables $y_\mu = l_\mu(s(k))$ for all non-linear predicates μ, and using piecewise-linear approximations of the output functions l_μ.

Assumption 4.1.2. *STL and wSTL+ specifications are in positive normal form [133].*

The assumption is not limiting because any STL, and by extension, the wSTL+ formula, can be put in positive normal form, where the negation operators are only in front of predicates. In the following, we eliminate negations by considering predicates defined with either $\geq$ or $\leq$ comparison operators.

The foundation of our encoding of STL and wSTL+ formulae is based on the fraction of subformulae-time pairs that signals satisfy. Instead of encoding margins that are propagated towards a formula's root to compute robustness [133, 130].

Let φ be a wSTL+ formula specification. Let us consider that every weight p and w are normalized such that

$$p_i = \frac{p'_i}{\max_{j=1}^{N} p'_j}, \quad w_i = \frac{w'_i}{\max_{j=1}^{|\bar{I}|} w'_j}.$$

The MILP encoding is defined recursively over the nodes of the AST of wSTL+ formula φ starting from the leaves, the *predicates*.

Let $\mu := s(k) \geq \pi$ be a predicate. We define the variable $z_k^\mu \in \mathbb{B}$ that take value one if predicate μ is considered in the satisfaction of formula φ at time $k \in [0..\|\varphi\|]$, where $\|\varphi\|$ is the time horizon of φ computed based on (2.5). Let M be a large enough number (e.g., larger than the largest upper bound of signals used in wSTL+ specification φ). The following constraints capture the satisfaction of predicates for STL and wSTL+.

$$\varphi = \mu \Rightarrow \begin{cases} s(k) - \pi + M(1 - z_k^{\mu}) \geq 0 \\ s(k) - \pi - M z_k^{\mu} \leq 0 \end{cases} . \tag{4.7}$$

Inclusive preferences constraints

For all-inclusive operators in wSTL+, let us define $z_k^{\varphi} = [0, 1]$ and $z_k^{\phi} = [0, 1]$, capturing the fraction of satisfaction for wSTL+ and STL, respectively.

For *inclusive conjunction* operator z_k^{φ} (z_k^{ϕ}) takes value one if all subformulae φ_i (ϕ_i) with $i \in [1..N]$ are satisfied. Let us define $z_k^{\varphi_i} \in [0, 1]$ ($z_k^{\phi} \in [0, 1]$) capturing the fraction of satisfaction of subformula φ_i (ϕ_i). The constraint capturing the inclusive conjunction semantics for WSTL+ (STL) is

$$\varphi = \bigwedge_i^p \varphi_i \Rightarrow \underbrace{z_k^{\varphi} = \frac{\sum_{i=1}^{N} p_i \cdot z_k^{\varphi_i}}{\sum_{i=1}^{N} p_i}}_{wSTL+} \quad | \quad \underbrace{z_k^{\phi} = \frac{\sum_{i=1}^{N} z_k^{\phi_i}}{N}}_{STL}. \tag{4.8}$$

Note that as weights are normalized, the constraint of the inclusive conjunction takes the form of a weighted arithmetic mean (arithmetic mean). Taking value one only if all $z_k^{\varphi_i}$ ($z_k^{\phi_i}$) are one, in case of conflicting subformulae, the ones with the highest weight p are prioritized.

For *inclusive disjunction* z_k^{φ} (z_k^{ϕ}) takes value one if at least one subformula φ_i (ϕ_i) is satisfied and the weight p_i equals one. Which captures the preference of choosing the subformula with the highest weight. The fraction of satisfaction of each subformula φ_i (ϕ_i) captured by $z_k^{\varphi_i} \in [0, 1]$ ($z_k^{\phi_i} \in [0, 1]$). The following constraint captures the inclusive

disjunction semantics for wSTL+ (STL)

$$\varphi = \bigvee_i^p \varphi_i \Rightarrow \underbrace{z_k^{\varphi} = \max_{i=1:N}\{z_k^{\varphi_i} \cdot p_i\}}_{wSTL+} \quad | \quad \underbrace{z_k^{\phi} = \max_{i=1:N}\{z_k^{\phi_i}\}}_{STL}. \tag{4.9}$$

Note that $z_k^{\varphi} = 1$ implies that at least one subformula $\varphi_i \sqsubset \varphi$ is satisfied. In contrast, $z_k^{\varphi} = 1$ implies that the most preferred subformula is satisfied. Thus, $z_k^{\varphi} < 1$ does not imply violation. Instead, the most preferred option was not the one chosen for satisfaction.

For the *inclusive always* operator, we consider the same logic as for *inclusive conjunction* but with weight and variables over time rather than over subformulae $z_{k'}^{\psi} \in [0, 1]$. Then we have

$$\varphi = \Box_I^w \psi \Rightarrow \underbrace{z_k^{\varphi} = \frac{\sum_{k' \in k+I} z_{k'}^{\psi} \cdot w_{k'}}{\sum_{k' \in k+I} w_{k'}}}_{wSTL+} \quad | \quad \underbrace{z_k^{\phi} = \frac{\sum_{k' \in k+I} z_{k'}^{\psi}}{|I|}}_{STL}. \tag{4.10}$$

The *inclusive eventually* operator follows the same as the *inclusive disjunction* operator but again considers time instead of subformulae for the weights and variables $z_{k'}^{\psi} \in [0, 1]$. Encoded as follows

$$\varphi = \Diamond_I^w \psi \Rightarrow \underbrace{z_k^{\varphi} = \max_{k' \in k+I}\{z_{k'}^{\psi} \cdot w_{k'}\}}_{wSTL+} \quad | \quad \underbrace{z_k^{\phi} = \max_{k' \in k+I}\{z_{k'}^{\psi}\}}_{STL}. \tag{4.11}$$

For the *inclusive until* operator $z_k^{\varphi} = [0, 1]$ ($z_k^{\phi} = [0, 1]$) takes value one if at least at one time $k' \in k + I$ $z_{k'}^{\varphi_2}$ ($z_{k'}^{\phi_2}$) is satisfied. Additionally, at all times in the interval $k'' \in [k..k']$ $z_{k''}^{\varphi_1}$ ($z_{k''}^{\phi_1}$) is satisfied. In wSTL+ the temporal weights indicate the most preferred time instance within the interval where the transition from φ_1 to φ_2 occurs, captured by the following

constraint

$$\varphi = \varphi_1 \mathcal{U}_I^w \varphi_2 \Rightarrow \underbrace{z_k^{\varphi} = \max_{k' \in k+I} \left\{ \frac{w_{k'}}{2} \cdot z_{k'}^{\varphi_2} + \frac{1}{2\bar{I}} \sum_{k'' \in [k..k']} z_{k''}^{\varphi_1} \right\}}_{wSTL+} \tag{4.12}$$

$$\mid \underbrace{z_k^{\phi} = \max_{k' \in k+I} \left\{ \frac{1}{2} \cdot z_{k'}^{\phi_2} + \frac{1}{2\bar{I}} \sum_{k'' \in [k..k']} z_{k''}^{\phi_1} \right\}}_{STL}. \tag{4.13}$$

Exclusive preferences constraints

For the *exclusive conjunction* is the same constraint as for *inclusive disjunction* $z_k^{\varphi_i} \in [0,1]$, but instead of $z_k^{\varphi} \in [0,1]$ we consider $z_k^{\varphi} \in \mathbb{B}$ notice that this capture that either all subformulae φ_i are satisfied, or not at all.

$$\varphi = \bigwedge_i^p \varphi_i \Rightarrow z_k^{\varphi} = \frac{\sum_i^N p_i \cdot z_k^{\varphi_i}}{\sum_i^N p_i} \tag{4.14}$$

For *exclusive disjunction*, we use the same constraint used for the *inclusive conjunction*, also considering $z_k^{\varphi} = [0,1]$ and $z_k^{\varphi_i} \in [0,1]$. Let us define an auxiliary variable $b_i \in \mathbb{B}$ taking value one if $z_k^{\varphi_i}$ is greater than zero. The following set of constraints captures the semantics of *exclusive disjunction*

$$\varphi = \bigvee_i^p \varphi_i \Rightarrow \begin{cases} z_k^{\varphi} = \max_{i=1:N} \{ z_k^{\varphi_i} \cdot p_i \} \\ z_k^{\varphi_i} \leq b_i \\ \sum_{i=1}^N b_i \leq 1 \end{cases}. \tag{4.15}$$

Note that the first constraint captures that at least one subformula has to be satisfied, preferably the one with the maximum weight. Furthermore, the last two capture that only one subformula has to be satisfied and not the rest, generating the desired mutual exclusivity between subformulae.

The *exclusive always* is captured similarly to *exclusive conjunction* but considering time weights and variables instead as subformulae $z_{k'}^{\psi} \in [0,1]$. Also, with $z_k^{\varphi} \in \mathbb{B}$ capturing that subformula $\psi \sqsubseteq \varphi$ is either fully satisfied or not considered at all in the satisfaction.

$$\varphi = \boxdot_I^w \psi \Rightarrow z_k^{\varphi} = \frac{\sum_{k' \in k+I} z_{k'}^{\psi} \cdot w_{k'}}{\sum_{k' \in k+I} w_{k'}}. \tag{4.16}$$

The *exclusive eventually*, is captured similarly as *exclusive disjunction* but considering time weights and variables $z_{k'}^{\psi} \in [0,1]$ and $z_k^{\varphi} \in [0,1]$, we also consider an auxiliary variable $b_i \in \mathbb{B}$ with $i \in [\underline{I}..\bar{I}]$

$$\varphi = \diamondsuit_I^w \psi \Rightarrow \begin{cases} z_k^{\varphi} = \max_{k' \in k+I} \{ z_{k'}^{\psi} \cdot w_{k'} \} \\ z_{k'}^{\varphi} \leq b_i \\ \sum_{i=1}^{\bar{I}} b_i \leq 1 \end{cases}. \tag{4.17}$$

Capturing that subformula, $\psi \sqsubset \varphi$ is satisfied at only one-time step and not at any other time step within the time interval. Making a mutual exclusivity at every time step within the time interval.

The *exclusive until* operator $z_k^{\varphi} = [0,1]$ takes value one if only at a one-time unit in $k' \in k + I$ $z_{k'}^{\varphi_2}$ is satisfied, and this is the most preferred time to transition from ϕ_1 to ϕ_2. Additionally, at all times in the interval $k'' \in [k..k']$ $z_{k''}^{\varphi_1}$ ($z_{k''}^{\phi_1}$) is satisfied.

$$\varphi = \varphi_1 \mathcal{U}_I^w \varphi_2 \Rightarrow \begin{cases} z_k^{\varphi} = \max_{k' \in k+I} \left\{ \frac{w_{k'}}{2} \cdot z_{k'}^{\varphi_2} + \right. \\ \qquad \left. \frac{1}{2\bar{I}} \sum_{k'' \in [k..k']} z_{k''}^{\varphi_1} \right\}, \\ \sum_{k=1}^{\bar{I}} z_k^{\varphi_2} = 1. \end{cases} \tag{4.18}$$

Partial satisfaction satisfiability problem

Finally, the inner level of the optimization problem is formulated as follows

$$\begin{aligned} \max_{s,u,z} \quad & z_0^{\varphi} \\ \text{s.t.} \quad & \text{(4.4)} \quad \text{(linear dynamics)} \\ & \text{(4.7) – (4.18)} \quad \text{(mission satisfaction)} \end{aligned} \tag{4.19}$$

where z_0^{φ} is the root node in the AST of the wSTL+ specification φ. Similarly, for z_o^{ϕ} if the formula is in STL form.

Lowest-depth encoding

Next, we capture the satisfaction of subformulae time pairs (φ, k) using the set $F_\varphi(s) = \{(\varphi_i, k) \mid \nexists(\varphi_i', k') \text{ s.t. } \varphi_i \sqsubset \varphi_i', (s,k) \vDash \varphi_i, (s,k') \vDash \varphi_i'\}$ which contains those subformulae that are satisfied by the signal s and are not superseded by any proper subformula that is also satisfied. This definition applies to both STL and wSTL+; in the latter, user-defined weights further influence the trade-off between satisfying lower-depth (i.e., higher-priority) subformulae and those preferred by the user.

To encode $F_\varphi(s)$ within a Mixed Integer Linear Programming (MILP) framework, we introduce binary variables $\eta_k^{\varphi} \in \{0,1\}$, which indicate that the subformula φ is satisfied at time k and qualifies as a lowest-depth subformula. We enforce this using the following constraints:

$$\eta_k^{\varphi} \leq z_k^{\varphi}, \tag{4.20}$$

$$\eta_k^{\varphi'} \leq 1 - \eta_{k'}^{\varphi''}, \forall \varphi' \sqsubset \varphi'' \sqsubseteq \varphi, k' \in \mathcal{K}', \tag{4.21}$$

where $\mathcal{K}' = \{k' \in [0 \,..\, \|\varphi\|] \mid (s,k') \vDash \varphi'' \Rightarrow (s,k) \vDash \varphi', \forall s\}$ is the finite set of all times where φ'' supersedes φ' at k (see Def. 4.1.5). Since the time horizon $|\varphi|$ and the

number of subformulae in φ are finite, the set $\mathcal{K}'$ can be constructed in polynomial time, making this lowest-depth encoding computationally tractable for both STL and wSTL+ formulations. Note that $\eta_k^{\varphi'}$ can take value one only when φ' is fully satisfy i.e., $z_k^{\varphi'} = 1$, due to constraint (4.20). All ancestors must not be fully satisfied, i.e., $\eta_{k'}^{\varphi''} = 0$, for φ' to be lowest depth as captured in (4.21).

In the case of wSTL+, where satisfaction levels are scaled by weights (e.g., via p for Boolean operators and w for temporal operators), this encoding provides a natural trade-off: it not only prioritizes lower-depth subformulae—thus ensuring that the most critical parts of the specification are satisfied—but also allows user preferences to influence the decision-making when conflicting subformulae exist.

Objectives functions for partial satisfaction

In this section, we formulate three MILP-based objective functions that solve the inner level of the partial satisfaction problem (4.6). These formulations apply to both STL and wSTL+ specifications. The first two are exact solutions, while the third is a relaxation.

Let $\gamma_{\varphi,d} = \sum_{(\varphi',k)\in\Upsilon_\varphi^d} z_k^{\varphi'}$, for all $d \in [0 \,..\, d_{\max}]$, where $d_{\max}$ is the maximum depth of φ, and $\Upsilon_\varphi^d = \{(\varphi', k) \mid \mathrm{depth}_\varphi(\varphi') = d, (s,0) \vDash \varphi \Rightarrow (s,k) \vDash \varphi', \forall s\}$ is the set of all pairs of subformulae φ' of depth d and time points k required for satisfaction of formula φ.

Hierarchical Optimization (HO)

In this approach, we define the multi-objective function

$$R_{HO} = (\gamma_{\varphi,0}, \gamma_{\varphi,1}, \ldots, \gamma_{\varphi,d_{\max}}),$$

where each term $\gamma_{\varphi,d}$ represents the number of subformulae satisfied at depth d. These objectives are optimized sequentially in lexicographical order [167], ensuring that lower-depth subformulae are maximized before considering higher-depth ones. This lead to the

hierarchical MILP problem

$$\begin{aligned} &\max_{s,u,z,\eta} R_{HO}, \quad \text{(optimized in order)} \\ &\text{s.t. (4.4), (4.7) – (4.18), (4.20), (4.21)} \end{aligned} \quad . \tag{4.22}$$

Lowest Depth First (LDF)

We transform the multi-objective function R_{HO} into a single scalar function R_{LDF} by weighting lower-depth subformulae more heavily using a method similar to the big-M trick. The objective is

$$R_{LDF} = \sum_{(\varphi',k)} \eta_k^{\varphi'} P^{-\text{depth}_\varphi(\varphi')} = \sum_{d=0}^{d_{\max}} \gamma_{\varphi,d} P^{-d},$$

that leads to the MILP problem

$$\begin{aligned} &\max_{s,u,z,\eta} R_{LDF}, \\ &\text{s.t. (4.4), (4.7) – (4.18), (4.20), (4.21),} \end{aligned} \tag{4.23}$$

where P is a large constant, greater than $\|\varphi\| \cdot |\varphi|$, and $|\varphi|$ is the length of φ, i.e., the number of Boolean and temporal operators and predicates.

Weighted Largest Number (WLN)

In this approach, we relax the strict requirement of prioritization of lowest depth subformulae and instead maximize the total number of satisfied subformulae, focusing on the root of the formula, which translates to solve directly (4.19). Taking $R_{WLN} = z_0^{\varphi}$ is a sensible choice, because subformulae are still weighted according to their depth due to the recursive constraints (4.7)-(4.18). Consequently, we obtain the MILP

$$\begin{aligned} &\max_{s,u,z} R_{WLN}, \\ &\text{s.t. (4.4), (4.7) – (4.18)} \end{aligned} \tag{4.24}$$

that does not require the additional binary variables η_k^{φ} and associated constraints.

Comparison of objectives functions

For STL, Hierarchical Optimization (HO) guarantees the best satisfaction of high-priority subformulae, providing precise control when prioritization by depth is essential. On the other hand, the Weighted Largest Number (WLN) approach offers a faster solution but without the same structural depth consideration, making it less targeted toward critical subformulae. The Lowest Depth First (LDF) function balances by favoring lower-depth subformulae through exponential weighting. However, the weight parameter P must be carefully adjusted to avoid numerical issues. For wSTL+, the WLN approach is often preferred due to its natural alignment with weight-based preferences, allowing user-defined priorities to be incorporated seamlessly. LDF remains flexible, offering a trade-off between depth-driven priorities and user-defined preferences. This makes it suitable for various scenarios where both factors are essential. However, while valid for wSTL+, HO does not account for these preferences directly, making it less effective when balancing the importance between structural depth and preferences embedded within the weights. Ultimately, selecting the objective function depends on the specific use case: HO is ideal for strict depth-based prioritization, LDF offers a balanced approach between depth and efficiency, and WLN excels when user-defined preferences are paramount. It is important to note that the HO approach can be solved directly using MILP solvers like Gurobi [66], though iterative methods can also be applied using any MILP solver [167].

Partial Satisfaction Robustness LP

In this section, we propose an LP formulation to approximate the robust solution of Problem 4.1.1 using the solutions from one of the MILP formulations described in Sec. 4.1.4. This approach applies to both STL and wSTL+ specifications.

Let $\{z_k^{\varphi}\}_{\varphi' \sqsubseteq \varphi, k \in [0..\|\varphi\|}$ denote the set of decision variables representing the satisfaction

fractions for subformulae (obtained from solving either (4.22), (4.23), or (4.24)). The following LP formulation approximates the outer-level optimization by maximizing the overall robustness of satisfying predicates:

$$\begin{aligned} \max_{s,u} \quad & \rho \\ \text{s.t. (4.4)}, \ & \rho \leq s_i(k) - \mu, \forall \mu \text{ with } z_k^{\mu} = 1 \end{aligned} \tag{4.25}$$

computes the signal s and control u that maximize the robustness of all predicates μ at all times k that are satisfied in the reference solution encoded by $z_k^{\varphi'}$.

Encoding analysis and comparisons

Rate of satisfaction for STL and wSTL+

For STL, the MILP encoding naturally captures the fraction of satisfaction of a formula, meaning that the variable z_0^{φ} directly represents how fully the formula is satisfied. However, in wSTL+, this property no longer holds because the value of z_0^{φ} instead indicates how close the induced trajectory is to capture all the user preferences specified in the formula.

In particular, for disjunctions, eventually, and until operators, if the subformula φ_i contributing to satisfaction is not the one (or not the one with the maximum weight p or w), then the resulting variable z_{φ}^k becomes less than 1. This reduction propagates to the root so that $z_0^{\varphi} < 1$. Importantly, this does not imply a specification violation—it merely reflects that the chosen trajectory is not the most preferred one according to the weight assignments.

Nonetheless, once the MILP computes a solution, it is possible to recover the actual fraction of satisfaction by recursively traversing the MILP variables z_k^{φ} (denoted equivalently as ξ_k^{φ} to avoid confusion) using the following recursive definition. This allows ξ_0^{φ} to correctly capture the fraction of satisfaction for wSTL+, which may differ from z_0^{φ} especially when φ contains disjunctions, eventually, and until operators. The recursive

definition is given by:

$$\xi_0^{\varphi} = \begin{cases} `\xi_k^{\mu}, & \text{if } \varphi = s_i \geq \pi, \\ \frac{\sum_i^N \xi_k^{\varphi_i}}{N}, & \text{if } \varphi = \bigwedge_{i\in[1..N]}{}^{p}\varphi_i, \\ \max\{\min_{i=1}^N\{\xi_k^{\varphi_i}\}, \min_{i=1}^N\{\xi_k^{\neg\varphi_i}\}\}, & \text{if } \varphi = \bigwedge_{i\in[1..N]}{}^{p}\varphi_i, \\ \max_i\{\xi_k^{\varphi_i}\}, & \text{if } \varphi = \bigvee_{i\in[1..N]}{}^{p}\varphi_i, \\ \min\{\max_{i=1}^N\{\xi_k^{\varphi_i}\}, \max_{i=1}^N\{\xi_k^{\neg\varphi_i}\}\}, & \text{if } \varphi = \forall_{i\in[1..N]}{}^{p}\varphi_i, \\ \max_{k'\in I}\{\xi_{k'}^{\varphi}\}, & \text{if } \varphi = \Diamond_I^w\varphi, \\ \min\{\max_{k'\in I}\{\xi_{k'}^{\varphi}\}, \max_{k'\in I}\{\xi_{k'}^{\neg\varphi}\}\}, & \text{if } \varphi = \Diamond_I^w\ \varphi, \\ \frac{\sum_{k'\in I}\xi_{k'}^{\varphi}}{|I|}, & \text{if } \varphi = \Box_I^w\varphi, \\ \max\{\min_{k'\in I}\{\xi_{k'}^{\varphi}\}, \min_{k'\in I}\{\xi_{k'}^{\neg\varphi}\}\}, & \text{if } \varphi = \boxdot_I^w\varphi, \end{cases} \tag{4.26}$$

STL encoding correctness

Proposition 4.1.2 (STL fraction of satisfaction correctness)**.** *Let ϕ be an STL formula in positive normal form, and let s be a signal generated by dynamics in* (4.4)*. Then, for every subformula $\psi \sqsubseteq \phi$ at every time $k \in [0..\|\phi\|]$. Suppose the MILP formulation in Sec. 4.1.4, computes a satisfaction fraction $z_k^{\psi} \in [0,1]$ according to constraints* (4.7) *to* (4.12)*. Then the following properties hold*

1. *Soundness: if $z_k^{\phi} = 1$ then the Boolean semantics of STL guarantee that $(s,k) \vDash \psi$*
2. *Completeness: Conversely, if $(s,k) \vDash \psi$ (i.e., ψ is fully satisfied at time k), then $z_k^{\psi} = 1$*
3. *When ψ is not fully satisfied according to the Boolean semantics, the value z_k^{ψ} exactly captures the fraction of satisfaction as defined by the MILP formulation.*

Proof. The proof follows by structural induction $\psi \sqsubseteq \phi$.

Base case: Let $\psi = \mu = (s \geq \pi) = (s - \pi \geq 0)$, and consider constraint in (4.7). Here $z_k^\mu \in \mathbb{B}$, suppose by contradiction that $z_k^\mu = 0$, then the first constraint becomes $s(k) - \pi + M \geq 0$, which is true, but the second constraint becomes $s(k) - \pi \leq 0$, which is a contradiction. Hence, the only feasible assignment is if $z_k^\mu = 1$. Now consider the case where $\psi = \mu = (s < \pi) = (s - \pi < 0)$ and consider $z_k^\mu = 1$, the first constraint will require $s(k) - \pi \geq 0$, which is a contradiction. Thus, z_k^μ is forced to be zero. Thus, we have $z_k^\psi = 1$ for atomic predicates if and only if $(s, k) \vDash \psi$. This establishes both soundness and completeness at the atomic level. *Inductive step:* For *conjunctive operators* (i.e., Inclusive conjunction and always operators) by the STL constraints in (4.8) and (4.10). Let $\psi = \bigwedge_{i \in [1..N]} \psi_i$ If $(s, k) \vDash \psi$ then each $(s, k) \vDash \psi_i$ for all $i \in [1..N]$. By the induction hypothesis, $z_k^{\psi_i} = 1$ for all $i \in [1..N]$. Hence, $z_k^\psi = \frac{N}{N}$, so $z_k^\psi = 1$. Conversely, if at least one ψ_i is not fully satisfied (i.e., $z_k^{\psi_i} < 1$), then the sum is less than N and consequently $z_k^\psi < 1$. Thus, $z_k^\psi = 1$ if and only if all subformulae are fully satisfied, which matches the Boolean semantics of conjunction. When not all subformulae are satisfied, z_k^ψ is exactly the satisfaction fractions' average (scaled by N). Note that the operator always follows the same logic but over time instances instead of Boolean choices. For *Disjunctive operators* (i.e., Inclusive disjunction and eventually operators) by the STL constraints in (4.9) and (4.11). Let $\psi = \bigvee_{i \in [1..N]} \psi_i$, if $(s, k) \vDash \psi$ then at least one of the disjuncts is fully satisfied; by induction, for some $i \in [1..N]$, $z_k^{\psi_i} = 1$, hence $z_k^\psi = 1$. If none of the ψ_i is fully satisfied, then z_k^ψ equals the maximum satisfaction fraction among them, which is exactly the intended "partial" measure for disjunction. Similar logic follows for the eventually operator. For *until operator* by the STL constraint in (4.12). If $z_k^\psi = 1$, then there exists some $k' \in k + I$ such that $z_{k'}^{\psi_2} = 1$ and $z_{k''}^{\psi_1} = 1$ for every $k'' \in [k, k']$. Hence, by the inductive hypothesis, ψ_2 holds at k' and ψ_1 holds continuously on $[k, k']$, so $(s, k) \vDash \psi$. Conversely, if $(s, k) \vDash \psi$, then there exists some $k' \in k + I$ with $(s, k') \vDash \psi_2$ and $(s, k'') \vDash \psi_1$ for all $k'' \in [k, k']$. Therefore, by induction, $z_{k'}^{\psi_2} = 1$ and $z_{k''}^{\psi_1} = 1$ for every $k'' \in [k, k']$. Inserting these into the constraint gives $\frac{1}{2} \cdot 1 + \frac{1}{2\overline{I}} (k' - k + 1)$, which can reach one

when $k' - k + 1 = \overline{I}$. Moreover, if only partial satisfaction occurs (i.e. some $z_{k''}^{\psi_1} < 1$ or $z_{k'}^{\psi_2} < 1$), the constraint produces a value strictly less than one, accurately quantifying the fraction of satisfaction. By induction, for every subformula ψ the MILP encoding produces a satisfaction fraction z_k^ψ such that: *Soundness:* If $z_k^\psi = 1$, by the recursive construction, every atomic and composite component is fully satisfied, so $(s, k) \vDash \psi$. *Completeness:* If $(s, k) \vDash \psi$ (i.e., ψ is fully satisfied in the Boolean sense), then the induction hypothesis applied to the components of ψ forces $z_k^\psi = 1$. Moreover, when ψ is not fully satisfied, the value z_k^ψ is exactly the fraction determined by the MILP constraints. □

Proposition 4.1.3 (Lowest-depth marking correctness)**.** *Let ϕ be an STL formula in positive normal form, and let s be a signal generated by dynamics in* (4.4)*. Then, for every subformula $\psi \sqsubseteq \phi$ at every time $k \in [0..\|\phi\|]$ $\eta_k^\psi = 1$ if and only if (1) $z_k^\psi = 1$ and (2) for every proper superformula ψ' (with $\psi \sqsubset \psi' \sqsubseteq \phi$) and every corresponding time $k' \in \mathcal{K}'$ we have $\eta_{k'}^{\psi'} = 0$. That is, the MILP "marks" (ψ, k) as a lowest-depth satisfaction if and only if no higher-level (i.e. lower-depth) subformula covering ψ is satisfied.*

Proof. The proof follows by structural induction on the depth d of STL formula ϕ. *Base case:* If ψ is atomic (a leaf in the AST), then it has no proper superformula. Constraint (9) enforces $\eta_k^\psi \leq z_k^\psi$, and since the optimization maximizes the number of marked subformulae, we set $\eta_k^\psi = 1$ exactly when $z_k^\psi = 1$. This is equivalent to ψ being satisfied. *Inductive step:* Assume the claim holds for all subformulae of depth less than d. Let ψ be a subformula of depth d. (Forward Direction): Suppose $\eta_k^\psi = 1$. Then by Constraint (9), $z_k^\psi = 1$. Moreover, Constraint (10) enforces $\eta_k^\psi \leq 1 - \eta_{k'}^{\psi'}$, for every proper superformula ψ' and corresponding time k'. Since $\eta_k^\psi = 1$, it follows that for each such ψ', we must have $\eta_{k'}^{\psi'} = 0$. Thus, ψ is marked only when it is satisfied, and no higher-priority (lower-depth) formula supersedes it. (Reverse Direction): Conversely, suppose $z_k^\psi = 1$ and for every proper superformula ψ' (with $\psi \sqsubset \psi'$) the marking is 0 (i.e. $\eta_{k'}^{\psi'} = 0$). Then Constraint (10) does not force η_k^ψ to be 0, and by the optimization objective—which favors marking as many

lowest-depth satisfactions as possible—we set $\eta_k^{\psi} = 1$. Thus, $\eta_{k'}^{\psi'} = 1$ if and only if ψ is fully satisfied at time k ($z_k^{\psi} = 1$) and no proper superformula is marked, which precisely captures the lowest-depth requirement. □

Theorem 4.1.2 (Correctness of partial satisfaction MILP encoding for STL)**.** *Let ϕ be an STL formula in positive normal form, and let s be a signal generated by dynamics in* (4.4)*, and let z_k^{ϕ} be the decision variable assigned by the MILP encoding for ϕ at time $k \in [0..\|\phi\|]$. The MILP encoding in Sec.4.1.4 for STL is correct if the following two conditions hold*

1. *There exists a control input u generating a signal s satisfying ϕ (i.e., $(s, 0) \vDash \phi$) if and only if $z_0^{\phi} = 1$.*

2. *The MILP encoding prioritizes the satisfaction of the lowest-depth subformulae.*

Proof. The first property follows by recursively using Proposition 4.1.2, such that $z_0^{\phi} = 1 \Leftrightarrow (s, 0) \vDash \phi$. The second property follows by Proposition 4.1.3, such that $\eta_k^{\psi} = 1 \Leftrightarrow (z_k^{\psi} = 1)$ with $\psi \sqsubseteq \phi$ and no proper superformula ψ' such that $\psi \sqsubset \psi' \sqsubseteq \phi$ is not selected. □

Theorem 4.1.3 (Correctness of partial satisfaction MILP encoding for wSTL+)**.** *Let φ be a wSTL+ formula in positive normal form, and let s be a signal generated by dynamics in* (4.4)*, and let z_k^{φ} be the decision variable assigned by the MILP encoding for φ at time $k \in [0..\|\varphi\|]$. The MILP encoding in Sec.4.1.4 for wSTL+ is correct if the following three conditions hold*

1. *$z_k^{\varphi} = 1$ if φ is fully satisfied according to the user preferences.*

Proof. (Sketch) The proof proceeds by structural induction on the abstract syntax tree (AST) of φ. The base case for atomic predicates is established via the MILP constraints that force the binary decision variables to match the Boolean predicate evaluation. For

the inductive step, assume correctness for all subformulae of φ, similarly as it was performed for Proposition 4.1.2. This completes the inductive argument, and hence the theorem holds. □

4.1.5 Case studies: Partial Satisfaction for STL

We describe two study cases showing how PS works and its performance over the three encodings. First, we consider a multi-robot motion planning problem in a planar, continuous environment. Second, we show routing for heterogeneous teams of robots with sets of capabilities over graphs that represent discrete finite abstractions. More details can be found in [76]. All computation was performed on a PC with 20 cores at 3.7GHz and 64 GB of RAM. We used Gurobi [66] as MILP solver, which ensures that as long as the encoding is correctly defined, corner cases such as fully satisfiable and non-satisfiable specifications are cover

Continuous Space Multi-robot Planning Let us consider first a single robot navigating in a planar environment $\mathcal{M} \subset \mathbb{R}^2$. Thus, $s(k) = [s_x, s_y]^\intercal \in \mathbb{R}^2$. We define regions of interest $\mathcal{A} = [-9.5, -5.5]^2$, $\mathcal{B} = [5.5, 9.5] \times [-9.5, -5.5]$, $\mathcal{C} = [5.5, 9.5]^2$, $\mathcal{D} = [-9.5, -5.5] \times [5.5, 9.5]$ in $\mathcal{M}$, and region $\mathcal{E} = [-2.5, 2.5]^2 \subseteq \mathcal{M}$ that robot $s(k)$ needs to avoid. We arbitrarily choose $A = B = I_{2\times2}$, and $D = 0_{2\times2}$ in (2.9). All three methods, HO, LDF, and WLN, are able to find trajectories that satisfy the given specification when they exist. For example, consider the following STL formula $\phi_{FS} = ((\Box_{[0,3]}\mathcal{A}) \wedge (\Diamond_{[7,14]}\mathcal{B}) \wedge (\Box_{[15,20]}\mathcal{D}) \wedge (\sim \Box_{[0,20]}\mathcal{E}))$. In English, it required that "within time interval $[0, 3]$ stay at region $\mathcal{A}$, eventually within $[0, 14]$ visit $\mathcal{B}$, stay at $\mathcal{D}$ within $[15, 20]$ and always avoid region $\mathcal{E}$". In Fig. 4.1.1, red lines show the solution for this specification using the three methods. All of three solution trajectories start at the blue star in $\mathcal{A}$, and then they visit $\mathcal{B}$ and end at $\mathcal{D}$, where squares denote the final position of the robot.

Next, consider the STL formula $\phi_{PS} = (\Box_{[0,3]}\mathcal{A}) \wedge (\Box_{[10,21]}\mathcal{B}) \wedge (\Box_{[10,21]}\mathcal{D}) \wedge (\sim \Box_{[0,20]}\mathcal{E})$

that can not be fully satisfied due to the competing subformulae that require staying at $\mathcal{B}$ and $\mathcal{D}$ in the same time interval $[10, 21]$. The solutions trajectories are shown in Fig. 4.1.1 with blue lines. Methods HO and LDF partially satisfy the specification by choosing to visit region $\mathcal{D}$, and, thus, satisfying the lowest-depth subformulae of ϕ_{PS}. In contrast, WLN splits the satisfaction between the predicates defining the two regions $\mathcal{B}$ and $\mathcal{D}$. In other words, the specification requires either $(s_x \geq 5.5) \wedge (s_x \leq 9.5) \wedge (s_y \geq -9.5) \wedge (s_y \leq -5.5)$ or $(s_x \geq -9.5) \wedge (s_x \leq -5.5) \wedge (s_y \geq 5.5) \wedge (s_y \leq 9.5)$ to hold, and WLN is satisfying half from the former, $(s_x \geq -9.5) \wedge (s_x \leq -5.5)$, and half of the latter, $(s_y \geq -9.5) \wedge (s_y \leq -5.5)$, in the time window $[10, 21]$. The WLN trajectory satisfies the same maximum satisfaction fraction as HO and LDF, but does not fully satisfy any of the two tasks. However, the case was hand-crafted to show that WLN does not always return the optimal solution. In particular, we made the tasks symmetric. If the time intervals for visiting $\mathcal{B}$ and $\mathcal{D}$ are different, then the weight of their associated predicates differs and leads to a preference for one of the two regions.

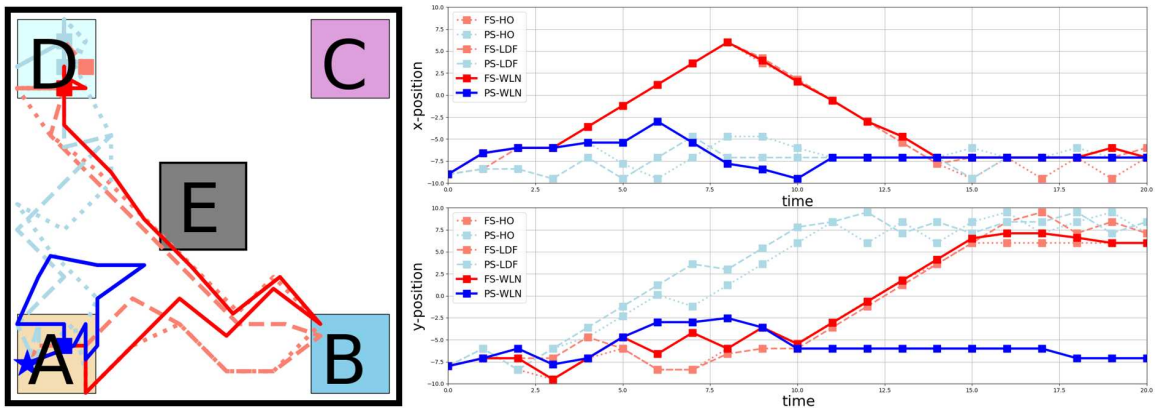


Figure 4.1.1: State space and timeline in x-position and y-position of ϕ_{FS} in red and ϕ_{PS} in blue using HO, LDF, and WLN encoding methods.

In a multi-robot setting, unsatisfiability can result due to robot dropout as opposed to competing tasks. For instance, in Fig. 4.1.2 we consider two scenarios satisfying the following specification $\phi_{mr} = ((\square_{[15,30]} s_1 \in \mathcal{C}) \wedge (\square_{[15,30]} s_2 \in \mathcal{D}) \wedge (\square_{[15,30]} s_3 \in \mathcal{C}) \wedge (\square_{[15,30]} s_4 \in \mathcal{B}) \wedge (\square_{[15,30]} s_5 \in \mathcal{C}) \wedge (\square_{[15,30]} s_6 \in \mathcal{C}))$ with six robots. First, we solve the

problem with the HO approach shown in Fig. 4.1.2 with dashed lines. Then, we consider the exact specification, but we make robot s_1 remain stuck in region $\mathcal{E}$. Even though it is just one agent specification, ϕ_{mr} is only partially satisfied. The PS solution computed using LDF is shown in solid lines.

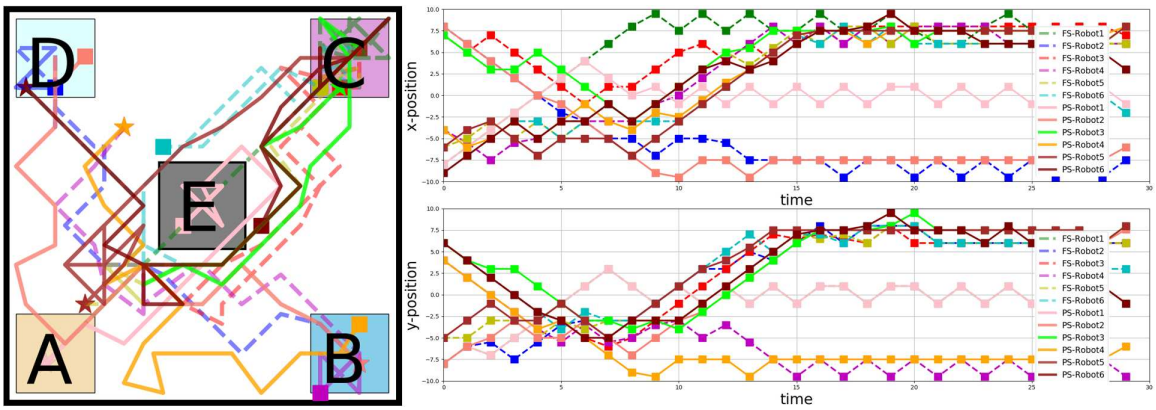


Figure 4.1.2: State space and timeline in x-position and y-position of ϕ_{mr} fully satisfied with HO approach in dashed lines. Then, the solution using LDF for robot one getting stuck in a solid.

Finally, we compare the runtime performance between methods HO, LDF, and WLN by increasing the number of robots from one to six. In Fig. 4.1.3, we see that when there are one or two robots, LDF and WLN are faster than HO. Nevertheless, when the number of robots increases, so does the depth in the specification, and LDF becomes the worst-performing method. The specification we used for this comparison is $\phi_n = \bigwedge_{j=1}^{n}(\bigwedge_{\ell=1}^{3}(\Box_{I_{\ell,j}}(s_n \in \mathcal{X}_{\ell,j})))$, with $n \in [1 \,..\, 6]$ the number of robots, $\mathcal{X}_{\ell,j}$ any arbitrary region in $\{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}, \mathcal{E}\}$, and $I_{\ell,j}$ arbitrary time interval, respectively. It is also relevant to highlight that even when WLN is the fastest, it can drive to undesired PS if the specification contains symmetric competing subformulae.

Route Planning with Capability Temporal Logic (CaTL) We consider the precision agriculture application shown in Fig. 4.1.4. The environment is abstracted as a set of states $\mathcal{Q} = \{q_1, q_2, ..., q_9\}$ corresponding to regions of interest. Edges $\mathcal{E}$, shown as black lines, between states represent feasibility of transition between regions with positive in-

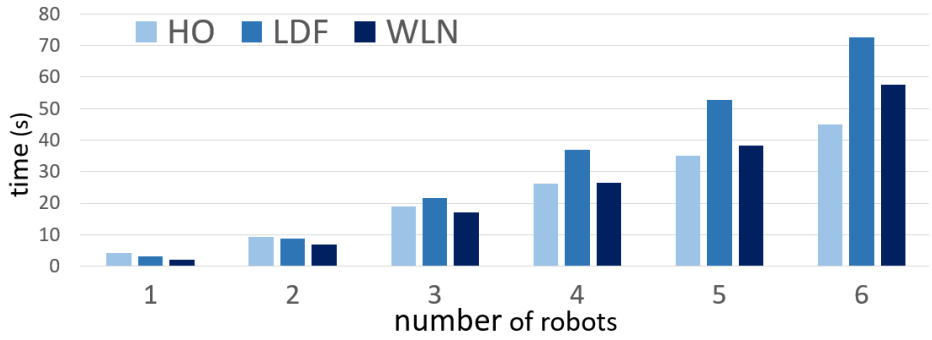


Figure 4.1.3: Runtime performance comparison between HO, LDF, and WLN varying the number of robots from one to six, satisfying specification ϕ_n.

teger durations $\mathcal{W}$. Self-loops (not shown) capture staying at states and have duration 1. States are labeled with atomic propositions from a set $\mathcal{AP}$. Agents have capabilities $Cap = \{Vis, IR, UV, Mo\}$. The capability set of an agent determines its class. Fig. 4.1.4 shows the initial states of each agent and their class. We use CaTL, a frag-

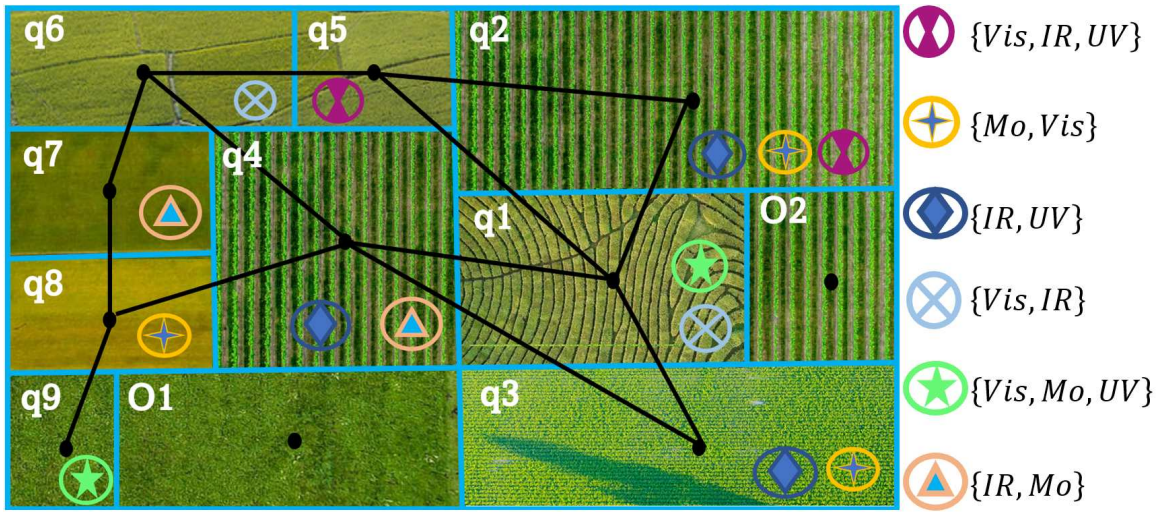


Figure 4.1.4: Schematic of the precision agriculture problem. Regions correspond to crop types. Symbols represent robots with their associated capabilities. Capabilities include ultraviolet sensing (UV), moisture sensing (Mo), infrared sensing (IR), and vision (Vis). Black lines denote the transition system between regions.

ment of STL, to specify the mission for the multi-robot team. The core units of CaTL are tasks $T = (d, \pi, cp)$ defined by a duration d, a label π of regions where it takes place, and the required capabilities cp. The map $cp : Cap \to \mathbb{Z}_{\geq 0}$ indicates the minimum number of agents with each capability; zero means that capability is not needed. Tasks are com-

bined using Boolean and temporal operators in the same way as STL (excluding negation). We use the encoding of team dynamics in [76], where robots are bundled together based on class. Thus, the team state is the number of robots of each class at each state of the environment. The team dynamics are encoded as flows on the environment graph, which gives rise to a time-delayed linear system. The MILP constraints and further details can be found in [76]. Consider tasks $T_1 = (3, q_2, (c_2, 3))$, $T_2 = (3, q_4, (c_1, 4))$, $T_3 = (6, q_9, (c_3, 3))$, $T_4 = (5, q_1, (c_4, 4))$, $T_5 = (3, q_6, (c_2, 1), (c_4, 1))$, and $T_6 = (2, q_8, (c_4, 4))$, and three CaTL specifications $\phi_1, \phi_2 = \bigwedge_i \square_{[0,20]} T_i$, with $i \in \{1, 3, 5\}$ and $i \in \{2, 4, 6\}$ respectively, and $\phi_3 = \bigwedge_{i=1}^{6} \Diamond_{[0,20]} T_i$. Partial satisfaction of ϕ_ℓ can arise due to the lack of robots with the required capabilities, transition system constraints, and competing subformulae. The following table shows the performance of the three methods for the three specifications.

Table 4.3: Comparison on time performance and satisfaction percentage of specifications ϕ_1, ϕ_2, and ϕ_3 by methods HO, LDF, WLN, and baseline CaTL [94].

spec	**HO**		**LDF**		**WLN**		**CaTL**	
	Satisfaction	**t**(s)	**Satisfaction**	**t**(s)	**Satisfaction**	**t**(s)	ρ	**t**(s)
ϕ_1	0.929	1.884	0.943	1.732	0.972	1.498	-6	2.075
ϕ_2	0.982	1.617	0.990	1.859	0.991	1.635	-4	2.341
ϕ_3	1	1.773	1	2.009	1	1.940	1	2.32

For the baseline CaTL encoding [94], negative robustness indicates that ϕ_1 and ϕ_2 are violated. However, it does not indicate the specification violation percentage, i.e., which subformulae are satisfied and how many.

Table 4.3 shows that all of the partial satisfaction approaches outperform the baseline encoding for CaTL [94]. Due to partial satisfaction, transform the specification's robustness as a linear problem approach of all predicates and time instances known to be satisfiable. We avoid adding binary variables for disjunctions and, eventually, operators since the inner optimization level indicates what needs to be satisfied. Furthermore, for small problems, WLN is faster than the other two. However, as the number of problem variables increases, the HO method using Gurobi becomes faster and more suitable to use.

4.1.6 Case Studies: Partial Satisfaction for wSTL

In this section, we showcase and test the functionality of the wSTL+ MILP encoding under partial satisfaction conditions (PS-wSTL+). First, we show the versatility of the exclusive operators added to the wSTL+ language specification. Then, we show the control synthesis for an agent satisfying a specification of navigating in a planar environment with temporal and logic constraints, modulating the partial satisfaction solution through the weights. Lastly, we show the time performance and complexity comparison between STL, PS-STL, and PS-wSTL+ under partial satisfaction MILP solutions.

Exclusive operators functionality

Let us consider a single robot navigating in a planar environment $\mathcal{M} \subset \mathbb{R}^2$ shown in Fig. 4.1.5. Regions of interest $\mathcal{R} = \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}$, with $\mathcal{A} = [-4, -9] \times [-9, 9]$, $\mathcal{B} = [-9, 4] \times [9, 9]$, $\mathcal{C} = [4, -9] \times [9, 9]$, and , $\mathcal{D} = [-9, -4] \times [9, -9]$. Note that some regions are not disjoint, e.g., $\mathcal{A} \cap \mathcal{B} \neq \varnothing$, or disjoint e.g., $\mathcal{A} \cap \mathcal{C} = \varnothing$. We arbitrarily choose $A = B = I_{2\times 2}$, for the robot dynamics as in (4.4). Thus, $s(k) = [s_x(k), s_y(k)]^\top \in \mathbb{R}^2$. We consider initial position as $s(0) = (s_x(0), s_y(0)) = (0, 0)$. We consider the following wSTL+ specifications

$$\varphi_1 = \wedge^{p_1}\left(\boxdot^w_{I_1}(\wedge^{p_1}(\mathcal{A}, \mathcal{B})), \boxdot^w_{I_2}(\wedge^{p_1}(\mathcal{B}, \mathcal{C})), \boxdot^w_{I_3}(\wedge^{p_1}(\mathcal{C}, \mathcal{D}))\right),$$
$$\varphi_2 = \wedge^{w}\left(\boxdot^w_{I_1}(\vee^{p}(\mathcal{A}, \mathcal{B}, \mathcal{D})), \boxdot^w_{I_2}(\vee^{p}(\mathcal{B}, \mathcal{C}, \mathcal{A})), \boxdot^w_{I_2}(\vee^{p}(\mathcal{C}, \mathcal{D}, \mathcal{A}))\right),$$

where $I_1 = [5..9]$, $I_2 = [14..18]$, $I_3 = [24..26]$, $p_1 = [1, 1]$ all weights w are defined to be one of appropriate size, and $p = [1, 0.5, 0.5]$. For simplicity, with a slight abuse of notation, we define the formulae directly over the regions instead of defining predicates over all four boundaries of each region.

In Fig. 4.1.5. we show the solution of both specifications. Note that φ_1 (red trajectory) uses exclusive conjunctions for specifying the regions that must be visited within the inter-

vals. As this operator impose that all subformulae have to be satisfied or not at all, the only solution possible is given when the robot visits the overlapping regions between areas of interest requested. On the other hand, φ_2 (blue trajectory) uses exclusive disjunction, and the weight p assigns a larger preference to the first region specified when the robot visits that region without crossing the other regions. Since exclusive disjunction imposes one subformula and not any other subformulae. Lastly, we show the importance of declaring exclusive operators correctly. We modify φ_1, defining exclusive conjunctions between disjoints regions, and therefore, the solution is set to not visit any region (green start) since visiting both regions simultaneously is physically impossible.

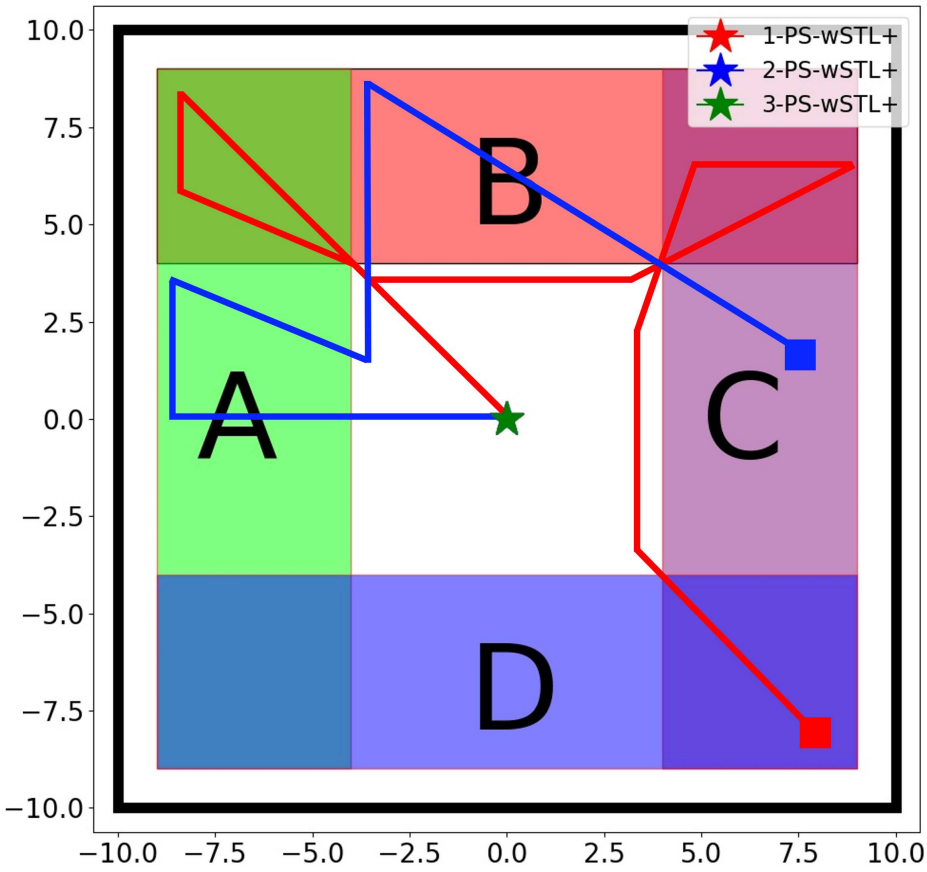


Figure 4.1.5: Exclusive operators functionality.

Control synthesis

Let us consider the same robot defined in the previous section, with the initial position as $s(0) = (s_x(0), s_y(0)) = (-9, -9)$. We define disjoint regions of interest $\mathcal{A} = [-9.5, -5.5]^2$, $\mathcal{B} = [5.5, 9.5] \times [-9.5, -5.5]$, $\mathcal{C} = [5.5, 9.5]^2$, $\mathcal{D} = [-9.5, -5.5] \times [5.5, 9.5]$ in $\mathcal{M}$, and region $\mathcal{E} = [-2.5, 2.5]^2 \subseteq \mathcal{M}$ that robot $s(k)$ needs to avoid. We consider the STL ϕ and wSTL φ specifications

$$\phi = (\square_{[0,1]}\mathcal{A}) \wedge (\square_{[10,15]}\mathcal{C}) \wedge (\square_{[25,30]}\mathcal{D}) \wedge (\square_{[0,30]}\mathcal{E}^c), \tag{4.27}$$

$$\varphi = \wedge^p \left((\square^w_{[0,1]}\mathcal{A}), (\square^w_{[10,15]}\mathcal{C}), (\square^w_{[25,30]}\mathcal{D}), (\square^w_{[0,30]}\mathcal{E}^c) \right), \tag{4.28}$$

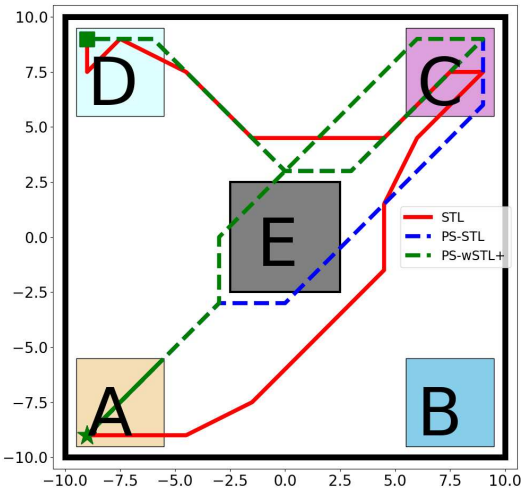
Going to regions of interest can be specified in STL and wSTL+ form by constraining s_x and s_y inside the region boundaries. Note that we use $\mathcal{E}^c$, indicating that the robot can move in any location out of this area.

In Fig. 4.1.6(a), we set all weights w and p to one. It can be seen that the wSTL+ (green dashed line) and PS-STL [33] (blue dashed line) have similar trajectories for satisfying the specification, and all three encodings including STL [133] (solid red line) can satisfy the mission.

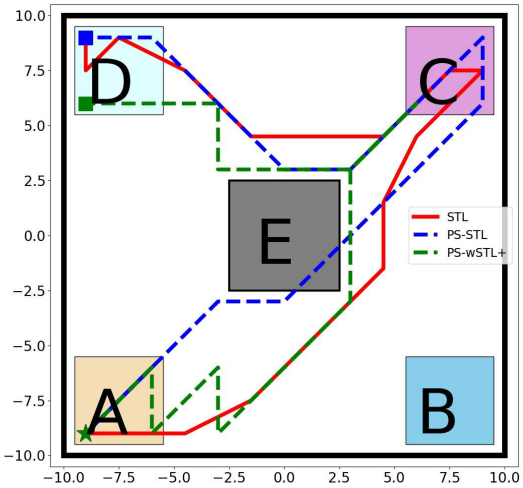
In Fig.4.1.6(b), we show how the solution for the wSTL+ (green dashed line) can change by using exclusive temporal and logical operators instead of standard operators specifically for avoiding region $\mathcal{E}$ the trajectory is carefully synthesized never to cross it.

In Fig.4.1.6(c), we show a case where there is a conflict between subformulae to visit two regions simultaneously which are physically impossible, specified in STL and wSTL+ as $\phi = \square_{[5,6]}(\mathcal{B} \wedge \mathcal{D})$, $\varphi = \square^w_{[5,6]}(\wedge^p(\mathcal{B}, \mathcal{D}))$, where $w = [1, 1]$ and $p = [0.5, 1]$. As expected, standard STL encoding [133] minimally violates both subformulae. By using the *hierarchical method* in PS-STL [33] the solution computed is going to region $\mathcal{B}$. However, this solution is obtained because it is the first specified. It might not be the most preferred option. In contrast, we specify a preference to visit region $\mathcal{D}$ by defining a larger weight,

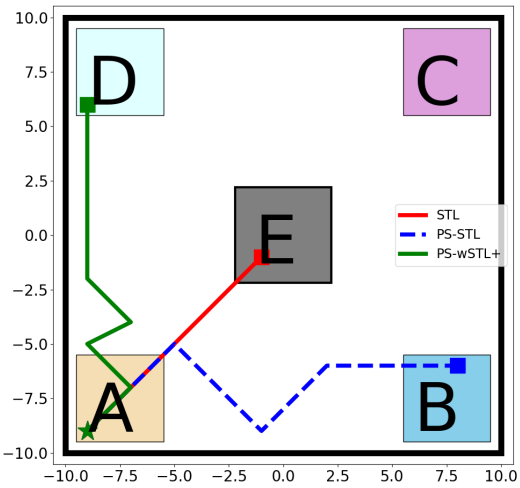
and the solution captures the user preference, and the robot visits the desired region.



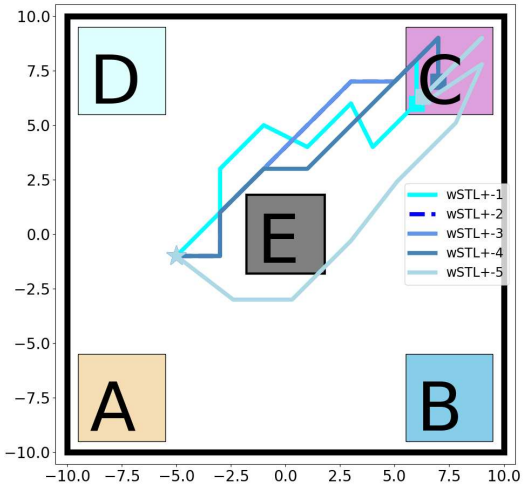
(a) Feasible symmetric specification for PS-STL and PS-wSTL+ (unit weights and inclusive operators).



(b) Feasible symmetric specification for PS-STL and PS-wSTL+ (unit weights and exclusive operators).



(c) Partial satisfaction choosing the most preferred subformula.



(d) Trajectories around an obstacle changing the weights in the wSTL specification

Figure 4.1.6: Trajectories for wSTL φ and STL ϕ specifications in a two-dimensional environment.

Lastly, let us consider the initial position of the agent as $s_x(0) = -5$, $s_y(0) = -1$, and the following wSTL specification

$$\varphi = \wedge^{p_i} \left((\Box^{w}_{[0,7]} \mathcal{E}^c), (\Box^{w}_{[8,18]} \mathcal{C}) \right),$$

see Fig. 4.1.6(d). We generate different trajectories, and the first four are constrained

by control inputs bounds of two units. Some of them are getting closer or farther to $\mathcal{E}$ according to the weights. However, for the last case, the control bounds are three units, and the solution is to go around the bottom boundary of $\mathcal{E}$, keep distance to the obstacle, and reach maximum robustness. Thus, varying weights may produce topologically similar trajectories or different from the STL one.

Time performance and complexity comparison

We show the run time performance comparison between STL [133], partial satisfaction for STL (PS-STL) [33], and our partial satisfaction with wSTL (PS-wSTL) encodings with random weights by gradually increasing the size of the mission specification. Let us consider six variables x, y, z, u, v, and w, all with a lower-bound of -9 and upper-bound of 9. The STL ϕ and wSTL φ specification are the following

$$\phi = \bigwedge_{1}^{n} \mathcal{T}_I \left((\mathfrak{s}_1 \otimes_1 \Xi_1) \, \mathcal{L} \, (\mathfrak{s}_2 \otimes_2 \Xi_2) \right), \quad \varphi = \bigwedge_{1}^{n\ p} \tilde{\mathcal{T}}_I^w \left(\tilde{\mathcal{L}}^p \left((\mathfrak{s}_1 \otimes_1 \Xi_1), \, (\mathfrak{s}_2 \otimes_2 \Xi_2) \right) \right),$$

where $\mathcal{T} \in \{\Box, \Diamond\}$, $\tilde{\mathcal{T}} \in \{\Box, \Diamond, \boxdot, \diamondsuit\}$, $\mathcal{L} \in \{\wedge, \vee\}$, $\mathcal{L} \in \{\wedge, \vee, \barwedge, \barvee\}$, $\mathfrak{s}_1$ and $\mathfrak{s}_2 \in \{x, y, z, u, v, w\}$, $\otimes \in \{<, \leq, >, \geq\}$, Ξ_1 and $\Xi_2 = \text{rand}(-8, 8)$ are variables randomly chosen, n is an iterator that grows from 1 to 200, and the time interval of the temporal operator is defined randomly as $I = [n + 4, \ldots, n + 4 + \text{rand}(1, 5)]$, and the weights for Boolean p and temporal w operators are randomly chosen in an interval $(0, 1]$. In Fig. 4.1.7, we compare time performance for STL, PS-STL, and PS-wSTL+ growing formulae with random weights. Note that the STL performance grows linearly and is faster than the other two. However, there is a slight difference between STL and PS-wSTL+, which is expected since more constraints are required in the encoding to capture the importance or preferences in the specification. Moreover, no partial satisfaction is imposed by the STL solution. The performance of PS-wSTL+ is better than PS-STL. Both capture partial satisfiability, but only PS-wSTL+ captures preferences and importance. We hypothesize that weights act as tie-breakers, al-

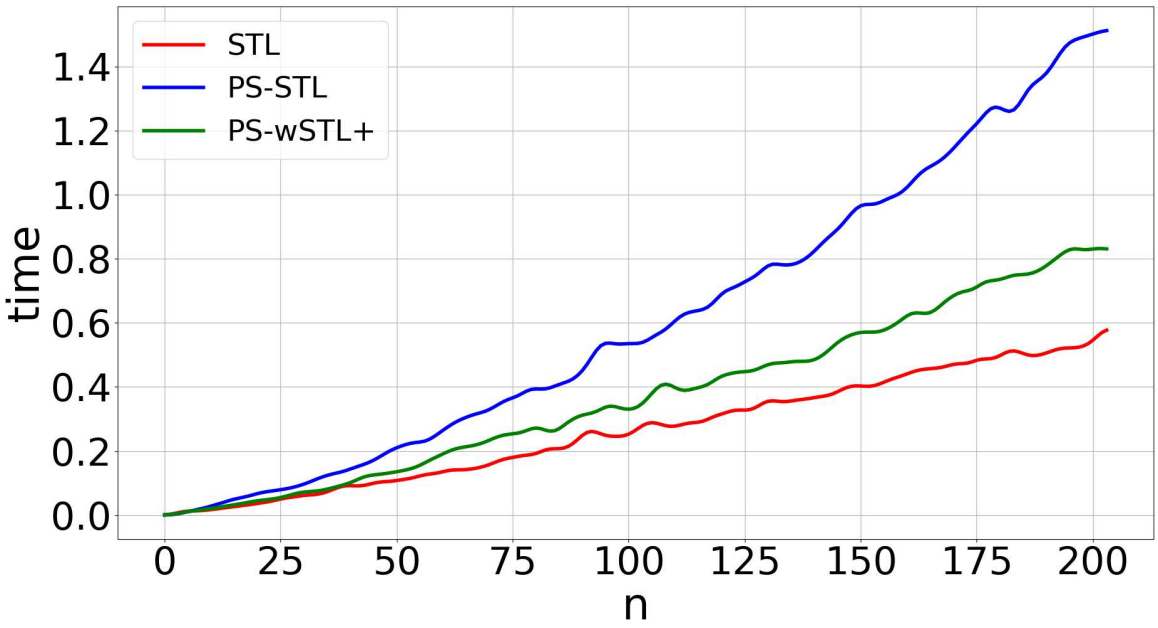


Figure 4.1.7: Time performance between random generated specification in STL [133], PS-STL [33], PS-wSTL+.

lowing the MILP solver to prune sub-optimal infeasible solutions.

4.1.7 Conclusions

This chapter presents a formal framework for achieving partial satisfaction (PS) of Signal Temporal Logic (STL) specifications to address conflicting, infeasible, or resource-constrained multi-robot missions. We extend traditional STL and Weighted STL (wSTL) to a new specification language, wSTL+, which is capable of capturing both inclusive (soft) and exclusive (hard) preferences over sub-formulae. This extension allows for more nuanced mission descriptions, where certain tasks must be strictly enforced while others can be relaxed or partially satisfied based on their assigned importance. We addressed a significant limitation of classical STL robustness metrics, which treat all sub-formulae equally and do not explicitly prioritize tasks in cases of conflict. By introducing fractional satisfaction metrics and defining preferences through both weight modulation and the depth of the abstract syntax tree (AST), our framework enables planners to assess the extent to which each task is satisfied rather than simply determining binary feasibility. This approach of-

fers a systematic way to guide system behavior toward fulfilling the most critical aspects of a mission, even when complete satisfaction is unachievable. Additionally, the chapter formalizes the PS problem as a bilevel optimization framework. The inner Mixed-Integer Linear Program (MILP) selects the optimal combination of satisfiable sub-formulae while adhering to user-defined preferences and exclusivity constraints. Meanwhile, the outer Linear Program (LP) maximizes the robustness of the chosen sub-formulae, ensuring that the system's behavior aligns as closely as possible with the preferred specifications. Through examples and motivating scenarios, we demonstrated the framework's capability to manage situations where traditional STL or wSTL might fail or compute hard-to-interpret solutions.

Chapter 5

Conclusions and Future Work

This section summarizes the key contributions of this dissertation, discusses the broader implications of the work, and outlines directions for future research.

5.1 Conclusions

This dissertation introduces a novel formal methods framework for multi-robot mission planning that combines high-level temporal logic specifications with optimization-based control synthesis. The primary contributions are as follows:

5.1.1 Novel MILP Encodings

A disjunction-centric Mixed-Integer Linear Programming (MILP) encoding has been developed to address the computational challenges inherent in traditional formulations. By introducing binary decision variables specifically for disjunctive operators and leveraging the natural structure of conjunctive constraints, this framework achieves significant improvements in scalability and efficiency, which are critical for real-time operations in large-scale multi-robot systems.

5.1.2 Modeling Complex Dynamics in Multi-Robot Systems

In addition to the formal synthesis contributions, this dissertation advances the modeling of complex multi-robot dynamics. The framework has been applied to scenarios that involve swarm behavior, modular robot reconfiguration, heterogeneous team coordination, and resource transportation. Novel models, including network flow formulations, capture the intricate interplay among agents, enabling effective planning for collective behaviors and resource-constrained logistics in diverse and challenging operational environments.

5.1.3 Handling Infeasible Specifications through Partial Satisfaction

Recognizing that real-world mission constraints may be conflicting or unattainable, a systematic approach to partial satisfaction has been introduced. This strategy allows the framework to relax non-critical constraints while ensuring the fulfillment of essential objectives, thereby enhancing multi-robot operations' overall robustness and adaptability according to user preferences.

5.2 Implications and Future Work

The contributions presented in this work advance the state-of-the-art in multi-robot coordination by introducing a unified framework that is both scalable and robust. The optimized Mixed Integer Linear Programming (MILP) formulations facilitate the efficient resolution of high-dimensional planning problems while incorporating weighted temporal logic, which provides a more expressive method for capturing complex mission requirements.

Future research directions include:

1. Extension to Nonlinear Dynamics: Adapting the current MILP-based approach to handle nonlinear system dynamics, potentially through hybrid or iterative optimiza-

tion methods, to expand the framework's applicability.

2. Integration with Learning-Based Approaches: Utilizing machine learning techniques to develop adaptive control strategies that can learn from operational experiences and dynamically adjust to new environments and task requirements.

3. Development of Distributed Solutions: Exploring decentralized or distributed control architectures to overcome the limitations of centralized planning, especially as the scale and complexity of multi-robot systems continue to increase.

4. Extensive Experimental Validation: Conducting real-world experiments to validate the proposed framework under practical constraints and uncertainties, which will help assess its performance in operational settings.

Bibliography

[1] Mohamed Abdelkader, Samet Güler, Hassan Jaleel, and Jeff S Shamma. "Aerial Swarms: Recent Applications and Challenges". In: *Current Robotics Reports* (2021), pp. 1–12.

[2] Sofie Ahlberg and Dimos V Dimarogonas. "Human-in-the-loop control synthesis for multi-agent systems under hard and soft metric interval temporal logic specifications". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2019, pp. 788–793.

[3] Hammad Ahmad and Jean-Baptiste Jeannin. "A program logic to verify signal temporal logic specifications of hybrid systems". In: *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*. 2021, pp. 1–11.

[4] Hossein Ahmadzadeh, Ellips Masehian, and Masoud Asadpour. "Modular robotic systems: Characteristics and applications". In: *Journal of Intelligent & Robotic Systems* 81.3 (2016), pp. 317–357.

[5] Apurva Badithela, Josefine B Graebener, Wyatt Ubellacker, Eric V Mazumdar, Aaron D Ames, and Richard M Murray. "Synthesizing reactive test environments for autonomous systems: testing reach-avoid specifications with multi-commodity flows". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 12430–12436.

[6] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT press, 2008.

[7] Fernando S Barbosa, Daniel Duberg, Patric Jensfelt, and Jana Tumova. “Guiding autonomous exploration with signal temporal logic”. In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3332–3339.

[8] Pierfrancesco Bellini, Riccardo Mattolini, and Paolo Nesi. “Temporal logics for real-time system specification”. In: *ACM Computing Surveys (CSUR)* 32.1 (2000), pp. 12–42.

[9] C. Belta and V. Kumar. “Abstraction and control for groups of robots”. In: *IEEE Transactions on robotics* 20.5 (2004), pp. 865–875.

[10] C. Belta, B. Yordanov, and E. A. Gol. *Formal methods for discrete-time dynamical systems*. Vol. 89. Springer, 2017.

[11] Spring Berman, Adám Halász, M Ani Hsieh, and Vijay Kumar. “Optimized stochastic policies for task allocation in swarms of robots”. In: *IEEE transactions on robotics* 25.4 (2009), pp. 927–937.

[12] Spring Berman, Ádám Halász, M. Ani Hsieh, and Vijay Kumar. “Optimized stochastic policies for task allocation in swarms of robots”. In: *IEEE Transactions on Robotics* 25.4 (2009), pp. 927–937. ISSN: 15523098. DOI: 10.1109/TRO.2009.2024997.

[13] Amit Bhatia, Lydia E Kavraki, and Moshe Y Vardi. “Sampling-based motion planning with temporal goals”. In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 2689–2696.

[14] Andreas Birk and Stefano Carpin. “Merging occupancy grid maps from multiple robots”. In: *Proceedings of the IEEE* 94.7 (2006), pp. 1384–1397.

[15] Andrea Bisoffi and Dimos V Dimarogonas. “Satisfaction of linear temporal logic specifications through recurrence tools for hybrid systems”. In: *IEEE Transactions on Automatic Control* 66.2 (2020), pp. 818–825.

[16] Giuseppe Bombara, Cristian-Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. “A decision tree approach to data classification using signal temporal logic”. In: *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. 2016, pp. 1–10.

[17] John L Bresina, Ari K Jónsson, Paul H Morris, and Kanna Rajan. “Activity Planning for the Mars Exploration Rovers.” In: *ICAPS*. 2005, pp. 40–49.

[18] Christoph Brzoska. “Programming in metric temporal logic”. In: *Theoretical computer science* 202.1-2 (1998), pp. 55–125.

[19] Ian Buckley and Magnus Egerstedt. “Infinitesimal Shape-Similarity for Characterization and Control of Bearing-Only Multirobot Formations”. In: *IEEE Transactions on Robotics* (2021).

[20] Ali Tevfik Buyukkocak and Derya Aksaray. “Temporal relaxation of signal temporal logic specifications for resilient control synthesis”. In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE. 2022, pp. 2890–2896.

[21] Alvaro Caballero and Giuseppe Silano. “A Signal Temporal Logic Motion Planner for Bird Diverter Installation Tasks with Multi-Robot Aerial Systems”. In: *IEEE Access* (2023).

[22] Mingyu Cai, Kevin Leahy, Zachary Serlin, and Cristian-Ioan Vasile. “Probabilistic Coordination of Heterogeneous Teams From Capability Temporal Logic Specifications”. In: *IEEE Robotics and Automation Letters* 7.2 (2021), pp. 1190–1197.

[23] Gustavo A Cardona and Juan M Calderon. “Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations”. In: *Applied Sciences* 9.8 (2019), p. 1702.

[24] Gustavo A Cardona, Disha Kamale, and Cristian-Ioan Vasile. “Mixed Integer Linear Programming Approach for Control Synthesis with Weighted Signal Temporal Logic”. In: *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*. 2023, pp. 1–12.

[25] Gustavo A Cardona, Disha Kamale, and Cristian-Ioan Vasile. “STL and wSTL control synthesis: A disjunction-centric mixed-integer linear programming approach”. In: *Nonlinear Analysis: Hybrid Systems* 56 (2025), p. 101576.

[26] Gustavo A Cardona, Kevin Leahy, Makai Mann, and Cristian-Ioan Vasile. “A Flexible and Efficient Temporal Logic Tool for Python: PyTeLo”. In: *arXiv preprint arXiv:2310.08714* (2023).

[27] Gustavo A Cardona, Kevin Leahy, and Cristian-Ioan Vasile. “Temporal Logic Swarm Control with Splitting and Merging”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 12423–12429.

[28] Gustavo A Cardona, Juan Ramirez-Rugeles, Eduardo Mojica-Nava, and Juan M Calderon. “Visual victim detection and quadrotor-swarm coordination control in search and rescue environment”. In: *International Journal of Electrical and Computer Engineering* 11.3 (2021), p. 2079.

[29] Gustavo A Cardona, David Saldaña, and Cristian-Ioan Vasile. “Planning for Modular Aerial Robotic Tools with Temporal Logic Constraints”. In: *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE. 2022, pp. 2878–2883.

[30] Gustavo A Cardona and Cristian-Ioan Vasile. “Partial Satisfaction of Signal Temporal Logic Specifications for Coordination of Multi-robot Systems”. In: *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*. Springer. 2022, pp. 223–238.

[31] Gustavo A Cardona and Cristian-Ioan Vasile. “Planning for heterogeneous teams of robots with temporal logic, capability, and resource constraints”. In: *The International Journal of Robotics Research* (), p. 02783649241247285.

[32] Gustavo A Cardona and Cristian-Ioan Vasile. “Preferences on Partial Satisfaction using Weighted Signal Temporal Logic Specifications”. In: *2023 European Control Conference (ECC)*. IEEE. 2023, pp. 1–6.

[33] Gustavo A. Cardona and Cristian-Ioan Vasile. “Partial Satisfaction of Signal Temporal Logic Specifications for Coordination of Multi-robot Systems”. In: *Algorithmic Foundations of Robotics XV*. Cham: Springer International Publishing, 2023, pp. 223–238. ISBN: 978-3-031-21090-7.

[34] Sebastian Castro, Sarah Koehler, and Hadas Kress-Gazit. “High-level control of modular robots”. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 3120–3125.

[35] Ji Chen, Salar Moarref, and Hadas Kress-Gazit. “Verifiable control of robotic swarm from high-level specifications”. In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pp. 568–576.

[36] Ji Chen, Hanlin Wang, Michael Rubenstein, and Hadas Kress-Gazit. “Automatic control synthesis for swarm robots from formation and location-based high-level specifications”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 8027–8034.

[37] Yushan Chen, Xu Chu Ding, Alin Stefanescu, and Calin Belta. “Formal approach to the deployment of distributed robotic teams”. In: *IEEE Transactions on Robotics* 28.1 (2011), pp. 158–171.

[38] Yushan Chen, Jana Tumova, and Calin Belta. “LTL robot motion control based on automata learning of environmental dynamics”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 5177–5182.

[39] Smriti Chopra, Giuseppe Notarstefano, Matthew Rice, and Magnus Egerstedt. “A distributed version of the hungarian method for multirobot assignment”. In: *IEEE Transactions on Robotics* 33.4 (2017), pp. 932–947.

[40] Howie Choset. “Coverage for robotics–a survey of recent results”. In: *Annals of mathematics and artificial intelligence* 31 (2001), pp. 113–126.

[41] Howie Choset. “Coverage of known spaces: The boustrophedon cellular decomposition”. In: *Autonomous Robots* 9 (2000), pp. 247–253.

[42] J. Cortés. “Global and robust formation-shape stabilization of relative sensing networks”. In: *Automatica* 45.12 (2009), pp. 2754–2762. ISSN: 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2009.09.019.

[43] Jorge Cortés and Magnus Egerstedt. “Coordinated control of multi-robot systems: A survey”. In: *SICE Journal of Control, Measurement, and System Integration* 10.6 (2017), pp. 495–503.

[44] Brian A Davey and Hilary A Priestley. *Introduction to lattices and order*. Cambridge university press, 2002.

[45] Angela Davids. “Urban search and rescue robots: from tragedy to technology”. In: *IEEE Intelligent systems* 17.2 (2002), pp. 81–83.

[46] Jyotirmoy V Deshmukh, Alexandre Donzé, Shromona Ghosh, Xiaoqing Jin, Garvit Juniwal, and Sanjit A Seshia. “Robust online monitoring of signal temporal logic”. In: *Formal Methods in System Design* 51 (2017), pp. 5–30.

[47] Yancy Diaz-Mercado, Austin Jones, Calin Belta, and Magnus Egerstedt. “Correct-by-construction control synthesis for multi-robot mixing”. In: *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE. 2015, pp. 221–226.

[48] Dr Shantanu K Dixit and Mr SB Dhayagonde. “Design and implementation of e-surveillance robot for video monitoring and living body detection”. In: *International Journal of Scientific and Research Publications* 4.4 (2014), pp. 2250–3153.

[49] Franck Djeumou, Zhe Xu, and Ufuk Topcu. “Probabilistic Swarm Guidance Subject to Graph Temporal Logic Specifications”. In: *Robotics: Science and Systems (RSS)*. 2020.

[50] Adel Dokhanchi, Bardh Hoxha, and Georgios Fainekos. “5th International Conference on Runtime Verification, Toronto, ON, Canada.” In: ed. by Borzoo Bonakdarpour and Scott A. Smolka. Springer, 2014. Chap. On-Line Monitoring for Temporal Logic Robustness, pp. 231–246. ISBN: 978-3-319-11164-3. DOI: 10.1007/978-3-319-11164-3_19.

[51] Alexandre Donzé and Oded Maler. “Robust satisfaction of temporal logic over real-valued signals”. In: *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer. 2010, pp. 92–106.

[52] Marco Dorigo, Guy Theraulaz, and Vito Trianni. “Reflections on the future of swarm robotics”. In: *Science Robotics* 5.49 (2020), eabe4385.

[53] Marco Dorigo, Guy Theraulaz, and Vito Trianni. “Swarm Robotics: Past, Present, and Future [Point of View]”. In: *Proceedings of the IEEE* 109.7 (2021), pp. 1152–1165. DOI: 10.1109/JPROC.2021.3072740.

[54] M. Egerstedt and X. Hu. “Formation constrained multi-agent control”. In: *IEEE Transactions on Robotics and Automation* 17.6 (2001), pp. 947–951.

[55] Georgios E Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J Pappas. “Temporal logic motion planning for dynamic robots”. In: *Automatica* 45.2 (2009), pp. 343–352.

[56] Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. “Temporal logic motion planning for mobile robots”. In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE. 2005, pp. 2020–2025.

[57] Georgios E Fainekos and George J Pappas. “Robustness of temporal logic specifications for continuous-time signals”. In: *Theoretical Computer Science* 410.42 (2009), pp. 4262–4291.

[58] Cameron Finucane, Gangyuan Jing, and Hadas Kress-Gazit. “LTLMoP: Experimenting with language, temporal logic and robot control”. In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 1988–1993.

[59] A. Franchi, C. Masone, V. Grabe, M. Ryll, H. H. Bülthoff, and P. R. Giordano. “Modeling and control of UAV bearing formations with bilateral high-level steering”. In: *The International Journal of Robotics Research* 31.12 (2012), pp. 1504–1525.

[60] James Guo Ming Fu, Tirthankar Bandyopadhyay, and Marcelo H Ang. “Local Voronoi decomposition for multi-agent task allocation”. In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 1935–1940.

[61] Avinash Gautam and Sudeept Mohan. “A review of research in multi-robot systems”. In: *2012 IEEE 7th international conference on industrial and information systems (ICIIS)*. IEEE. 2012, pp. 1–5.

[62] M. Guo and D. Dimarogonas. “Multi-agent plan reconfiguration under local LTL specifications”. In: *The International Journal of Robotics Research* 34.2 (2015), pp. 218–235.

[63] M. Guo and D. V. Dimarogonas. “Task and Motion Coordination for Heterogeneous Multi-agent Systems With Loosely Coupled Local Tasks”. In: *IEEE Transactions on Automation Science and Engineering* 14.2 (Apr. 2017), pp. 797–808.

[64] Meng Guo and Dimos V Dimarogonas. “Task and motion coordination for heterogeneous multiagent systems with loosely coupled local tasks”. In: *IEEE Transactions on Automation Science and Engineering* 14.2 (2016), pp. 797–808.

[65] Meng Guo and Michael M Zavlanos. “Distributed data gathering with buffer constraints and intermittent communication”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 279–284.

[66] LLC Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2020. URL: `http://www.gurobi.com`.

[67] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. “Exploring Network Structure, Dynamics, and Function using NetworkX”. In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[68] Iman Haghighi, Sadra Sadraddini, and Calin Belta. “Robotic swarm control from spatio-temporal specifications”. In: *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE. 2016, pp. 5708–5713.

[69] Dorit S Hochbaum. “The pseudoflow algorithm: A new algorithm for the maximum-flow problem”. In: *Operations research* 56.4 (2008), pp. 992–1009.

[70] Wolfgang Hönig, TK Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. “Multi-agent path finding with kinematic constraints”. In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 26. 2016, pp. 477–485.

[71] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. “Introduction to automata theory, languages, and computation”. In: *Acm Sigact News* 32.1 (2001), pp. 60–65.

[72] Sofia Hustiu, Dimos V Dimarogonas, Cristian Mahulea, and Marius Kloetzer. “Multi-robot Motion Planning under MITL Specifications based on Time Petri Nets”. In: *2023 European Control Conference (ECC)*. IEEE. 2023, pp. 1–8.

[73] A. Jadbabaie, J. Lin, and A.S. Morse. “Coordination of groups of mobile autonomous agents using nearest neighbor rules”. In: *IEEE Transactions on Automatic Control* 48.6 (June 2003), pp. 988–1001. ISSN: 0018-9286. DOI: 10.1109/TAC.2003.812781.

[74] Gangyuan Jing, Tarik Tosun, Mark Yim, and Hadas Kress-Gazit. “An end-to-end system for accomplishing tasks with modular robots”. In: *Robotics: Science and Systems* 12 (2016). ISSN: 2330765X. DOI: 10.15607/rss.2016.xii.025.

[75] A. 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, pp. 224–241.

[76] Austin M Jones, Kevin Leahy, Cristian Vasile, Sadra Sadraddini, Zachary Serlin, Roberto Tron, and Calin Belta. “ScRATCHS: Scalable and Robust Algorithms for Task-based Coordination from High-level Specifications”. In: *International Symposium of Robotics Research*. 2019.

[77] Disha Kamale, Sofie Haesaert, and Cristian-Ioan Vasile. “Cautious planning with incremental symbolic perception: Designing verified reactive driving maneuvers”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 1652–1658.

[78] Disha Kamale, Eleni Karyofylli, and Cristian-Ioan Vasile. “Automata-based Optimal Planning with Relaxed Specifications”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021.

[79] Y. Kantaros, M. Guo, and M. Zavlanos. “Temporal logic task planning and intermittent connectivity control of mobile robot networks”. In: *IEEE Transactions on Automatic Control* 64.10 (2019), pp. 4105–4120.

[80] Sertac Karaman and Emilio Frazzoli. “Linear temporal logic vehicle routing with applications to multi-UAV mission planning”. In: *International Journal of Robust and Nonlinear Control* 21.12 (2011), pp. 1372–1395.

[81] Sertac Karaman and Emilio Frazzoli. “Vehicle routing problem with metric temporal logic specifications”. In: *2008 47th IEEE conference on decision and control*. IEEE. 2008, pp. 3953–3958.

[82] Jesper Karlsson, Fernando S Barbosa, and Jana Tumova. “Sampling-based motion planning with temporal logic missions and spatial preferences”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 15537–15543.

[83] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. “Multi-robot task allocation: A review of the state-of-the-art”. In: *Cooperative robots and sensor networks 2015* (2015), pp. 31–51.

[84] Marius Kloetzer and Calin Belta. “Temporal logic planning and control of robotic swarms by hierarchical abstractions”. In: *IEEE Transactions on Robotics* 23.2 (2007), pp. 320–330.

[85] Marius Kloetzer and Cristian Mahulea. “A Petri net based approach for multi-robot path planning”. In: *Discrete Event Dynamic Systems* 24 (2014), pp. 417–445.

[86] Ron Koymans. “Specifying real-time properties with metric temporal logic”. In: *Real-time systems* 2.4 (1990), pp. 255–299.

[87] Ron Koymans. “Specifying real-time properties with metric temporal logic”. In: *Real-time systems* 2.4 (1990), pp. 255–299.

[88] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. “Temporal-logic-based reactive mission and motion planning”. In: *IEEE transactions on robotics* 25.6 (2009), pp. 1370–1381.

[89] Hadas Kress-Gazit, Morteza Lahijanian, and Vasumathi Raman. “Synthesis for robots: Guarantees and feedback for robot behavior”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 211–236.

[90] Vince Kurtz and Hai Lin. “A more scalable mixed-integer encoding for metric temporal logic”. In: *IEEE Control Systems Letters* 6 (2021), pp. 1718–1723.

[91] Vincent Kurtz and Hai Lin. “Mixed-integer programming for signal temporal logic with fewer binary variables”. In: *IEEE Control Systems Letters* 6 (2022), pp. 2635–2640.

[92] Bruno Lacerda and Pedro U Lima. “Petri net based multi-robot task coordination from temporal logic specifications”. In: *Robotics and Autonomous Systems* 122 (2019), p. 103289.

[93] Kevin Leahy, Austin Jones, Mac Schwager, and Calin Belta. “Distributed information gathering policies under temporal logic constraints”. In: *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE. 2015, pp. 6803–6808.

[94] Kevin Leahy, Austin Jones, and Cristian Ioan Vasile. “Fast decomposition of temporal logic specifications for heterogeneous teams”. In: *IEEE Robotics and Automation Letters* (2022).

[95] Kevin Leahy, Zachary Serlin, Cristian-Ioan Vasile, Andrew Schoer, Austin M Jones, Roberto Tron, and Calin Belta. “Scalable and robust algorithms for task-based coordination from high-level specifications (ScRATCHeS)”. In: *IEEE Transactions on Robotics* (2021).

[96] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. “Geometric tracking control of a quadrotor UAV on SE (3)”. In: *49th IEEE conference on decision and control (CDC)*. IEEE. 2010, pp. 5420–5425.

[97] Danyang Li, Mingyu Cai, Cristian-Ioan Vasile, and Roberto Tron. “Learning signal temporal logic through neural network for interpretable classification”. In: *2023 American Control Conference (ACC)*. IEEE. 2023, pp. 1907–1914.

[98] Kaier Liang and Cristian-Ioan Vasile. “Fair Planning for Mobility-on-Demand with Temporal Logic Requests”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 1283–1289.

[99] Lars Lindemann and Dimos V Dimarogonas. “Control barrier functions for signal temporal logic tasks”. In: *IEEE control systems letters* 3.1 (2018), pp. 96–101.

[100] Lars Lindemann, Jakub Nowak, Lukas Schönbächler, Meng Guo, Jana Tumova, and Dimos V Dimarogonas. “Coupled multi-robot systems under linear temporal logic and signal temporal logic tasks”. In: *IEEE Transactions on Control Systems Technology* 29.2 (2019), pp. 858–865.

[101] Yehonathan Litman, Neeraj Gandhi, Linh Thi Xuan Phan, and David Saldaña. "Vision-Based Self-Assembly for Modular Multirotor Structures". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2202–2208.

[102] Chao Liu, Michael Whitzer, and Mark Yim. "A distributed reconfiguration planning algorithm for modular robots". In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 4231–4238.

[103] Wenliang Liu, Kevin Leahy, Zachary Serlin, and Calin Belta. "Robust multi-agent coordination from catl+ specifications". In: *2023 American Control Conference (ACC)*. IEEE. 2023, pp. 3529–3534.

[104] Zhiyu Liu, Jin Dai, Bo Wu, and Hai Lin. "Communication-aware motion planning for multi-agent systems from signal temporal logic specifications". In: *2017 American Control Conference (ACC)*. IEEE. 2017, pp. 2516–2521.

[105] Angel Madridano, Abdulla Al-Kaff, David Martín, and Arturo De La Escalera. "Trajectory planning for multi-robot systems: Methods and applications". In: *Expert Systems with Applications* 173 (2021), p. 114660.

[106] Ashutosh Mahajan. "Presolving mixed-integer linear programs". In: *Wiley Encyclopedia of Operations Research and Management Science* (2010), pp. 4141–4149.

[107] MHA Majid, MR Arshad, and RM Mokhtar. "Swarm Robotics Behaviors and Tasks: A Technical Review". In: *Control Engineering in Robotics and Industrial Automation* (2022), pp. 99–167.

[108] Oded Maler and Dejan Nickovic. "Monitoring temporal properties of continuous signals". In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[109] Oded Maler and Dejan Nickovic. "Monitoring temporal properties of continuous signals". In: *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[110] Noushin Mehdipour, Cristian-Ioan Vasile, and Calin Belta. “Arithmetic-geometric mean robustness for control from signal temporal logic specifications”. In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 1690–1695.

[111] Noushin Mehdipour, Cristian-Ioan Vasile, and Calin Belta. “Specifying user preferences using weighted signal temporal logic”. In: *IEEE Control Systems Letters* 5.6 (2020), pp. 2006–2011.

[112] Claudio Menghi, Sergio Garcia, Patrizio Pelliccione, and Jana Tumova. “Multi-robot LTL planning under uncertainty”. In: *International Symposium on Formal Methods*. Springer. 2018, pp. 399–417.

[113] M. Mesbahi and M. Egerstedt. *Graph theoretic methods in multiagent networks*. Vol. 33. Princeton University Press, 2010.

[114] N. Michael, C. Belta, and V. Kumar. “Controlling three dimensional swarms of robots”. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. 2006, pp. 964–969.

[115] Salar Moarref and Hadas Kress-Gazit. “Automated synthesis of decentralized controllers for robot swarms from high-level temporal logic specifications”. In: *Autonomous Robots* 44.3 (2020), pp. 585–600.

[116] Salar Moarref and Hadas Kress-Gazit. “Decentralized control of robotic swarms from high-level temporal logic specifications”. In: *2017 international symposium on multi-robot and multi-agent systems (MRS)*. IEEE. 2017, pp. 17–23.

[117] Robin R Murphy and Jennifer L Burke. “Up from the rubble: Lessons learned about HRI from search and rescue”. In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*. Vol. 49. 3. SAGE Publications Sage CA: Los Angeles, CA. 2005, pp. 437–441.

[118] Alexandros Nikou, Jana Tumova, and Dimos V Dimarogonas. “Cooperative task planning of multi-agent systems under timed temporal specifications”. In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 7104–7109.

[119] Gennaro Notomista, Siddharth Mayya, Seth Hutchinson, and Magnus Egerstedt. “An optimal task allocation strategy for heterogeneous multi-robot systems”. In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 2071–2076.

[120] R. Olfati-Saber and R. M. Murray. “Distributed cooperative control of multiple vehicle formations using structural potential functions”. In: *IFAC Proceedings Volumes* 35.1 (2002), pp. 495–500.

[121] Yasemin Ozkan-Aydin and Daniel I Goldman. “Self-reconfigurable multilegged robot swarms collectively accomplish challenging terradynamic tasks”. In: *Science Robotics* 6.56 (2021), eabf1628.

[122] Yash Vardhan Pant, Houssam Abbas, Rhudii A Quaye, and Rahul Mangharam. “Fly-by-logic: Control of multi-drone fleets with temporal logic objectives”. In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE. 2018, pp. 186–197.

[123] Michael Park, Sachin Chitta, Alex Teichman, and Mark Yim. “Automatic configuration recognition methods in modular robots”. In: *The International Journal of Robotics Research* 27.3-4 (2008), pp. 403–421.

[124] Terence Parr. *The definitive ANTLR reference: building domain-specific languages*. Pragmatic Bookshelf, 2007.

[125] Janko Petereit, Jürgen Beyerer, Tamim Asfour, Sascha Gentes, Björn Hein, Uwe D Hanebeck, Frank Kirchner, Rüdiger Dillmann, Hans Heinrich Götting, Martin Weiser, et al. “ROBDEKON: Robotic systems for decontamination in hazardous environments”. In: *2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE. 2019, pp. 249–255.

[126] Erion Plaku and Sertac Karaman. “Motion planning with temporal-logic specifications: Progress and challenges”. In: *AI communications* 29.1 (2016), pp. 151–162.

[127] Aniruddh Puranic, Jyotirmoy Deshmukh, and Stefanos Nikolaidis. “Learning from demonstrations using signal temporal logic”. In: *Conference on Robot Learning*. PMLR. 2021, pp. 2228–2242.

[128] Hazhar Rahmani and Jason M O’Kane. “Optimal temporal logic planning with cascading soft constraints”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 2524–2531.

[129] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia. “Model predictive control with signal temporal logic specifications”. In: *53rd IEEE Conference on Decision and Control*. Dec. 2014, pp. 81–87. DOI: 10.1109/CDC.2014.7039363.

[130] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. “Model predictive control with signal temporal logic specifications”. In: *53rd IEEE Conference on Decision and Control*. IEEE. 2014, pp. 81–87.

[131] Yara Rizk, Mariette Awad, and Edward W Tunstel. “Cooperative heterogeneous multi-robot systems: A survey”. In: *ACM Computing Surveys (CSUR)* 52.2 (2019), pp. 1–31.

[132] E. Rohmer, S. P. N. Singh, and M. Freese. “CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework”. In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. 2013. URL: www.coppeliarobotics.com%22.

[133] Sadra Sadraddini and Calin Belta. “Robust temporal logic model predictive control”. In: *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE. 2015, pp. 772–779.

[134] Indranil Saha, Rattanachai Ramaithitima, Vijay Kumar, George J Pappas, and Sanjit A Seshia. “Automated composition of motion primitives for multi-robot systems from safe LTL specifications”. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2014, pp. 1525–1532.

[135] Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. “Multirobot coordination with counting temporal logics”. In: *IEEE Transactions on Robotics* 36.4 (2019), pp. 1189–1206.

[136] Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. “Provably-correct coordination of large collections of agents with counting temporal logic constraints”. In: *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE. 2017, pp. 249–258.

[137] Yunus Emre Sahin, Petter Nilsson, and Necmiye Ozay. “Synchronous and asynchronous multi-agent coordination with cLTL+ constraints”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 335–342.

[138] David Saldana, Bruno Gabrich, Guanrui Li, Mark Yim, and Vijay Kumar. “ModQuad: The flying modular structure that self-assembles in Midair”. In: *Proceedings - IEEE International Conference on Robotics and Automation* (2018), pp. 691–698. ISSN: 10504729. DOI: 10.1109/ICRA.2018.8461014.

[139] David Saldana, Bruno Gabrich, Guanrui Li, Mark Yim, and Vijay Kumar. “Modquad: The flying modular structure that self-assembles in midair”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 691–698.

[140] N Sariff and Norlida Buniyamin. “An overview of autonomous mobile robot path planning algorithms”. In: *2006 4th student conference on research and development*. IEEE. 2006, pp. 183–188.

[141] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. “Multi-objective search for optimal multi-robot planning with finite LTL specifications and resource constraints”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 768–774.

[142] Philipp Schillinger, Mathias Bürger, and Dimos V Dimarogonas. “Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems”. In: *The international journal of robotics research* 37.7 (2018), pp. 818–838.

[143] Philipp Schillinger, Mathias Bürger, and Dimos V. Dimarogonas. “Simultaneous task allocation and planning for temporal logic goals in heterogeneous multi-robot systems”. In: *International Journal of Robotics Research* 37.7 (2018), pp. 818–838. ISSN: 17413176. DOI: 10.1177/0278364918774135.

[144] Jungwon Seo, Jamie Paik, and Mark Yim. “Modular Reconfigurable Robotics”. In: *The Annual Review of Control, Robotics, and Annu. Rev. Control Robot. Auton. Syst* 2 (2019), pp. 63–88. DOI: 10.1146/annurev-control-053018.

[145] T. Setter, A. Fouraker, M. Egerstedt, and H. Kawashima. “Haptic interactions with multi-robot swarms using manipulability”. In: *Journal of Human-Robot Interaction* 4.1 (2015), pp. 60–74.

[146] Mayank Sewlia, Christos K Verginis, and Dimos V Dimarogonas. “MAPS2: Multi-Robot Anytime Motion Planning under Signal Temporal Logic Specifications”. In: *arXiv preprint arXiv:2309.05632* (2023).

[147] Shiva Shahrokhi and Aaron T Becker. “Stochastic swarm control with global inputs”. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 421–427.

[148] Karun B Shimoga. “Robot grasp synthesis algorithms: A survey”. In: *The International Journal of Robotics Research* 15.3 (1996), pp. 230–266.

[149] Rohit Singh and Indranil Saha. “An Online Planning Framework for Multi-Robot Systems with LTL Specification”. In: *2024 ACM/IEEE 15th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE. 2024, pp. 180–191.

[150] Dawei Sun, Jingkai Chen, Sayan Mitra, and Chuchu Fan. “Multi-agent Motion Planning from Signal Temporal Logic Specifications”. In: *arXiv preprint arXiv:2201.05247* (2022).

[151] Jugesh Sundram, Duong Van Nguyen, Gim Song Soh, Roland Bouffanais, and Kristin Wood. “Development of a miniature robot for multi-robot occupancy grid mapping”. In: *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*. IEEE. 2018, pp. 414–419.

[152] Mohamed S Talamali, Arindam Saha, James AR Marshall, and Andreagiovanni Reina. "When less is more: Robot swarms adapt better to changes with constrained communication". In: *Science Robotics* 6.56 (2021), eabf1416.

[153] Sebastian Thrun. "Probabilistic algorithms in robotics". In: *Ai Magazine* 21.4 (2000), pp. 93–93.

[154] Ilya Tkachev and Alessandro Abate. "Formula-Free Finite Abstractions for Linear Temporal Verification of Stochastic Hybrid Systems". In: *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*. HSCC '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 283–292. ISBN: 9781450315678. DOI: 10.1145/2461328.2461372. URL: https://doi.org/10.1145/2461328.2461372.

[155] Ilya Tkachev and Alessandro Abate. "Formula-Free Finite Abstractions for Linear Temporal Verification of Stochastic Hybrid Systems". In: *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*. HSCC '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 283–292. ISBN: 9781450315678. DOI: 10.1145/2461328.2461372. URL: https://doi.org/10.1145/2461328.2461372.

[156] James Trevelyan, William R Hamel, and Sung-Chul Kang. "Robotics in hazardous applications". In: *Springer handbook of robotics* (2016), pp. 1521–1548.

[157] Paolo Tripicchio, Massimo Satler, Giacomo Dabisias, Emanuele Ruffaldi, and Carlo Alberto Avizzano. "Towards smart farming and sustainable agriculture with drones". In: *2015 International Conference on Intelligent Environments*. IEEE. 2015, pp. 140–143.

[158] Jana Tumova and Dimos V Dimarogonas. "Multi-agent planning under local LTL specifications and event-based synchronization". In: *Automatica* 70 (2016), pp. 239–248.

[159] Alphan Ulusoy, Stephen L Smith, Xu Chu Ding, Calin Belta, and Daniela Rus. "Optimal multi-robot path planning with temporal logic constraints". In: *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2011, pp. 3087–3092.

[160] Alphan Ulusoy, Stephen L Smith, Xu Chu Ding, Calin Belta, and Daniela Rus. "Optimality and robustness in multi-robot path planning with temporal logic constraints". In: *The International Journal of Robotics Research* 32.8 (2013), pp. 889–911.

[161] Cristian-Ioan Vasile, Derya Aksaray, and Calin Belta. "Time window temporal logic". In: *Theoretical Computer Science* 691 (2017), pp. 27–54.

[162] Cristian-Ioan Vasile, Derya Aksaray, and Calin Belta. "Time window temporal logic". In: *Theoretical Computer Science* 691 (2017), pp. 27–54.

[163] Cristian-Ioan Vasile, Jana Tumova, Sertac Karaman, Calin Belta, and Daniela Rus. "Minimum-violation scLTL motion planning for mobility-on-demand". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 1481–1488.

[164] Cristian-Ioan Vasile and Alphan Ulusoy. *LTL Optimal Multi-Agent Planner (LOMAP)*. `https://github.com/wasserfeder/lomap`. 2024.

[165] Juan Pablo Vielma and George L Nemhauser. "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints". In: *Mathematical Programming* 128 (2011), pp. 49–72.

[166] Vojtěch Vonásek, Martin Saska, Lutz Winkler, and Libor Přeučil. "High-level motion planning for CPG-driven modular robots". In: *Robotics and Autonomous Systems* 68 (2015), pp. 116–128.

[167] Yonah Wilamowsky, Sheldon Epstein, and Bernard Dickman. "Optimization in multiple-objective linear programming problems with pre-emptive priorities". In: *Journal of the Operational Research Society* 41.4 (1990), pp. 351–356.

[168] Zhe Xu and A Agung Julius. "Census signal temporal logic inference for multiagent group behavior analysis". In: *IEEE Transactions on Automation Science and Engineering* 15.1 (2016), pp. 264–277.

[169] Ruixuan Yan and Agung Julius. "A Decentralized B & B Algorithm for Motion Planning of Robot Swarms With Temporal Logic Specifications". In: *IEEE Robotics and Automation Letters* 6.4 (2021), pp. 7389–7396. DOI: `10.1109/LRA.2021.3098059`.

[170] Ruixuan Yan and Agung Julius. “Distributed Consensus-Based Online Monitoring of Robot Swarms With Temporal Logic Specifications”. In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 9413–9420.

[171] Ruixuan Yan, Zhe Xu, and Agung Julius. “Swarm signal temporal logic inference for swarm behavior analysis”. In: *IEEE Robotics and Automation Letters* 4.3 (2019), pp. 3021–3028.

[172] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. “A survey and analysis of multi-robot coordination”. In: *International Journal of Advanced Robotic Systems* 10.12 (2013), p. 399.

[173] Guang Yang, Calin Belta, and Roberto Tron. “Continuous-time signal temporal logic planning with control barrier functions”. In: *2020 American Control Conference (ACC)*. IEEE. 2020, pp. 4612–4618.

[174] Tiange Yang, Yuanyuan Zou, Shaoyuan Li, Xiang Yin, and Tianyu Jia. “Signal temporal logic synthesis under Model Predictive Control: A low complexity approach”. In: *Control Engineering Practice* 143 (2024), p. 105782.

[175] Jingjin Yu and Steven M LaValle. “Multi-agent path planning and network flow”. In: *Algorithmic foundations of robotics X*. Springer, 2013, pp. 157–173.

[176] Pian Yu and Dimos V Dimarogonas. “Distributed motion coordination for multirobot systems under LTL specifications”. In: *IEEE Transactions on Robotics* 38.2 (2021), pp. 1047–1062.

[177] Xinyi Yu, Xiang Yin, and Lars Lindemann. “Efficient STL Control Synthesis under Asynchronous Temporal Robustness Constraints”. In: *arXiv preprint arXiv:2307.12855* (2023).

[178] D. Zhou, Z. Wang, and M. Schwager. “Agile coordination and assistive collision avoidance for quadrotor swarms using virtual structures”. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 916–923.

Chapter 6

Vita

Gustavo Andres Cardona Calderon obtained his B.E. degree in Electrical Engineering at Universidad Nacional de Colombia in 2017. He obtained his M.S. degree in Industrial Engineering at Universidad Nacional de Colombia in 2023. He has been pursuing his Ph.D. in Mechanical Engineering and Mechanics at the Autonomous and Intelligent Robotics (AIR) Laboratory, Department of Mechanical Engineering, Lehigh University, working under the advisory of Prof. Cristian-Ioan Vasile since 2020. His research interests are in the planning and decision-making of multirobot systems. He served as a workshop organizer in IEEE ICRA 2024.