Safe and Interpretable Autonomous Systems Design: Behavioral Contracts and Semantic-Based Perception

Thesis by
Karena Xin Cai

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2021
Defended March 5, 2021

Karena Xin Cai
ORCID: 0000-0002-8392-4158

ACKNOWLEDGEMENTS

I would like to thank my advisor Richard Murray. He has been a bastion of wisdom and moral support for me throughout my graduate career. I feel so grateful that Richard gave me the space to explore new ideas and that he created a place for me at Caltech where I felt challenged but also always supported. He has greatly shaped my PhD and how I think about innovation, mentorship, and thoughtful collaboration.

I would also like to thank my advisor Soon-Jo Chung, who took me under his wings earlier in my graduate school career and brought me into the fold of his lab group. I was always in awe by his many suggestions and the ideas he proposed that challenged conventional wisdom. I cannot thank him enough for always going the extra mile to foster my intellectual curiosity, and to support and mentor me.

I would also like to thank K. Mani Chandy. His research has been a vital source of inspiration, and his words of encouragement have given me courage to pursue new things. I also want to express my gratitude to Joel Burdick for being my first mentor at Caltech. His reassuring words in those first years played a key role in giving me the strength to forge forward in graduate school.

I want to thank my collaborators Tung Phan-Minh and Alexei Harvard for working with me on intellectually stimulating and rewarding research directions. I would also like to thank the NCS and ARCL groups for always providing thoughtful feedback on my research and beyond.

I would not be who I am today without my family—Kathy, Stephanie, Jason, Mom, and Dad. My parents and siblings have always gone above and beyond in being sources of unconditional love and support for me. They have been there for me always, and for that I am immensely grateful.

I would be remiss to go without thanking my friends from all walks of my life who have kept me grounded and also made my time at Caltech fun and wholesome. I want to express my special gratitude for Rishita Patlolla, Richard Cheng, Zoila Jurado, Marcus Lee, and Audrey Hwang. They have been with me through the thick and thin and have helped me in more ways than I could have ever hoped for. I also want to thank Florian Schäfer and Mandy Huo, my officemates with whom I shared so many intellectually-stimulating conversations, excursions to CDS tea, much laughter, and long days and nights in the office. I would also like to thank Ke Yu, Apurva Badithela, Jennifer J Sun, Danilo Kusanovic, Joel Lawson, Kostas

Karapiperis, Tomo Oniyama, Benjamin Rivière, Guanya Shi, my friends from back home, and my Princeton friends for enriching my life with their company, laughter, support, and talks over good food about research and life.

ABSTRACT

We are on the verge of experiencing a new, integrated society where autonomous vehicles will become a fabric of our everyday lives. And yet, seamless integration of autonomous vehicles into our society will require vehicles to interface safely with humans in an incredibly complex, fast-paced, and dynamic environment. Premature deployment of these new autonomous systems—without safety guarantees or interpretability of algorithms, could prove catastrophic. How can algorithms governing vehicle behavior be designed in a way that guarantees safety, performance, interpretability and scalability? This is the question this thesis seeks to answer.

First, we present a framework for architecting the decision-making module of autonomous vehicles so that safety and progress of agents can be formally guaranteed. In particular, all agents are defined to act according to what is termed an assume-guarantee contract, which is broadly defined as a set of behavioral preferences. The first version of the assume-guarantee contract is a behavioral profile, which is a set of ordered rules that agents must use to select actions in a way that is interpretable. With all agents operating according to a behavioral profile, the interactions however, are not necessarily coordinated. We then constrain agent behavior with an additional set of interaction rules. The behavioral profile combined with these additional constraints, are what we term a behavioral protocol. With all agents operating according to a local, decentralized behavioral protocol, we can provide formal proofs of the correctness of agent behavior, i.e. all agents will never collide and agents will make it to their respective destinations. Not only does the protocol so defined allow us to make formal guarantees, but it is also designed in a way that scales well in the number of agents and provides interpretability of agent behaviors. Safety and progress guarantees are proven and verified in simulation.

Second, we focus on using information from object classifiers to enhance an autonomous vehicle's ability to localize where it is within its environment. The proposed approach for incorporating this semantic information is based on solving the maximum likelihood problem. With a hierarchical formulation, we are not only able to improve upon the accuracy of traditional localization techniques, but we are also able to improve our confidence in the accuracy of object detection classifications. The improvement in robustness and accuracy of these algorithms are shown in simulation.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] K. X. Cai, T. Phan-Minh, S-J. Chung, and R. M. Murray. "Rules of the Road: Safety and Liveness Guarantees for Autonomous Vehicles". In: *arXiv preprint arXiv:2011.14148* (2021). URL: https://arxiv.org/pdf/2011.14148v2.pdf.

K. X. C. participated in the conception of the project, developing the problem formulation and theoretical approach, working out the main proofs, coding the simulations, and writing the manuscript.

[2] K. X. Cai, A. Harvard, R. M. Murray, and S-J. Chung. "Robust Estimation Framework with Semantic Measurements". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 3809–3816. URL: https://ieeexplore.ieee.org/document/8814793.

K. X. C. participated in the conception of the project, developing the two semantic algorithms, coding the simulations, and writing the manuscript.

[3] T. Phan-Minh, K. X. Cai, and R. M. Murray. "Towards assume-guarantee profiles for autonomous vehicles". In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. Nice, France: IEEE, 2019, pp. 2788–2795. URL: https://ieeexplore.ieee.org/abstract/document/9030068.

K. X. C. participated in the conception of the project, developing the problem formulation and theoretical approach, and writing the manuscript.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Motivation

In the last few decades, there has been a significant push to innovate autonomous robotic technologies in industries ranging from transportation to healthcare. Because of the widespread availability of computational resources and recent advances in robotics, humans and robots will likely be able to operate side-by-side in the very near future.

The Uber and Tesla crashes, where software algorithms caused fatalities, are grim reminders that the algorithms that we design for these autonomous vehicles have very real and potentially catastrophic consequences [1, 39]. More of these incidents would cost lives and would justifiably obliterate human trust in robotic technologies. A prerequisite for introducing new technologies into society is thus providing compelling proof of their safety (showing there is an extremely low probability of the technology to cause harm or injury to humans), performance (the technology performs some functional task, presumably better than humans can), and interpretablility (there is an explanation for the reasoning agents use to make decisions).

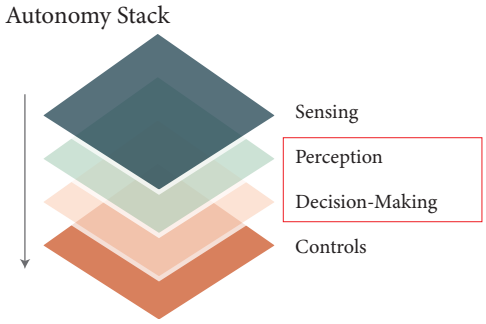

Figure 1.1: A typical autonomy stack. The thesis will focus on the perception and decision-making modules.

How can algorithms governing vehicle behavior be designed in a way that guarantees safety, performance, and interpretability? In this thesis, we provide solutions to this question in two different domains of a robot's autonomy stack: 1) the decision-making module, where agents make high-level behavioral choices and 2)

the perception module that agents use to localize and understand their surroundings, as can be seen in Fig. 1.1.

In the following sections, we pose the particular problems and challenges for designing safe and interpretable algorithms in each of these modules and present existing approaches.

1.2 Decision-Making

To define the agent's high-level behavior, we must define the strategy it uses to make decisions. The problem we address in this thesis is how to define a local, decentralized strategy for an agent that guarantees "correct" agent behavior. Here, correct behavior implies safety, lawfulness, and its ability to make progress. Other desirable properties of the decision-making module include scalability and interpretability.

Design of such a decision-making module is extremely challenging because 1) the set of actions agents are choosing will only partially satisfy some set of specifications, and it becomes unclear how actions should be ranked relative to each other [18], 2) the robot-freezing problem, in which the uncertainty propagation of other agent behaviors can cause the robot to freeze altogether [91], 3) the joint action space grows exponentially with every additional agent and quickly becomes computationally intractable, and 4) safety and progress proofs for these decision-making modules are extremely difficult to formalize because of unbounded rationality.

There are two main approaches in the multi-agent literature in defining the decision-making module for autonomous agents. The first more common approach involves designing a decision-making module for an individual agent, with some assumed behavioral model for how other agents will interact with it. The second involves a more collective approach where the focus is on designing efficient conventions or protocols that specify how agents should negotiate and resolve conflicts while interacting. The assumption here is that all agents are acting according to these social laws or conventions. In the following sections, we provide details of these approaches and their inherent limitations.

Behavior-Aware Motion-Planning for Individual Agents

Agents need to engage in interactive and cooperative decision-making in order to choose actions that are safe, but also not maximally prohibitive and prevent the agent from moving altogether like in the robot freezing problem [91]. How do agents make decisions while taking into account the complex inter-dependencies

that emerge from interactions with other agents?

Interacting Behavioral Models

One approach to solving this problem has been to use prediction methods to form a set of possible hypotheses other agents may take [43, 53, 55]. Contingency motion-planning based on these predictions has been proposed to guarantee that agents will be able to take coordinated actions while retaining a safe trajectory [38]. These approaches do not fully accommodate for reactivity of agent behaviors and tend to be overly conservative. A second approach has been to use probabilistic models to capture the interactivity among a set of self-interested agents. By modeling agents as interacting Gaussian processes, some level of agent reactivity is captured. These methods, however, lack performance guarantees [91]. A third approach has been taking a game-theoretic perspective and modeling agent decision-making with respect to a Markov game (and in the case of partially-observable information) as interacting partially-observable Markov Decision Processes (i-POMDPs) [10, 32]. These methods often capture the reactivity of agents by modeling a reward function defined on a joint action space, but this joint action space grows exponentially in the number of agents. Methods like using factor-graph Markov Decision Processes or imposing additional assumptions on agents have been proposed to reduce the complexity of the problem [34, 81, 56]. All the interactive models of agent behavior and associated decision-making strategies are limited because they do not scale well and fail to provide either safety or performance guarantees.

Data-Driven Methods

A wide breadth of data-driven methods have been proposed so autonomous agents can learn how to make decisions in an interactive multi-agent setting. Gaussian Mixture Models, parameterized by neural networks, have been used to predict the coupled behaviors of vehicles in a highway setting based on features like the ego vehicle state, surrounding vehicle past and current states, specifications and road geometry [43]. A large thread of research has been on using inverse reinforcement learning (IRL) to learn the reward function that agents are using to make behaviorally-aware decisions, and design control strategies for the ego agent accordingly [75, 77, 78]. Maximum-entropy IRL methods, a form of learning where the entropy-based cost-function reduces the chance of overfitting, is used to help agents learn human behavior and perform motion-planning in a socially-compliant

manner [40, 48, 71]. In the case where agent transition functions and their joint reward functions are not known, reinforcement learning algorithms can provide a way to learn best policies in a multi-agent game with self-interested agents [12, 44, 84]. These approaches are often based on the joint action space and may not always guarantee convergence to an optimal equilibrium policy. Furthermore, decision-making policies defined based on any learned models, lack interpretability and do not allow for providing formal guarantees on the safety or performance of the decision-making modules.

Formal Methods

Formal methods offer tools for designing provably correct control strategies for complex systems like autonomous vehicles that satisfy high-level behavioral specifications like safety and liveness [6] for each individual vehicle. Linear temporal logic (LTL) and signal temporal logic (STL) are used to define formal specifications or rules agents should follow, and correct-by-construction controllers are then synthesized to satisfy these specifications [3, 96]. Oftentimes agents will not be able to satisfy all specifications at once and there may exist many conflicting specifications the agent must follow [18, 92]. Minimum violation motion-planning has been proposed to help the vehicle choose the trajectory that minimizes violation of a set of ordered rules [92, 93, 98]. Unfortunately many of these algorithms rely on a joint product automata which is inherently hard to scale with an increasing number of agents. Recent work has been shown to reduce this computational tractability with additional assumptions on the types of specifications [97]. Even so, the algorithms used for synthesizing formally-correct strategies for the vehicles cannot guarantee global safety properties since they do not make the assumptions that must hold on other vehicle behaviors explicit.

Social Laws and Conventions for a Collective Set of Agents

A separate branch of research for designing decision-making modules has been focused on the entire collective set of agents instead of individual agents. These methods suggest defining behavioral restrictions or protocols for all agents to follow so agent motion becomes coordinated and emergent global properties like safety can be guaranteed [9, 85]. Shoham and Tenneholtz defined the idea of social laws, which are a set of behavioral conventions that guarantee safe coordination of agents [85]. Alternating temporal logic model checkers can be used to solve related feasiblity and synthesis problems based on these social laws [23]. The complexity

of designing social laws, however, has been shown to be NP-complete [85]. Other conventions include defining a lexicographical ordering over the joint action space and choosing an action based on the ordering, but this requires substantial overhead about knowledge of other agents [9, 10]. The Responsibility-sensitive-safety (RSS) framework proposed by Mobile-Eye is a similar approach in that it defines a set of rules or behaviors all agents should satisfy to guarantee safety [83]. This framework, however, is very scenario dependent and does not guarantee that agents will be able to make progress.

Alternative methods focus on designing communication protocols and mechanism design via actions to facilitate coordinated agent behaviors. For instance, Pearl in [69] suggests message passing via coordination graphs, where agents repeatedly send messages to neighboring agents to establish coordinated behaviors. Auction-based methods designed for resource allocation are those in which incentive structures are defined to establish global properties of the collective system [17, 67, 94]. Other auction-based methods solve the assignment problem to coordinate agents in a decentralized way [8, 61]. These types of methods require some infrastructural overhead and often suffer from the state-space explosion problem, but are powerful in that they allow more global guarantees to be defined on the entire system.

The Approach

We can see that existing methods offer a variety of approaches that partially solve the proposed problem, but there are none that propose a fully comprehensive decision-making module that guarantees properties like safety and liveness (progress) for all agents.

In our approach, we propose a framework that bridges a lot of elements of the above theory and ideas in the literature. The main distinction of our proposed architecture from the existing literature is the shift from thinking of each agent as separate, individual entities, to agents as a collective where *all* agents adopt a *common* local, decentralized protocol (where additional customization can be built in later). In particular, we define a specific behavioral contract for one type of agent class in a specific type of road network environment. The behavioral contract assumes agents are constrained to reason about agents in a local region about it and that they have minimal communication (i.e. token querying) capabilities. If all agents are following this protocol, then we can guarantee safety and progress for all agents for certain road network conditions. Moreover, the protocol is interpretable because

the structure imposed on the set of specifications the agent is following is defined in a particular way that guarantees transparency in the way agents select actions.

The work builds off of local, decentralized algorithms proposed for the solution to the drinking philosopher problem [19]. The details of the approach can be found in Chapter 1 and 2. Unlike existing works, the result of our work is a behavioral protocol that: 1) defines a set of behavioral specifications where the behavior is interpretable (explainable), 2) leverages road network structure, 3) allows for inertial vehicle dynamics, 4) includes a notion of locality, 5) is scalable with respect to the number of agents (because of the invariance of agents' safety backup plan action), and 6) can formally guarantee safety and liveness.

Summary of Contributions

We define a behavioral protocol that specifies the set of rules agents should use to ultimately select their actions. For this behavioral protocol, we introduce:

1. A behavioral profile that orders a set of agent specifications in such a way that actions are chosen in an interpretable manner.
2. A quasi-simultaneous discrete game, a turn-based game in which turn-order of agents is dependent on agent states.
3. A conflict resolution scheme based on agent precedence and token-querying.
4. Safety proofs based on all agents always having a safe backup plan action.
5. Progress proofs based on inductive reasoning where all agents are shown to always eventually take a forward progress action.

1.3 Perception

The role of the perception module in the autonomy stack is to allow for an agent to have some representation of its environment—where other agents are located, and where it is located within this environment. In this thesis, we focus on the localization problem. Conventional simultaneous localization and mapping (SLAM) algorithms typically rely on geometric measurements. Access to semantic information (like what objects are and/or contextual knowledge about the environment) can enrich an agent's perceptual understanding of its environment and where it is within that environment.

Vision-based object classifiers, combined with an a-priori map of object locations, offer a way to improve existing pose estimation methods. Augmenting existing localization methods with semantic information from these classifiers can be challenging because 1) the measurement model is inherently different than more traditional continuous measurements and 2) the algorithm must be robust to false positive and negative measurements.

In the following sections, we provide some more context on traditional SLAM methods and ways traditional methods have been augmented to accommodate semantic information.

Traditional SLAM Methods

Standard SLAM algorithms are posed as the following problem: given a set of measurements $\mathscr{Z} \triangleq \{z\}_{t=0}^{T}$, find the best estimate for both the vehicle poses $\mathscr{X} \triangleq \{x\}_{t=0}^{T}$ and the landmark positions $\mathscr{L} \triangleq \{l\}_{t=0}^{T}$ seen in the environment during the vehicle trajectory [13]. Batch estimation techniques are when all pose-estimate and landmark positions are solved for at once. The maximum likelihood formulation turns into a nonlinear least-squares minimization problem under some Gaussian assumptions about the model and measurement noise and can thus be solved via QR-factorization. The gtsam algorithm in particular is a widely-used pose-graph estimation algorithm that can solve the nonlinear batch-optimization formulation in an incremental fashion [22]. Solving the optimization problem is often referred to as solving the 'back-end' of the SLAM problem. The front-end of the SLAM problem involves solving the data association problem (matching measurements to landmarks in the map) and the loop-closure problem (identifying when an agent has reached its original location) [22]. Several algorithms have been developed to ensure that the factor-graph formulation for SLAM problems are robust to loop-closure errors [16, 31, 87]. In particular, the covariance value associated with loop closure measurements, which are referred to as switchable constraint variables, are introduced into the optimization framework to improve robustness to false detections [2, 87]. These same methods can be extended to make semantic estimation algorithms robust to false positive measurements as well.

Semantic SLAM Methods

Semantic data can be modeled as binary measurements that have state-dependent probabilistic likelihood functions [5, 45, 46]. The probability of a positive detection measurement is modeled as an inverse exponential function of the distance to the

detected object in [45], meaning positive object detections are modeled to occur with higher probability when the vehicle is close to the detected object. In [11], Bowman et al compute the likelihood function of an object detection event as a function of the object classifier confusion matrix, and solve the coupled data association and estimation problem by iteratively solving an expectation-maximization problem. In these algorithms, however, the likelihood functions lack the ability to capture false positive or negative detections. A likelihood function that captures these types of errors is derived in [5], but requires additional assumptions on the probability of false positive detections generated by clutter.

While semantic measurements have been shown to improve the accuracy, there has been an active effort or research in actively leveraging vehicle dynamics to enhance its ability to recognize objects [68, 74, 89]. Pose estimation has been shown to improve the accuracy of object classification algorithms [68, 74]. These methods take into account the motion profile taken when observing an object, but do not consider the dynamics between object detections [74].

The Approach

Existing methods for semantic estimation either use semantic measurements to improve the accuracy of localization methods or use the vehicle dynamics to improve the accuracy of object classifiers—but few do both simultaneously. In our approach, we fully leverage the information offered by object classification information. In particular, we introduce a hierarchical framework where both object-detection events and the object state are estimated. The hierarchical framework that we propose robustly handles false positive measurements used in localization and also allows for our greater certainty in the object classification measurements. This framework enables the agent to have a more enriched semantic understanding of where it is located.

Summary of Contributions

We define two methods for incorporating semantic measurements into more traditional SLAM algorithms:

1. A maximum likelihood framework with semantic information that is shown to improves both the accuracy and the robustness of traditional SLAM algorithms in simulation.

2. A hierarchical estimation framework, where object detection events as well

as robot poses are estimated. In this framework, robustness and accuracy of the estimation, along with the confidence of object classifiers, are improved.

Chapter 2

DECISION-MAKING: BEHAVIORAL PROFILES

2.1 Introduction

In this chapter, we focus on designing the decision-making module for autonomous vehicles with the specific focus of designing the set of rules (and the ordering on this set of rules) that agents use to select their actions. Although the theory introduced here can be applicable to any sets, we focus on the particular use-case of sets of specifications (for self-driving cars).

Fundamental to defining agent behavior is deciding which rules or specifications agents must follow. While defining the set of rules agents should follow is quite easy, it is often the case that agents cannot always choose actions that satisfy all actions simultaneously. It thus becomes necessary to define the priority among each of these specifications. Currently, rules or specifications for autonomous vehicles are formulated on a case-by-case basis, and put together in a rather ad-hoc fashion [38, 54].

As a step towards eliminating this practice, we propose a systematic procedure for generating a set of specifications for self-driving cars that are 1) associated with a distributed assume-guarantee structure and 2) characterizable by the notion of consistency and completeness. The behavioral profile, which is a product of this systematic procedure, is a mathematical structure on the set of specifications that ultimately defines a version of a rulebook that agents can use to transparently select their actions. The work presented in this chapter has been published in [72], and was done in joint collaboration with Tung Phan-Minh.

2.2 Assume-Guarantee Profiles

In a dynamic and interactive environment, the problem of providing guarantees for a single agent without making any assumption on the behaviors of other agents is ill-posed. The inherent coupling between the assumptions on the environment and the system's guarantees can be seen in Fig. 2.1. Assume-guarantee contracts provide a formalism for defining contracts between modules of a complex system so that system-level specifications can be met if the contracts among the individual modules are satisfied.

In the case of autonomous systems, each agent can be seen as an individual module of the complex, multi-agent system. In this setting, we propose that this assume-guarantee contracts should be defined in the form of a set of behavioral preferences or rules that agents adhere to when selecting actions to take. Specifically, we propose the framework of assume-guarantee profiles as follows.

Definition 1 (Assume-guarantee profile)**.** An *assume-guarantee profile* for an agent is a 2-tuple $(\mathcal{A}, \mathcal{G})$ where

- $\mathcal{A}$ is a set of behavioral preferences or characteristics that the agent assumes its environment to have.
- $\mathcal{G}$ is a set of behavioral preferences or characteristics that it is obligated to behave according to as long as its environment makes decisions in accordance with $\mathcal{A}$.

Autonomous Vehicle Profile for Individual Agent

The first version of a behavioral contract for an autonomous agent that we propose is a profile each agent uses to select actions. The purpose of the behavioral profile is to define the behavioral preferences by ranking different actions (in most cases, trajectories) relative to one another so that the optimal one can be selected. Ideally, there would always be an action that the agent could select that would satisfy all agent specifications.

Realistically, however, an action that satisfies all specifications does not always exist. The individual behavioral profile therefore must have both 1) a set of specifications (or rules) that agents should follow, but also 2) an ordering that defines a hierarchy of importance on the specifications. As shown in Fig. 2.1, the functionality of a profile serves to distinguish which action among a set of actions, has the highest-priority with respect to some set of ordered specifications.

To order these actions, it is first necessary to know which subset of specifications each of these actions satisfies. In other words, the agent needs a way to resolve whether an action satisfies or does not satisfy each of the relevant specifications. This is the role of the oracle, defined as follows.

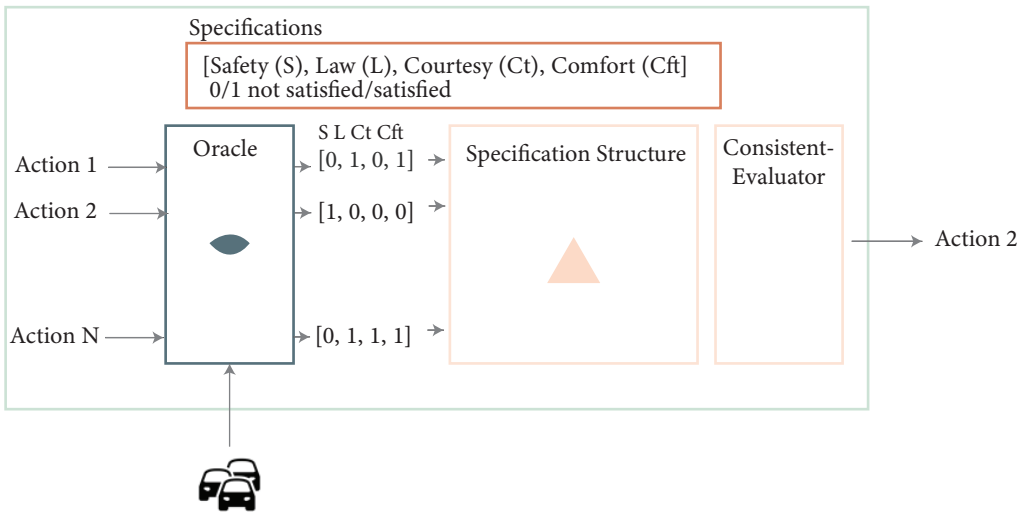

Figure 2.1: A high-level system architecture capturing the inherent coupling of the behavioral specifications for an agent and its environment is shown in the bottom figure. Each agent identifies the best action to take based on which subset of rules are satisfied by that action. The specification structure and consistent evaluating function define a unique ordering on all subsets of rules that ultimately determine which actions are better or worse than others.

We assume that each autonomous agent relies on an *oracle* [86] that provides predictions as to which set of specifications of interest that a particular action will satisfy. The input to the oracle is a set of specifications, a potential action the car can choose to take, and the current world state configuration. The output of the oracle is a prediction of what specifications will be satisfied if the action is taken. In the simplest case, the oracle could return a valuation of a set of a Boolean variables, each indicating whether or not a property is violated. Note, these specifications could be refactored to accommodate specifications that are more continuous in nature (e.g. speed limit, extent of lane violation, etc.). Although many decision/optimization problems currently posed for autonomous vehicles are of high computational complexities, not to mention undecidable [59, 66], we expect future technology to be capable of approximating the oracle to an acceptable level of fidelity.

The oracle thus maps each action to a subset of specifications that are satisfied if an agent were to take that action. The remaining role of the behavioral profile is to evaluate how each action (subset of satisfied specifications) compares to every other action (subsets of satisfied specifications), allowing the agent to ultimately select the most preferable action. In order to perform this evaluation, we define a

mathematical object termed a *specification structure* that imposes a hierarchy on a set of specifications.

The hierarchy on the set of specifications cannot be partially defined, as this would lead to a lack of transparency and interpretability behind agent decisions and would thus undermine the purpose of defining a behavioral contract in the first place. On the other hand, defining a total order on the powerset of the set of specifications would not be amenable to computational tractability and would greatly restrict agent behaviors. In Section 2.3, we thus introduce something in between the two. We define a set of rules that must hold on the set of specifications so the profile is minimally restrictive (still allows for customization), clearly defines a unique line of reasoning for the agent's decision-making process, and can be defined in a computationally tractable and intuitive way. The specific mathematical properties that the specification structure must satisfy for these characteristics to hold are defined more rigorously in the following section.

2.3 The Specification Structure and Consistency

In order to motivate the definition of properties that must hold on the ordering imposed on the specifications, we first consider an example where there exists a partial order on a set of specifications. We show that the partial order on the set of specifications is not enough to define a unique and transparent ordering on the powerset of specifications. This would imply that there would be many ways to rank different actions to each other, leading to inconsistencies in agent behavior.

Example 1. Consider a set $S = \{a, b, c, d, e\}$ that is partially ordered (a poset) such that $b \prec a$, $c \prec a$, $d \prec c$, and $e \prec c$. Here, each element in the set represents a specification like safety, the law, performance, etc. By this partial order, the node b cannot be compared to c or d or e. Since b cannot be compared to c or d or e, it is ambiguous whether a self-driving car should take an action that satisfies the properties a, b, and d or an action that satisfies a, b, and e. More specifically, there exist multiple ways to rank subsets relative to one another, thereby causing an ill-posed definition of agent behavior.

The above example illustrates that there is not enough mathematical structure on this partially-ordered set to resolve the ambiguity on the set of all possible subsets of specifications. In order to define what additional constraints must hold on the ordering for the rankings to be well-posed, we first introduce the idea of *consistent evaluators*, which are a class of functions that endow a partially-ordered set (in our

case, of specifications) with a unique *weak* order on their powersets. Being weakly ordered means that all subsets are comparable, but some subsets may have equal values to each other (these are considered indistinguishable).

We argue that a weak order on the powerset of specifications is preferable to imposing a total order. In a practical setting, if a self-driving car manufacturer wanted to impose a total order instead of a weak order on the powerset, they would have to face the challenging task of defining how any one set of specifications is strictly better or worse than another set of specifications. This is arguably impractical not only because of the exponential growth in the size of the powerset, but also because sometimes a *strict* comparison among sets of properties is simply unnecessary. A consistent evaluator, which allows for sets in the powerset to have equal value, therefore allows for a more sensible way of resolving comparisons between subsets of specfications.

In summary, the role of the consistent evaluating function is to take as input some partially-ordered set of specifications. It then defines a weak order on the powerset of that set of specifications, thereby ranking every subset of specifications to every other subset of specifications in a unique way. In the way the consistent-evaluator function is defined, the ranking of subsets will respect the partial-order on the specifications.

We refer to maximal chains (antichains) in our definitions and proofs, so we present the definitions here.

Definition 2 (Maximal Chain (Antichain))**.** A *chain (antichain)* is a subset of a partially ordered set such that any two distinct elements in the subset are comparable (incomparable). A *chain (antichain)* is *maximal* when it is not a proper subset of another chain (antichain).

In order for the consistent-evaluator function to be a well-posed function, we define it as follows:

Definition 3 (Consistent evaluator)**.** Given a set of specifications $\mathcal{P}$ and its powerset $2^{\mathcal{P}}$, we define the consistent evaluator function as $f : 2^{\mathcal{P}} \to T$, where T is a totally ordered set with $\leq$ as an ordering relation. For all subsets, $P_1, P_2 \subseteq \mathcal{P}$, the following must hold:

1. $\forall p_1 \in P_1, P_1 \neq \varnothing \Rightarrow f(\varnothing) < f(P_1)$;

2. $\forall p_1 \in P_1.\forall p_2 \in P_2.p_1 \in P_1 \wedge p_2 \in P_2 \wedge f(\{p_1\}) = f(\{p_2\}) \Rightarrow (f(P_1) \leq f(P_2) \Rightarrow f(P_1 - \{p_1\}) \leq f(P_2 - \{p_2\}))$*; and*

3. $(\forall p_1 \in P_1.\forall p_2 \in P_2.f(\{p_1\}) \neq f(\{p_2\})) \Rightarrow (\max_{p \in P_1} f(\{p\}) < \max_{q \in P_2} f(\{q\}) \Rightarrow f(P_1) < f(P_2))$.

If $\mathcal{P}$ is partially ordered by $\preceq$ and $\mathfrak{A}_{(P,\preceq)}$ is the set of all antichains of P, we further require that for any $p_1, p_2 \in P$

4. $\forall p_1 \in P_1.\forall p_2 \in P_2.p_1 \prec p_2 \Rightarrow f(\{p_1\}) < f(\{p_2\})$*; and*

5. $\forall p_1 \in P_1.\forall p_2 \in P_2.(\{p_1, p_2\} \in \mathfrak{A}_{(P,\preceq)} \wedge f(\{p_1\}) < f(\{p_2\})) \Rightarrow (\exists s, t \in P.p_1 \prec s \wedge f(\{s\}) = f(\{p_2\}) \wedge f(\{p_1\}) = f(\{t\}) \wedge t \prec p_2)$.

Intuitively, the conditions in Definition 3 mean

1. The evaluator will assign the worst value when no property is satisfied. This ensures that every property included in $\mathcal{P}$ matters to the evaluator.

2. Properties of equal value to the evaluator can be disregarded without affecting the result of the evaluation.

3. For sets that do not have properties with the same values, the one with the most highly valued property is preferable.

4. If there exists a pre-imposed hierarchy between some of the properties via some partial ordering, then the evaluator must respect it.

5. Given a pre-imposed hierarchy on the properties, the evaluator must be impartial: it will only assign different values to two properties whose relationship is not defined in the hierarchy when they are comparable via two equally valued "proxies." In Condition 5, the proxies are s and t .

First, we present an example of a partially-ordered set and its respective consistent-evaluation function. In particular, the consistent-evaluating function is a function that takes in a partially-ordered set and outputs a weak order on the powerset of specification, thereby allowing us to rank the subsets of specifications relative to one another—in what we later prove to be a unique way.

Example 2. Consider a partially ordered set Q in which p is the greatest element and all other elements belong to an antichain. Then we can define f as the function $f(\tilde{Q}) \coloneqq \mathbb{1}_{p\in\tilde{Q}}|Q| + |\tilde{Q} - \{p\}|$ for all $\tilde{Q} \subseteq Q$ where $\mathbb{1}$ is the indicator function. This function evaluates any subset with the maximal element in it as the cardinality of Q plus the dimension of the subset not including the element p. It also evaluates any subset without the maximal element as the dimension of that subset. One can easily verify that f is a consistent evaluator for Q.

Do all partially-ordered sets have a consistent evaluator? The answer is no. We present an example to show that not all posets admit a consistent evaluator.

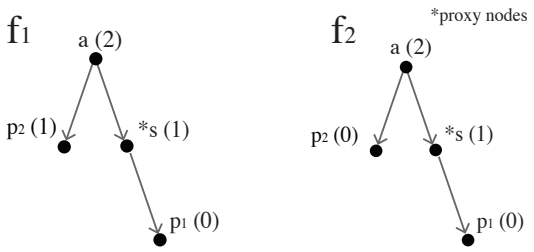

Figure 2.2: A poset that does not admit a consistent evaluator. The values in parentheses denote the value of the singleton set containing that node given by the evaluator f_i. In both cases, requirement 5 is violated. © 2019 IEEE.

Example 3 (Poset without consistent evaluator)**.** Consider the poset with the structure shown in Fig. 2.2. We cannot define a consistent evaluator that satisfies all five requirements on this partially-ordered set. In order to satisfy requirement 4, such that the partial order established in the poset is preserved, the consistent evaluator function f_1 on the left and f_2 on the right can, WLOG, assign all nodes on the right branch of the poset with the values shown in Fig. 2.2. To respect the partial order, by requirements 4 and 5 of consistent evaluators, p_2 must be assigned a value in $\{0, 1\}$. The left figure shows what happens if the function takes on the value 1 for the left node, i.e., $f_1(\{p_2\}) = 1$. If this happens, then $f_1(\{p_1\}) < f_1(\{p_2\})$, but there is no proxy node that is comparable to p_2 in the poset and has a value equal to $f_1(\{p_1\})$. This clearly violates requirement 5. The right figure shows what happens if the function takes on the value 0, i.e. $f_2(\{p_1\}) = 0$. A similar violation is incurred by f_2 (see Appendix A).

Through the previous example, we see that not all posets admit a consistent evaluator. The natural question to ask is: what makes posets consistently evaluable? The next theorem will answer this question.

Theorem 1. *A finite poset P of specifications has a consistent evaluator if and only if it can be partitioned by a set $\mathcal{A}$ of N maximal antichains such that*

1. *Maximal Antichain and Rank Criterion: The maximal antichains $\mathcal{A}$ can be assigned ranks in such a way that the partial order is respected.*
2. *The Maximal Chain Criterion: For each node (dimensional property), there exists a maximal chain containing the node of length N.*

The proof of Theorem 1 is as follows:

Proof: ($\Rightarrow$): Suppose that P is a poset of specifications with the ordering relation $\preceq$ such that P has a consistent evaluator f. Since P is finite, the set $f_P \coloneqq \{f(\{p\}) \mid p \in P\}$ is also finite. Furthermore, the range of f being totally ordered implies that we can write $f_P = \{z_1, z_2, \ldots, z_n\}$ for $n = |f_P|$ such that $z_1 < z_2 < \ldots < z_n$. For each $z_i \in f_P$, let $f^{-1}(z_i) \subseteq P$ be defined by $f^{-1}(z_i) \coloneqq \{p \mid p \in P \wedge f(\{p\}) = z_i\}$. Observe that requirement 4 of Definition 3 implies that for each i, $f^{-1}(z_i)$ is an antichain. Consequently, the $f^{-1}(z_i)$'s form a partition of P by antichains. By ranking each $f^{-1}(z_i)$ by the corresponding z_i, it also follows that the antichains respect the partial order defined by $\preceq$. To show maximality, suppose that there exists $j \in [n]$ such that $f^{-1}(z_j)$ is not a maximal antichain. This implies that there exists $k \in [n] - \{j\}$ and there is a property $q^\star \in f^{-1}(z_k)$ such that $\{q^\star\} \cup f^{-1}(z_j)$ is an antichain. WLOG, suppose $j < k$ so that $z_j < z_k$ implies $\forall q \in f^{-1}(z_j).f(q) < f(q^\star)$. But the existence of any $\tilde{q} \in f^{-1}(z_j)$ such that $f(\tilde{q}) < f(q^\star)$ implies, by requirement 5 of Definition 3, that there exists $q' \in f^{-1}(z_j)$ such that $q' \prec q^\star$. But this contradicts the assumption that $\{q^\star\} \cup f^{-1}(z_j)$ is an antichain. From this, we hence conclude that 1) holds. To see that 2) holds, observe that any property $p \in f^{-1}(z_j)$, if $j \neq n \wedge n \geq 2$, then by requirement 5 and the antichain property of the $f^{-1}(z_i)$, there exists $q \in f^{-1}(z_{j+1})$ such that $p \prec q$. Similarly, if $j \neq 1 \wedge n \geq 2$, there exists $r \in f^{-1}(z_{j-1})$ such that $r \prec p$. Applying this argument to r and/or q inductively yields a chain of length n that contains q. This chain is maximal by the contradiction resulting from applying the pigeonhole principle [36] to the assignment of properties from any chain of greater length to the maximal antichains.
($\Leftarrow$): Let the maximal antichains in the partition of P be $P_0, P_1, \ldots, P_{m-1}$ with ranks $r(P_0) < r(P_1) < \ldots < r(P_{m-1})$. We construct a function $W : 2^P \to \mathbb{N}^m$ as follows. For any subset $S \subseteq P$, we define $A_{S,r}$ to be the set of all elements in the subset S with rank r. $W(S) \in \mathbb{N}^m$ is an (m)-tuple whose ith digit is such

that $W_i(S) := |A_{S,i}|$. This means that the ith element in the tuple is the number of elements in the subset S with rank i.

The ith digit of $W(S)$ is defined to be more significant than the jth digit if the former is associated with a higher rank. This induces a natural total ordering relation $\leq$ on the set $\{W(P) \mid P \subseteq \mathcal{P}\}$ by most significant digits. In particular, this means, $W(S_a) \leq W(S_b)$ if and only if all corresponding entries are equal or the first most significant differing pair satisfies $W_i(S_a) < W_i(S_b)$. The rest of the proof involves checking that W has all the properties of a consistent evaluator.

We can easily verify that requirements 1-4 of a consistent evaluator are satisfied. Now, we show that requirement 5 holds as well. Let us show this by contradiction. Consider that there exists a node p_1 and p_2 such that $f(\{p_1\}) < f(\{p_2\})$, but there does not exist a node s or t such that $p_1 \prec s$, $t \prec p_2$, and $f(\{p_2\}) = f(\{s\})$ and $f(\{p_1\}) = f(\{t\})$. WLOG, consider p_1 to be a node where there does not exist a node s such that $p_1 \prec s$ and $f(\{p_2\}) = f(\{s\})$. Since there is no node that is directly comparable to p_1 in the antichain with value equal to $f(\{p_2\})$, there exists a maximal chain containing p_1 that has length strictly less than m. This is a violation of property 2) characterizing the poset P.

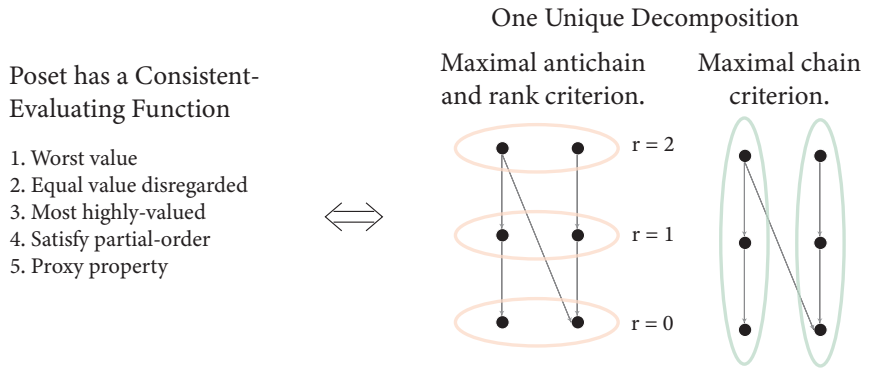

Figure 2.3: A visualization of Theorem 1.

Is it possible that there may be multiple such decompositions of maximal antichains, making the ordering that is induced via the corresponding rankings not unique? Luckily, the answer is a reassuring negative.

Theorem 2. *Such a partition in Theorem 1 is unique.*

Proof: Suppose that $P_1, P_2, \ldots, P_m$ is also a partition of maximal antichains of P with ranks $r(P_1) < r(P_2) < \ldots < r(P_m)$ that respect the partial order. Suppose

that $m \neq n$ where $n = |f_P|$. If $m > n$, then by 2) of Theorem 1 there is a chain of length m. However, assigning these m properties to the $f^{-1}(z_i)$ means by the pigeonhole principle that there are at least two properties that are assigned to the same $f^{-1}(z_j)$ for some j, implying that $f^{-1}(z_j)$ is not an antichain, a contradiction. It follows that $m \leq n$. Similarly, we can argue that $m \geq n$ and therefore $m = n$. Now, we claim that $P_i = f^{-1}(z_i)$ for all $i \in \{1, 2, \ldots, n\}$. Suppose this is not the case, then there exists $p \in P_k$ such that $p \in f^{-1}(z_h)$ for $k \neq h$. Then by 2), there are two chains of length n: $p_1 \prec p_2 \prec \ldots \prec p_n$ and $f_1 \prec f_2 \prec \ldots \prec f_n$ such that $p_i \in P_i$ and $f_i \in f^{-1}(z_i)$. We also have $p_k = f_h = p$. WLOG, assume $h < k$. This implies that $p_1 \prec p_2 \prec \ldots \prec p_k = p = f_h \prec f_{h+1} \prec \ldots \prec f_n$. However, this chain has length $k + n - h > n$ since $k > h$. This contradicts the fact that P can be partitioned into n antichains.

Since the consistent-evaluating function is defined on this uniquely-defined decomposition, there is only one unique weak order induced on the powerset of specifications. The uniqueness of the weak order means there is only a single way to rank subsets of specifications relative to one another up the the weak order. The line of reasoning the agent uses for selecting a given action, based on this profile, is well-defined and therefore interpretable. In the following theorem, we show that all consistent-evaluator functions defined on this partial-order have to assign the same, unique weak-order on the partially-ordered set.

Consistency

The notion of consistency comes from Theorem 3, which says that there is a unique weak order on the powerset of a specification structure regardless of the consistent evaluator used.

Theorem 3 (Consistency implies uniqueness)**.** *If $\mathcal{P}$ is a poset with an ordering relation $\preceq$ that can be consistently evaluated, then all consistent evaluators of $\mathcal{P}$ are equivalent. That is, for any pair of consistent evaluators f_a, f_b of $\mathcal{P}$, for all $P_1, P_2 \subseteq \mathcal{P}$, we have $f_a(P_1) \leq f_a(P_2) \Leftrightarrow f_b(P_1) \leq f_b(P_2)$.*

Proof: By symmetry, it is sufficient to prove the ($\Rightarrow$) direction. Suppose $f_a(P_1) \leq f_a(P_2)$. Now since $\mathcal{P}$ is consistently evaluable, by Theorems 1 and 2, it can be partitioned by a unique set of maximal antichains $\{A_r\}_{r=1}^{R}$. By requirement 4 of Definition 3, one can show that any consistent evaluator will rank these antichains the same way. Namely, any consistent evaluator f of $\mathcal{P}$ can be assumed, WLOG, to satisfy the following conditions

1. $f(\{p_1\}) < f(\{p_2\}) < \ldots < f(\{p_R\})$, for $p_r \in A_r, r \in \{1, \ldots, R\}$,
2. $f(\{q_i\}) = f(\{r_i\})$, for any $q_r, r_r \in A_r, r \in \{1, \ldots, R\}$.

Condition 2 above implies that all pairs of nodes that are of equal value to f_a are also of equal value to f_b and vice versa. So by requirement 2 of Definition 3, we can assume that P_1 and P_2 do not overlap in property values due to either f_a or f_b. If $P_1 = \emptyset$, then by requirement 1, we have $f_b(P_1) < f_b(P_2)$. Otherwise, by requirement 3, let $p_i^\star \in P_i$ be the property that maximizes the value of f on P_i. We have $f_a(P_1) \leq f_a(P_2)$, which implies that $p_1^\star \prec p_2^\star$ since $f_a(\{p_1^\star\}) \neq f_a(\{p_2^\star\})$ due to P_1 and P_2 not overlapping in property values, and therefore $f_b(P_1) < f_b(P_2)$.

We have thus defined the necessary and sufficient properties that must hold on a partially-ordered set of specifications such that it is consistently-evaluable, i.e. subsets of specifications can be ranked in a unique, weak order. Note, we use the term specification structure interchangeably with consistently-evaluable posets. We believe that the criteria for determining whether a partially-ordered set is consistently-evaluable is not particularly intuitive, so we introduce a simpler class of consistently-evaluable posets in the following section.

A Simpler Class of Consistently-Evaluable Posets

Reasoning about whether a partially-ordered set is consistently-evaluable is non-trivial. For example, imagine that a designer adds or removes a pairwise comparison between a pair of specifications. It is not clear to see whether the criteria of consistently-evaluable posets still holds. We therefore introduce a simpler class of partially-ordered sets which we show satisfy the criteria for being consistently-evaluable, but are easier to reason about. The clarity comes at the cost of being more restrictive (i.e. they are a subset of consistently-evaluable posets). This can be seen in Fig. 2.4. This simpler class of partially-ordered sets are what are termed graded posets.

Definition 4 (Graded Specification Poset)**.** A *graded specification poset* is a finite, graded, poset of specifications $\mathcal{P}$. Namely, if $\preceq$ is the ordering relation for $\mathcal{P}$ and $\prec$ is the strict version thereof satisfying $x \prec y \Leftrightarrow (x \preceq y \wedge x \neq y)$, then there exists a ranking function $\rho : \mathcal{P} \to \mathbb{N}$ such that

1. $p_1 \prec p_2 \Rightarrow \rho(p_1) < \rho(p_2)$.
2. $p_1 \lessdot p_2 \Rightarrow \rho(p_2) = \rho(p_1) + 1$.

3. p is a minimal element of $\mathcal{P} \Rightarrow \rho(p) = 0$,

where $\lessdot$ denotes the *covering relation* on $\mathcal{P}$ that satisfies

$$p_1 \lessdot p_2 \Leftrightarrow p_1 \prec p_2 \wedge \forall p \in \mathcal{P}.\neg(p_1 \prec p \wedge p \prec p_2).$$

The following corollary follows from the properties of graded posets.

Corollary 1. *Any graded specification poset can be consistently-evaluated.*

Proof: This follows directly from Theorem 1 and the fact that any graded poset has properties 1) and 2) defined therein.

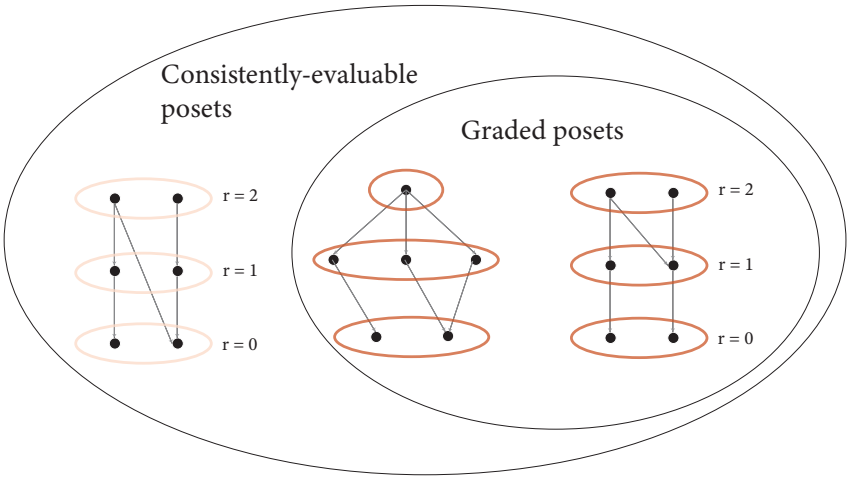

Figure 2.4: Graded specification posets are a subset of consistently-evaluable posets.

The relation between consistently-evaluable sets and partially-ordered sets is shown more clearly in Fig. 2.4. The main difference between consistently-evaluable posets and graded posets is the constraint that graded posets are defined such that the ranks assigned to two nodes which are comparable have to have a difference of one. It is easier to check whether a set is graded as opposed to consistently-evaluable because of the following lemma:

Lemma 1. *A poset is graded if and only if all of its maximal chains have the same length.*

Any consistently evaluable poset can be reduced to a "canonical" form that has the graded property with the same consistent evaluation.

Theorem 4. *Each consistently evaluable poset can be turned into a graded poset that is equivalent under consistent evaluation.*

Proof: This is achieved by removing all "edges" that span more than 2 levels of antichains in the unique partition of Theorem 1. One can without much difficulty verify that doing so will remove all maximal chains with length strictly less than the total number of these antichains, which by Lemma 1 implies that the resulting poset is graded. Since the other antichains are not affected by these operations, the resulting evaluation is not affected either.

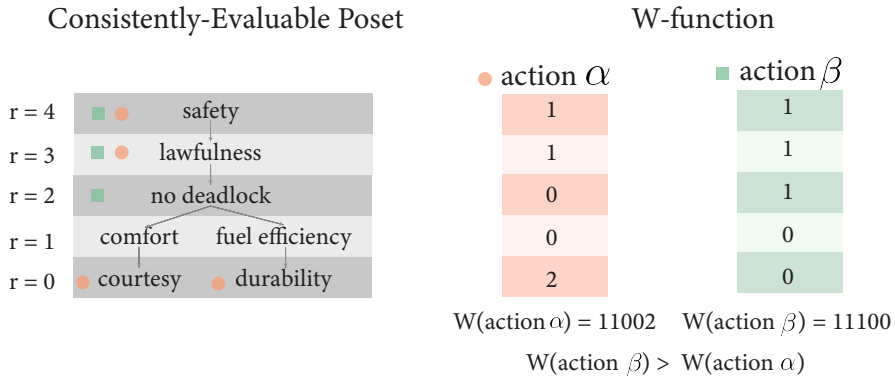

Figure 2.5: This shows how the consistent evaluator function W works on a graded specification poset. The function W computes a tuple for each subset, and compares the elements from most significant to least significant digits (left to right).

We now present an example of a graded specification poset and its respective consistent-evaluator function.

Example 4 (Consistent-Evaluator for a Graded Poset)**.** Let S, L, ND, Cf, FE, C, and D be specifications denoting safety, lawfulness, no deadlock, comfort, fuel efficiency, courtesy, and durability respectively. Let $P \coloneqq \{S, L, ND, Cf, FE, C, D\}$. The partial order on these specifications is shown in Fig. 2.5. Given the current world configuration, we assume the oracle can determine which subset of specifications will be satisfied by taking a given action. Let $P_\alpha \coloneqq \{S, L, C, D\}$ denote the subset of specifications satisfied by taking action α. Similarly, let $P_\beta \coloneqq \{S, L, ND\}$. To compare the actions α and β, given P_α, P_β, we use the evaluator W defined in the proof sketch of Theorem 1 to make the comparison. $W(P_\alpha) = [1, 1, 0, 0, 2]$ since this denotes the number of specifications satisfied in every ranking that can be satisfied by taking the action α, and $W(P_\beta) = [1, 1, 1, 0, 0]$ since there is one property in the top three ranked antichains that can be satisfied by taking action β. Therefore,

to evaluate their relative importance, the most significant figure corresponds to the left-most element in the tuple since that element has the highest rank. We begin our comparison there. Note that W_i represents the ith the element of the tuple. Since $W_0(P_\alpha) = 1$ and $W_0(P_\beta) = 1$, we have to keep comparing elements in the tuple to determine which one has higher ordering. We find $W_2(P_\alpha) < W_2(P_\beta)$. Therefore, P_β dominates P_α by the weak order imposed by the W evaluator and therefore, the action β should be chosen over α.

We introduced the notion of consistency of consistently-evaluable sets of specifications above. Now, we would like to introduce the idea of completeness of a consistently-evaluable set of specifications.

2.4 The Specification Structure and Completeness

Here, we define completeness by the number of specifications that are specified. A specification structure that has more specifications (i.e. encompasses a broader range of specifications) is therefore more complete.

In order to make an existing specification structure more complete, we must be able to refine the graph in a consistent manner. Refinement is equivalent to adding specifications (nodes) or comparisons (edges) to the specification structure in a way that preserves the gradedness property of a specification structure. We now define how to properly add a node or edge into the specification structure in a way that preserves the specification structure's mathematical properties.

The following is a direct corollary of Lemma 1.

Corollary 2 (Proper mode or edge refinement)**.** *If a node (or an edge) is added to the specification structure such that its relationship to the other nodes (the comparison it makes) is defined in a way that all maximal chains have the same length, then the resulting partially ordered set is also a specification structure.*

Examples for proper (and improper) ways of adding a new node are shown in Fig. 2.6. We have also included examples of how to make minimal modifications to accommodate for an improperly-added nodes and edges in Appendix A.

Thus far, we have focused on defining the mathematical structure on each individual profile. In the following section, we look at these profiles in the context of assume-guarantee contracts.

Figure 2.6: This shows proper and improper ways of adding a node into the poset. The addition is improper when the resulting poset is no longer graded (as seen on the right).

2.5 Behavioral Profiles in the Assume-Guarantee Context

Assume-guarantee contracts, introduced in Definition 1, are behavioral contracts that define agents' behavioral preferences. In the context of the assume-guarantee contracting framework, the guarantees are formulated such that the car will exhibit "correct" behavior by guaranteeing that the car will choose actions that are consistent with their respective profiles, where in this section we use the term profiles and specification structures (and their respective consistent-evaluating functions) interchangeably.

Example 5. Here, we give a very simple specification structure: lawfulness (L) $\prec$ no deadlock (ND) $\prec$ safety (S). We consider the consistent evaluator W (presented in the proof sketch of Theorem 1). W will have the ordering $W(\{L\}) < W(\{ND\}) < W(\{S\}) < W(\{S, L\}) < W(\{S, ND\}) < W(\{S, L, ND\})$. The ordering intuitively means that a car should always prioritize taking actions that satisfy all three types of specifications. However, if there is a situation where a car cannot ensure safety without breaking the law, then it should break the law to maintain safety since $W(\{S\}) > W(\{L\})$. Also, this hierarchy says if there is a situation where the car is in a deadlock, it can break the law since $W(\{S, ND\}) > W(\{S, L\})$ as long as the action is still safe.

As long as the car chooses behaviors that respect the weak order from the consistent evaluator on the specification structure, the system will satisfy the guarantees part of the assume-guarantee contract, and therefore perform actions that are "correct." We now introduce how assumptions can be defined with respect to the behavioral profiles.

While each autonomous vehicle should only guarantee that it will behave according

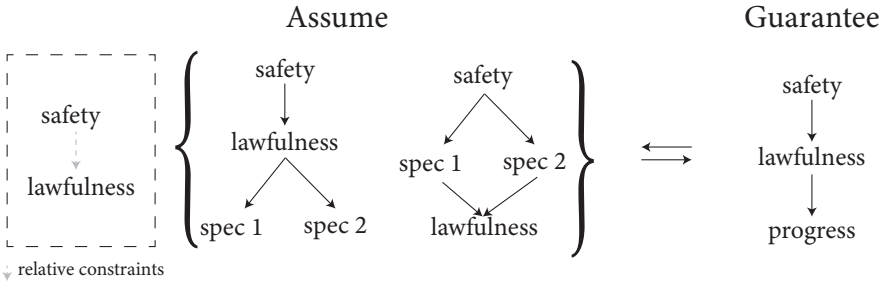

Figure 2.7: The assumptions are based on a set of specification structures that satisfy some constraints, which is shown on the left. The guarantees are based on a single specification structure that is shown on the right.

to a single profile, we want our assumptions on the environment (other agents) to accommodate for the diverse behaviors displayed by human drivers who may not follow the law all the time. This implies that other agents might choose to follow any one of a large number of possible profiles. We constrain the set of profiles of other agents to always prioritize safety first. Since other agents presumably follow the law most of the time, we also include a relative ordering constraint where safety is prioritized before the law. We have only defined a relative ordering between safety and law in the assumptions since we do not exactly know where other specifications will fit within that agent's specification structure. Therefore, our assumptions on the environment can be defined as follows:

Example 6 (Assumption set)**.** Let S denote the set of all specification structures. Let P be a set of specifications. Let $p \in P$. The *assumption set* in the assume-guarantee contract is defined as:

$$A_{spec} = \{S_i \in S | (safety \in S_i) \wedge (lawfulness \in S_i) \\ \wedge (\forall p \in S_i . p \preceq safety)\}.$$

It is the set of all specification structures that both safety and lawfulness are included in the specification structure and that safety has the highest rank out of all specifications included in the specification structure.

The following revised assume-guarantee definition of Definition 1 characterizes the set of specification structures agents in the environment can be assumed to have and the specifications that an individual self-driving car can guarantee.

Definition 5 (Assume-guarantee profiling revised)**.** An *assume-guarantee contract* C defined for an agent is a pair $(\mathcal{A}, \mathcal{G})$, where

1. Agents are assumed to select actions in accordance with one of the specification structures in $\mathcal{A}$. An example set of specification structures is given in Example 6.

2. $\mathcal{G}$ is the guarantee that the agent will select actions in accordance to a single, pre-defined specification structure.

This assume-guarantee profiling is shown in Fig. 2.7. Let $\mathcal{J}$ be the index set for a set of agents. Agents will invariably violate the assume-guarantee contracts. This assume-guarantee contract formalism, however, allows us to define a formal way to assign blame for the agent that causes something like a collision to occur. Before defining blame, we must introduce the definition of a compatible set of agents.

Let $C_j = (\mathcal{A}_j, \mathcal{G}_j)$, where j is the index of an agent and $\mathcal{A}_j$ are the assumptions that agent j is making about its environment while $\mathcal{G}_j$ is its guarantees. We say that the group of agents indexed by $\mathcal{J}$ are *compatible* if

$$\forall j \in \mathcal{J}.\forall i \in \mathcal{J} - \{j\}.\mathcal{G}_j \subseteq \mathcal{A}_i.$$

This says that each agent is compatible with every other agent as long as the agents only make assumptions on other agents that are guaranteed by all other agents. If one agent i has guarantees corresponding to a specification structure that is not included in another agent k's assumptions, then correct behavior (i.e. each agent is operating according to its respective protocol) cannot be guaranteed. Similarly, if one agent i has assumptions on another agent that are not guaranteed by an agent k, then correct behavior cannot be guaranteed. Assuming that all agents' assumptions and guarantees are compatible, we can formulate the notion of a *blame-worthy* action/strategy.

Definition 6 (Blameworthy action)**.** A *blameworthy action* is one in which an agent violates its guarantees, thereby causing another agent's assumptions not to be satisfied and thus resulting in an unwanted situation where blame must be assigned.

In order to show an example of an assume-guarantee contract that might be legally imposed for self-driving cars, we present a *set of axioms for the road*. The specification structures defined in the assumptions and guarantees of this contract are intentionally left unrefined, since it would ultimately be up to a car-manufacturer to determine the remaining ordering of specification properties.

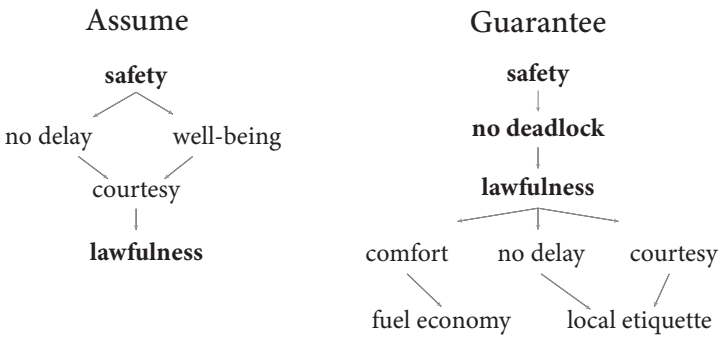

Figure 2.8: Two examples of refined assumption (left) and guarantee (right) specification structures. Specifications in the root structures are in bold text. The left could represent an ambulance and the right could represent a civilian vehicle.

A1 Other agents will not act such that collision is inevitable.

A2 Other agents will often act corresponding to traffic laws, but will may or may not follow them.

G1 An agent will take no action that makes collision inevitable.

G2 An agent will follow traffic laws, unless following them leads to inevitable collision.

G3 An agent may violate the law if by doing so, it can safely get out of a deadlock situation.

We can see from Fig. 2.8 how these axioms have a direct mapping to a specification structure. We argue that this sort of root structure might be imposed by a governing body to ensure the safe behaviors of self-driving cars.

2.6 Game Examples

In this section, we present some preliminary examples of how these types of high-level behavioral specifications that are defined via these specification structures might be applied in some traffic scenarios.

Under the simplified assumption that each agent has a single specification structure (i.e. agents are not human), each agent will have a well-defined ordering of which actions have higher value, and will therefore have a well-defined utility function over actions. Game theory provides a mathematical model of strategic interaction between rational decision-makers that have known utility functions [30]. We can

therefore use game-theoretic concepts to analyze which pair of actions will be jointly advantageous for the agents given their specification structures.

Multiple Nash Game

Consider the case where there are two agents, each of whose specification structures are specified in Fig. 2.9. In this game, Player Y encounters some debris, and must choose an action. Player Y can either choose to stay in its current location, or do a passing maneuver that requires it to break the law. Player X represents a car moving in the opposite direction of Player Y. In this case, Player X can either move at its current velocity or accelerate. The move and accelerate action make Player X move one and two steps forward, respectively. The W function is the same as the one provided in the proof sketch of Theorem 1.

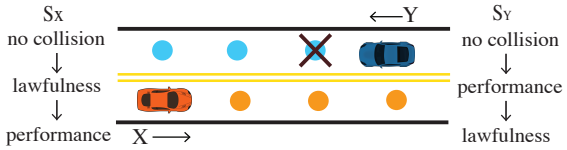

Figure 2.9: The game scenario when Player Y encounters debris on its side of the road. The specification structures of each of the agents are given by S_x and S_y, where S_x and S_y are different since they are presumed to come from different manufacturers.

W_x is evaluated on the specification structure S_x shown on the left side of Fig. 2.9 and W_y is evaluated on S_y. Assuming there is a competent oracle who gives the same predictions for both agents, the resulting payoff matrix according to the specification structures are given in Table 2.1 (note that an equivalent decimal conversion of the scores is given for ease of reading).

Table 2.1: Two-Player Game with Multiple Nash

playerX/playerY	Stay	Pass
Move	$W_x(1,1,0) \sim 6$ $W_y(1,0,1) \sim 3$	$W_x(1,1,0) \sim 6$ $W_y(1,1,0) \sim 6$
Accelerate	$W_x(1,1,1) \sim 7$ $W_y(1,0,1) \sim 3$	$W_x(0,0,0) \sim 0$ $W_y(0,0,0) \sim 0$

A Nash equilibrium is a set of strategies, one for each of the n agents in the game, for which each agent's choice is the best response to each of the $n-1$ other agents [42]. From the table, we can see that there are two Nash equilibria in this game scenario. In

this case, the two equilibria are Pareto efficient, meaning there are no other outcomes where one player could be made better off without making the other player worse off. Since there are two equilibria, there is ambiguity in determining which action each player should take in this scenario despite the fact that the specification structures are known to both players.

There is a whole literature on equilibrium selection [30]. The easiest way to resolve this particular stand-off, however, would be to either 1) communicate which action the driver will take or 2) define a convention that all self-driving cars should have when such a situation occurs. In this particular scenario, however, Player X can certainly avoid accident by choosing to maintain speed while Player Y can also avoid accident by staying. Any "greedy" action of either Player X or Y would pose the risk of crashing depending on the action of the other player. This suggests a risk-averse resolution in accident-sensitive scenarios like this one. We will focus on defining a more systematic way of resolving multiple Nash equilibria in future work.

Faulty Perception Game

In this work, we have abstracted the perception system of the self-driving car to the all-knowing oracle. We first consider the case where the oracles on each of the cars are in agreement, and then consider the potential danger when the oracles of the cars differ. In this scenario, we assume that there are two cars that are entering an intersection with some positive velocity, as shown in Fig. 2.10.

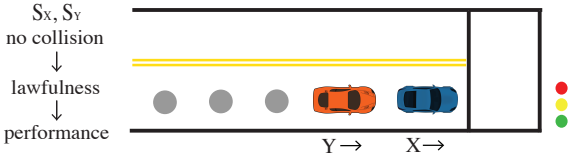

Figure 2.10: The game scenario where two cars are approaching an intersection, but have different beliefs about the state of the traffic light.

In the case where both vehicles' oracles agree on the same information, i.e. that the yellow light will remain on for long enough for both vehicles to move past the intersection, the best action for both Player X and Player Y is to move forward.

Now, consider the case where the oracles are giving incompatible beliefs about the environment, namely, the state of the traffic signal. Let X have the erroneous belief that the traffic light will turn red very soon, and it assumes that Y's oracle believes

Table 2.2: Two-Player Game with Perfect Perception

playerX/playerY	Slow	Move
Slow	$W_x(1,1,0) \sim 6$ $W_y(1,1,0) \sim 6$	$W_x(0,1,0) \sim 2$ $W_y(0,0,0) \sim 0$
Move	$W_x(1,0,1) \sim 3$ $W_y(1,1,0) \sim 6$	$W_x(1,0,1) \sim 3$ $W_y(1,0,1) \sim 3$

the same thing. X's oracle gives rise to Table 2.2, according to which the conclusion that Player X will make is that both of the cars should choose to slow down.

Assume that Y has a perfect oracle that predicts the traffic light will stay yellow for long enough such that Y would also be able to make it through the intersection. If Y assumes that X has the same information (see Table 2.3), then the best choice for both is to move forward into the intersection.

Table 2.3: Two-Player Game with Faulty Perception

playerX/playerY	Slow	Move
Slow	$W_x(1,1,0) \sim 6$ $W_y(1,1,0) \sim 6$	$W_x(0,1,0) \sim 2$ $W_y(0,0,0) \sim 0$
Move	$W_x(1,1,1) \sim 7$ $W_y(1,1,0) \sim 6$	$W_x(1,1,1) \sim 7$ $W_y(1,1,1) \sim 7$

The incompatible perception information will thus cause Player X to stop and Player Y to move forward, ultimately leading to collision.

This particular collision is caused by errors in the perception system. Future work will need to focus on developing a better perception system or on creating a system that will yield correct behaviors even in the presence of perception uncertainty.

2.7 Conclusion

In summary, we have proposed the idea of assume-guarantee contracts for multi-agent systems that we broadly define as a set of behavioral rules that each agent is constrained to act according to. In this work, we present a first iteration of a behavioral contract. In particular, we have defined a behavioral profile, which is a mathematically-ordered structure on a set of specifications, that agents use to transparently define which action to prioritize (and ultimately select) at every time-step. Through some simple game-theoretic scenarios, we show the inherent limitations of all agents selecting actions according to these behavioral profiles. In particular, we see how agents are not able to coordinate safely when multiple Nash

equilibria arise. In Chapter 3, we will extend upon this work to enable safe and high-performing coordination of multi-agent systems.

Chapter 3

DECISION-MAKING: BEHAVIORAL PROTOCOLS

3.1 Introduction

In the previous chapter, we discussed assume-guarantee contracts, where each agent is making decisions in accordance with a behavioral profile [72]. The behavioral profiles enabled a decision-making strategy that was interpretable, flexible, and prioritized safety. The limitations of using these profiles, however, became evident when modeling agents making decisions simultaneously. The existence of multiple-nash equilibria and the lack of coordination among agents made it ambiguous as to which action agents should ultimately select that were mutually beneficial and safe.

In this chapter, we introduce a new framework to resolve the ambiguity among agents with conflicting intentions in a way that guarantees both safety and progress of all agents. A preliminary version of the work presented in this chapter appears in [15], and was done jointly with Tung Phan-Minh.

Here, we present a shift from thinking in terms of behavioral profiles to behavioral protocols. The behavioral protocol can be thought of as a set of rules agents must follow to select their action (like the behavioral profile), but that also dictates when agents are allowed to take their intended action or have to defer to other agents. In this way, the behavioral profiles are a single element of the behavioral protocol—the profile serves to help an agent select which action it intends to take. The behavioral protocol builds on the behavioral profile by adding in constraints that determine whether or not an agent is allowed to take its intended action. By constraining agent behavior in this way, we can guarantee safety and liveness properties. Note that the safety property is that agents do not collide and the liveness property is equivalent to all agents making progress towards their respective destinations.

In this chapter, we introduce the framework necessary for defining the agent protocol. In particular, we: 1) The introduction of a new game paradigm, which we term the quasi-simultaneous discrete-time multi-agent game, 2) the definition of an agent protocol that defines local rules agents must use to select their actions, 3) safety and liveness proofs when all agents operate according to these local rules and 4) simulations as proof of concept of the safety and liveness guarantees.

3.2 Quasi-Simultaneous Discrete-Time Game

We propose a quasi-simultaneous discrete-time game paradigm, which is motivated by the shortcomings of more traditional game paradigms. In simultaneous games, all agents in the game are making decisions simultaneously. Since agents are making decisions in the absence of knowing other agent decisions, it does not capture the sequential and reactive nature of real-life decision making. Turn-based games offer potential for capturing sequential decision-making, but the turns are often assigned arbitrarily. The quasi-simultaneous discrete-time game is a turn-based game, but instead of being randomly assigned, the turn order is determined by the agent states defined with respect to the road network.

A *state* associated with a set of variables is an assignment of values to those variables. A game evolves by a sequence of state changes. A quasi-simultaneous game has the following two properties regarding state changes: 1) Each agent will get to take a turn in each time-step of the game and 2) Each agent must make their turn in an order that emerges from a locally-defined precedence assignment algorithm. Thus, the state-change is simultaneous yet locally sequential because each agent must make a state-change in a given time step, but it must wait for its turn according to turn order (defined based on the locally-defined precedence assignment algorithm) during this time-step. We define a quasi-simultaneous game where all agents act in a local, decentralized manner as follows:

$$\mathfrak{G} = \langle \mathfrak{A}, \mathcal{Y}, Act_{[\cdot]}, \rho_{[\cdot]}, \tau_{[\cdot]}, P \rangle \tag{3.1}$$

where

- $\mathfrak{A}$ is the set of all agents in the game $\mathfrak{G}$.
- $\mathcal{Y}$ is the set of all variables in the game $\mathfrak{G}$.
- For each agent $\text{Ag} \in \mathfrak{A}$, let:
  - S_{Ag} be the set that contains all possible states of V_{Ag}, where V_{Ag} are the variables associated with each agent Ag and $V_{\text{Ag}} \subseteq \mathcal{Y}$.
  - Act_{Ag} be the set of all possible actions Ag can take.
  - $\tau_{\text{Ag}} : S_{\text{Ag}} \times Act_{\text{Ag}} \rightarrow S_{\text{Ag}}$ be the transition function that defines the state an agent will transition to when taking an action $a \in Act_{\text{Ag}}$ from a given state.

- $\rho_{\text{Ag}} : S_{\text{Ag}} \rightarrow 2^{Act_{\text{Ag}}}$ be a state-precondition function that defines a set of actions an agent can take at a given state.

- $P : \mathcal{Y} \rightarrow \text{PolyForest}(\mathfrak{A})$, is the precedence assignment function where PolyForest is an operator that maps a set to a polyforest graph object. The polyforest, with its nodes and directed edges, defines the global turn order (of precedence) of the set of all agents $\mathfrak{A} \in \mathfrak{G}$ based on the agent states.

Note, the transition function τ_{Ag} and the state-precondition function ρ_{Ag} must be compatible for any agent Ag. In particular, $\forall \text{Ag} \in \mathfrak{A}$ and $\forall s \in S_{\text{Ag}}$, $\text{Domain}(\tau_{\text{Ag}}(s, \cdot)) = \rho_{\text{Ag}}(s)$.

3.3 Specific Agent Class

In order to make global guarantees on safety and progress, we first only consider a single specific class of agents whose attributes, dynamics, motion-planner, and perception capabilities are described in more detail in the following section. Although assuming a single class of agents seems very restrictive, the work can be easily extended to accommodate additional variants of the agent class. These extensions, however, are beyond the scope of this work.

Agent Attributes

Each *agent* Ag is characterized by a set of variables $\mathcal{V}_{\text{Ag}} \subseteq \mathcal{Y}$. In this work, we only consider *car* agents such that if $\text{Ag} \in \mathfrak{A}$, then $\mathcal{V}_{\text{Ag}}$ includes x_{Ag}, y_{Ag}, θ_{Ag}, v_{Ag}, namely its absolute coordinates, heading and velocity. $\mathcal{V}_{\text{Ag}}$ also has parameters $a_{\text{minAg}} \in \mathbb{Z}$, $a_{\text{maxAg}} \in \mathbb{Z}$, $v_{\text{minAg}} \in \mathbb{Z}$ and $v_{\text{maxAg}} \in \mathbb{Z}$ which define the minimum and maximum accelerations and velocities, respectively. The agent also has the variables: $\{\texttt{Id}_{\text{Ag}}, \texttt{Tc}_{\text{Ag}}, \texttt{Goal}_{\text{Ag}}\} \subseteq \mathcal{V}_{\text{Ag}}$ where $\texttt{Id}_{\text{Ag}}$, $\texttt{Tc}_{\text{Ag}}$, and $\texttt{Goal}_{\text{Ag}}$ are the agent's ID number, token count, and goal, respectively, where the token count and ID are used in the conflict-cluster resolution defined in Section 3.5. Agents are assumed to have the capability of querying the token counts of neighboring agents.

The agent control actions are defined by two parameters: 1) an acceleration value acc_{Ag} between a_{minAg} and a_{maxAg} and 2) a steer maneuver $\gamma_{\text{Ag}} \in \{\texttt{left-turn}, \texttt{right-turn}, \texttt{left-lane change}, \texttt{right-lane change}, \texttt{straight}\}$.

The discrete agent dynamics works as follows. At a given state $s \in S_{\text{Ag}}$ at time t, for a given control action $(\text{acc}_{Ag}, \gamma_{Ag})$, the agent first applies the acceleration to update its velocity $s.v_{Ag,t+1} = s.v_{Ag,t} + \text{acc}_{\text{Ag}}$. Once the velocity is applied, the

steer maneuver (if at the proper velocity) is taken and the agent occupies a set of grid-points, specified in Fig. 3.1, while taking its maneuver.

Before and after a state transition, the agent is assumed to occupy only a single grid point. During an agent state transition, an agent might occupy one or more grid points. Fig. 3.1 shows the grid point occupancy for different agent maneuvers. The concept of grid point occupancy is defined as follows:

Definition 7 (Grid Point Occupancy)**.** The notion of *grid point occupancy* is captured by the definitions of the following maps for each $\mathrm{Ag} \in \mathfrak{A}$. To define the grid point an agent is occupying at a given time, we use the map: $\mathcal{G}_{\mathrm{Ag},t} : S_{\mathrm{Ag}} \rightarrow 2^G$, mapping each agent to the single grid point the agent occupies. Note G is a grid point cell of the road network. By a slight abuse of notation, we let $\mathcal{G}_{Ag,t} : S_{\mathrm{Ag}} \times Act_{\mathrm{Ag}} \rightarrow 2^G$ be a function that maps each $s \in S_{\mathrm{Ag}}$ and $a \in \rho_{\mathrm{Ag}}(s)$ to denote the set of all nodes that are occupied by the agent Ag when it takes an allowable action a from state s at the time-step t.

The occupancy grids associated with each of the maneuvers allowed for the agents, and the velocity that the agent has to be at to take the maneuver, are shown in Fig. 3.1.

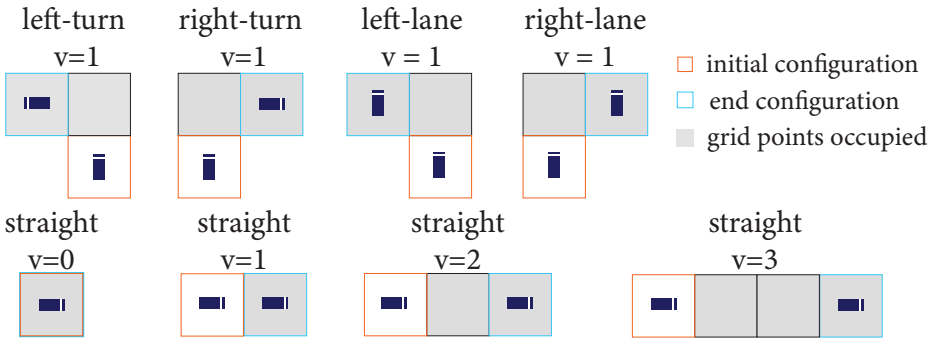

Figure 3.1: Different grid point occupancy associated with different discrete agent maneuvers. Note that the grid point occupancy represents a conservative space which the agent may occupy when taking the associated maneuver.

Note, the safety and liveness guarantees will hold for any choice of agent dynamic parameters (i.e. $a_{\min}, a_{\max}, v_{\min}, v_{\max}$), but only under the condition that all agents have the same set of dynamic parameters. The maneuvers must be the ones specified above.

We assume that any graph planning algorithm can be used to specify an agent's motion plan. The motion plan must be divided into a set of critical points along the graph that the agent must reach in order to get to its destination, but should not specify the exact route agents must take to get to these critical points. It should be noted that the liveness guarantees rely on the assumption that rerouting of the agent's motion plan is not supported.

Agent Backup Plan Action

A *backup plan* is a reserved set of actions an agent is *entitled* to execute at any time while being immune to being at fault for a collision if one occurs. In other words, an agent will always be able to safely take its backup plan action. We show that if each agent can maintain the ability to safely execute its own backup plan (i.e. keep a far enough distance behind its lead agent), the safety of the collective system is guaranteed.

The default backup plan adopted here is that of applying maximal deceleration until a complete stop is achieved, which is defined as:

Definition 8 (Backup Plan Action)**.** *The backup plan action a_{bp} is a control action where $a = a_{min}$ and when applying a_{min} causes the agent's velocity to go below 0, $a = max(a_{min}, -s.v_{Ag})$, and $\gamma_{Ag} =$ `straight`.*

Note, it may take multiple time-steps for an agent to come to a complete stop because of the inertial dynamics of the agent.

Limits on Agent Perception

In real-life, agents make decisions based on local information. We model this locality by defining a region of grid points around which agents have access to the full state and intention information of the other agents. We assume agents have different perception capabilities in different contexts of the road network when making decisions on 1) road segments and 2) at intersections.

Road Segments

For road segments, the region around which agents make decisions cannot be arbitrarily defined. In fact, an agent's bubble must depend on its state, and the agent attributes and dynamics of all agents in the game. In particular, the bubble can be defined as follows:

Definition 9 (Bubble)**.** Let Ag be an agent with state $s_0 \in S_{\text{Ag}}$. Let agent Ag′ be another agent. Then the *bubble* of Ag with respect to agents of the same type as Ag′ is given by $\mathcal{B}_{\text{Ag/Ag}'}(s_0)$. The bubble is the minimal region of space (set of grid points) agents need to have full information over to guarantee they can make a decision that will preserve safety under the defined protocol. Since all agents considered in this work have the same attributes, for ease of notation, we refer to the bubble of Ag as $\mathcal{B}_{\text{Ag}}$.

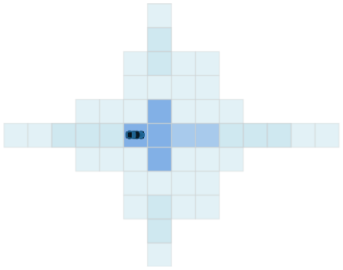

Figure 3.2: Bubble if all Ag $\in \mathfrak{A}$ have the agent dynamics specified in Section 3.3. The transparency is a property of how the bubble is constructed. Details can be found in Appendix B.

For our protocol, the bubble contains any grid points in which another agent Ag′ occupying those grid points can interfere with at least one of Ag's next possible actions and the backup plan it would use if it were to take any one of those next actions. With a slight abuse of notation, we say $\text{Ag}' \in \mathcal{B}_{\text{Ag}}(s)$ if $(s.x_{\text{Ag}'}, s.y_{\text{Ag}'})$ is on a grid point in the set $\mathcal{B}_{\text{Ag}}(s)$.

Intersections

The locality of information that agents are restricted to is relaxed at intersections because agents can presumably see across the intersection when making decisions about crossing the intersection. More precisely, any Ag must be able to know about any Ag′ $\in \mathfrak{A}$ that is in the lanes of oncoming traffic (when performing an unprotected left turn). The computation of the exact region of perception necessary depends on the agent dynamics. Locality for the local-precedence assignment algorithm is also extended to this larger region at intersections as well.

3.4 Road Network Environment

Here we introduce the structure of the road network environment that agents are assumed to be operating on. The road network is a grid world with additional

structure (e.g. lanes, bundles, road segments, intersections, etc.). The road network is formalized as follows:

Definition 10 (Road Network)**.** A *road network* $\mathfrak{R}$ is a graph $\mathfrak{R} = (G, E)$ where G is the set of grid points and E is the set of edges that represent immediate adjacency in the Cartesian space among grid points. Note that each grid point $g \in G$ has a set of associated properties $\mathcal{P}$, where $\mathcal{P} = \{p, d, \texttt{lo}\}$ which denote the Cartesian coordinate, drivability of the grid point, and the set of legal orientations allowed on the grid point, respectively. Note, $p \in \mathbb{Z}^2$, $d \in \{0, 1\}$ and $\texttt{lo}$ is a set of headings ϕ_l where each $\phi_l \in \{\texttt{north}, \texttt{east}, \texttt{south}, \texttt{west}\}$.

Grid points where specific properties hold are given special labels, which can be seen in Fig. 3.3. These labels and the associated properties are defined as follows:

- $\mathcal{S}_{\text{sources}}$, ($\mathcal{S}_{\text{sinks}}$): A set of grid points designated for Ag to enter (or leave) the road network $\mathfrak{R}$ from.
- $\mathcal{S}_{\text{intersection}}$: A set of grid points that contains all grid points with more than one legal orientation.
- $\mathcal{S}_{\text{traffic light}}$: A set of grid points that represents the traffic light states in the vertical or horizontal direction via its color (for every intersection).

The road network is hierarchically decomposed into lanes and bundles, which are defined as follows:

- Lanes: Let lane $La(g)$ define a set of grid points that contains g and all grid points that form a line going through g.
- Bundles: Let $Bu(g)$ be a set of grid points that make up a set of lanes that are adjacent or equal to the lane containing g and have the same legal orientation.

Each bundle can be decomposed into a set of road segments, which we refer to as RS, where the intersections are used to partition each bundle into a set of road segments. Example of a bundle partitioned into road segments is shown in Fig. 3.3.

We introduce the following graph definition since it will be used in the liveness proof.

Definition 11 (Road Network Dependency Graph)**.** *The road network dependency graph is a graph* $G_{dep} = (RS, E)$ *where nodes are road segments and a directed edge* (rs_1, rs_2) *denotes that agents on* rs_1 *depend on the clearance of agents in* rs_2 *to make forward progress.*

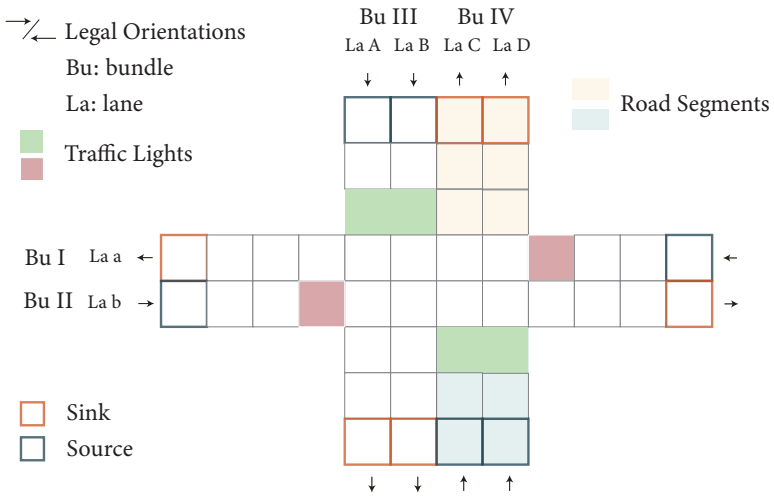

Figure 3.3: Road network decomposition where each box represents a grid point.

With slight abuse of notation, we let $La(\text{Ag})$ refer to the lane ID associated with the grid point $(s.x_{\text{Ag}}, s.y_{\text{Ag}})$, and $Bu(\text{Ag})$ mean the bundle ID associated with the lane $La(\text{Ag})$.

Scope of Road Network Environments

The safety and progress guarantees do not generalize to all possible road networks. Instead, the road network environments are any straight road segments with traffic intersections governed by traffic lights (with unprotected left turns) and less than 3 lanes per bundle. Extending the work to accommodate more road network environments, however, is not difficult but it is beyond the scope of the work presented in this chapter.

3.5 The Agent Protocol

Now that we have defined the game, the specific class of agents (and their associated attributes) and the road network environment, we introduce the agent protocol. The protocol serves to 1) define the method agents use to choose an intended action and 2) define rules that an agent uses to determine whether it has priority to take its

intended action, and if not, which alternative, less-optimal actions it is allowed to take. The protocol, as defined, will ensure safety and progress for all agents in the multi-agent game.

Motivation of Protocol Design

In this section, we introduce the agent protocol. The agent protocol is defined to establish local rules agents must follow while making decisions on the road network. The protocol must be defined in a way such that it 1) scales well, 2) is interpretable so there is a consistent and transparent way agents make their decisions, 3) ensures the safety of all agents, and 4) ensures progress for all agents. In this section, we introduce the components that form the agent protocol that make it such that all these properties are satisfied.

High-Level Overview

The introduction of a single backup plan action that all agents rely on is fundamental for making the protocol work. Dependence on this backup plan action is what ultimately allows for the decoupling of agent dependencies when reasoning about one another, while still allowing us to guarantee global properties like safety and liveness.

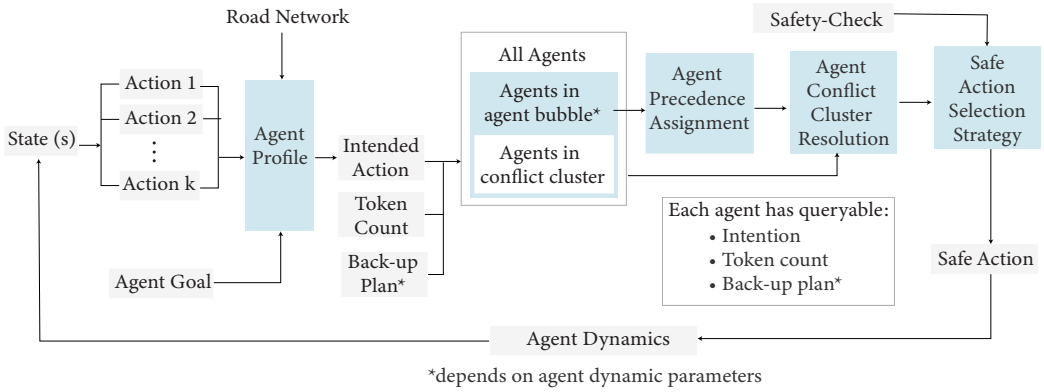

Figure 3.4: Agent protocol architecture.

The following is an overview of the agent protocol. For each time step of the game, each agent first assigns local precedence, thereby establishing a consistent turn order (in a local manner). Then, each agent evaluates a set of actions and chooses the best action according to its behavioral profile as its intended action. Since the turn-order does not fully resolve ambiguity on which agents should be allowed to take their intended action at a given time, the action selection strategy,

based on 1) what actions agents ahead of it (in turn) have taken and 2) the results of the conflict-cluster resolution, defines how agents should ultimately decide which action to select (whether its the agent's intended action or an alternative sub-optimal action). The next sections of this paper will formalize these ideas and go into greater depth of how these ideas work for a specific class of agents with the particular set of dynamics described in Section 3.3.

Agent Precedence Assignment

The definition of the quasi-simultaneous game requires agents to locally assign precedence, i.e. have a set of rules to define how to establish which agents have higher, lower, equal, or incomparable precedence to it. Our precedence assignment algorithm is motivated by capturing how precedence among agents is generally established in real-life scenarios on a road network. In particular, since agents are designed to move in the forward direction, we aim to capture the natural inclination of agents to react to the actions of agents visibly ahead of it.

Before presenting the precedence assignment rules, we must introduce a few definitions. Let us define: $\text{proj}^{B}_{\text{long}} : \mathfrak{A} \to \mathbb{Z}$, which is restricted to only be defined on the bundle B. In other words, $\text{proj}^{B}_{\text{long}}(Ag)$ is the mapping from an agent (and its state) to its scalar projection onto the longitudinal axis of the bundle B the agent Ag is in. If $\text{proj}^{B}_{\text{long}}(\text{Ag}') < \text{proj}^{B}_{\text{long}}(\text{Ag})$, then the agent Ag$'$ is behind Ag in B.

The following rules can be used to define the precedence relation among agents Ag and Ag'. In other words, for a given Ag, they tell the agent whether to assign higher, equivalent, or lower precedence to another agent Ag'—or whether they are incomparable.

Local Precedence Assignment Rules

1. If $\text{proj}^{B}_{\text{long}}(\text{Ag}') < \text{proj}^{B}_{\text{long}}(\text{Ag})$ and $Bu(\text{Ag}') = Bu(\text{Ag})$, then $\text{Ag}' \prec \text{Ag}$, i.e. if agents are in the same bundle and Ag is longitudinally ahead of Ag$'$, Ag has higher precedence than Ag$'$.

2. If $\text{proj}^{B}_{\text{long}}(\text{Ag}') > \text{proj}^{B}_{\text{long}}(\text{Ag})$ and $Bu(\text{Ag}') = Bu(\text{Ag})$, then $\text{Ag} \prec \text{Ag}'$, i.e. if agents are in the same bundle and Ag' is longitudinally ahead of Ag, Ag has lower precedence than Ag$'$.

3. If $\text{proj}^{B}_{\text{long}}(\text{Ag}') = \text{proj}^{B}_{\text{long}}(\text{Ag})$ and $Bu(\text{Ag}') = Bu(\text{Ag})$, then $\text{Ag} \sim Ag'$ and we say that Ag and Ag$'$ are equivalent in precedence.

4. If Ag′ and Ag are not in the same bundle, then the two agents are incomparable.

Each agent Ag $\in \mathfrak{A}$ only assigns precedence according to the above rules locally to agents within its perception region (e.g. bubble on road segments and a slightly larger region at intersections, defined in Section 3.3) when making a decision of which action to take.

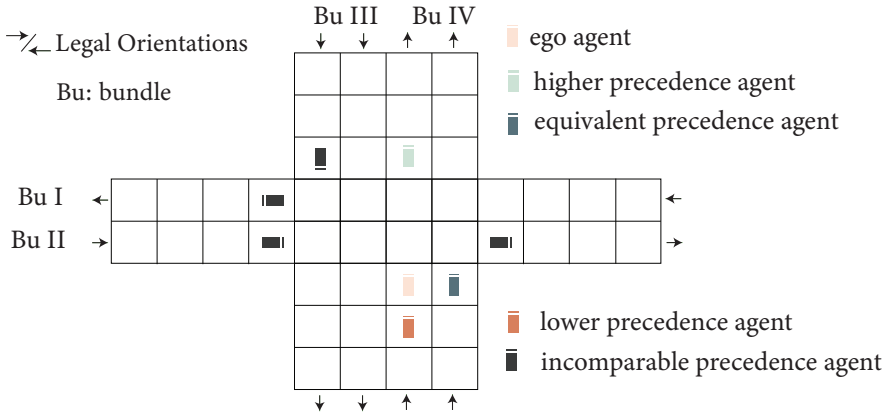

Figure 3.5: Rules for precedence assignment.

Thus, we must show if all agents locally assign precedence according to these rules, a globally-consistent turn precedence among all agents is established. In particular, we need to prove the following lemma.

Lemma 2. *If all agents assign precedence according to the local precedence assignment rules to agents in their respective bubbles, then the precedence relations will induce a polyforest on* $\mathfrak{A}/\sim$ *(the quotient set of S by* $\sim$*).*

Proof: Suppose there is a cycle C in $\mathfrak{A}/\sim$. For each of the equivalent classes in C (C must have at least 2 to be a cycle), choose a representative from $\mathfrak{A}$ to form a set R_C. Let Ag $\in R_C$ be one of these representatives. Applying the third local precedence assignment rule inductively, we can see that all agents in R_C must be from Ag's bundle. By the first local precedence assignment rule, any C edge must be from an agent with lower projected value to one with a higher projected value in this bundle. Since these values are totally ordered (being integers), they must be the same. This implies that C only has one equivalence class, a contradiction.

The acyclicity of the polyforest structure implies the consistency of local agent precedence assignments. Note, the local precedence assignment algorithm establishes the order in which agents are taking turns. Even when this order is established, it is ambiguous what actions the agents should select when 1) agents of equal precedence have conflicting intentions, since they select their actions at the same time or 2) an agent's intended action is a lane-change action and requires agents of lower or equivalent precedence to change their behavior so the lane-change action is safe. The additional set of rules introduced to resolve this ambiguity is what we refer to as conflict cluster resolution, which is defined later on in this section.

Behavioral Profile

The way in which agents select actions is the fundamental role of the behavioral protocol. The behavioral profile serves the purpose of defining which action an agent intends to take at a given time-step t.

We define a specific assume-guarantee profile, with the mathematical properties described in Chapter 2, for the agent. In particular, we define a set of ten different specifications (rules) and the hierarchy of importance (ordering) on these specifications.

Each specification is associated with an oracle that interprets whether or not taking a given action will satisfy the specification. More formally, let $r \in R$ denote a specification for an agent and $\text{Ag} \in \mathfrak{A}$. The oracle, for a specification r is defined as follows $O_{\text{Ag}r} : S_{\text{Ag}} \times Act_{\text{Ag}} \times \mathcal{U} \rightarrow \mathbb{B}$ where $\mathcal{U}$ is the set of all possible states of the game, $\mathbb{B} = \{\texttt{T}, \texttt{F}\}$, and the subscript t denotes the time-step the oracle is evaluated. In other words, given a specification and an action, the oracle will evaluate to T (or F) depending on whether an agent taking that action in its current state will satisfy (or not satisfy) the specification.

There are ten specifications in the assume-guarantee profile we consider for the agent protocol. The following define how the oracles evaluate satisfaction of each of the ten specifications:

1. $O_{\textbf{Ag,dynamic safety}}(s, a, u)$**:** returns T when the action a from state s will not cause Ag to either collide with another agent or end up in a state where the agent's safety backup plan a_{bp} is no longer safe with respect to other agents (assuming other agents are not simultaneously taking an action).

2. $O_{\textbf{Ag,unprotected left-turn safety}}(s, a, u)$ returns T when the action a from the state s

will result in the complete execution of a safe, unprotected left-turn (invariant to agent precedence). Note, an unprotected left turn spans over multiple time-steps. The oracle will return T if Ag has been waiting to take a left-turn (while the traffic light is green), the traffic light turns red, and no agents are present in oncoming lanes.

3. $O_{\textbf{Ag,static safety}}(s, a, u)$ returns T when the action a from state s will not cause the agent to collide with a static obstacle or end up in a state where the agent's safety backup plan a_{bp} with respect to the static obstacle is no longer safe.

4. $O_{\textbf{Ag,traffic light}}(s, a, u)$ returns T if the action a from the state s satisfies the traffic light laws (not crossing into intersection when red. It also requires that Ag be able to take a_{bp} from $s' = \tau_{\text{Ag}}(s, a)$ and not violate the traffic-light law.

5. $O_{\textbf{Ag,legal orientation}}(s, a, u)$ returns T if the action a from the state s follows the legal road orientation.

6. $O_{\textbf{Ag,traffic intersection clearance}}(s, a, u)$ returns T if the action causes the agent to enter the intersection and has enough clearance to safely exit the intersection. It also must be such that the action causes the agent to end up in a state where if it performs its backup plan action, it will still be able to leave the intersection.

7. $O_{\textbf{Ag,traffic intersection lane change}}(s, a, u)$ returns T if the action is not one where $\gamma_{Ag} = \{\texttt{left-lane change}, \texttt{right-lane change}\}$ and the agent either begins in an intersection or ends up in the intersection after taking the action.

8. $O_{\textbf{Ag,destination reachability}}(s, a, u)$ returns T if the action a from the state s will allow Ag$'s$ planned path to its goal to remain reachable.

9. $O_{\textbf{maintains progress}}(s, a, u)$ returns T if the action a from the state s stays the same distance to its goal $\texttt{goal}_{\text{Ag}}$.

10. $O_{\textbf{Ag,forward progress}}(s, a, u)$ returns T if the action a from the state s will improve the agent's progress towards it's goal $\texttt{goal}_{\text{Ag}}$.

The partial ordering of the specifications that each agent must follow is shown in Fig. 3.6. As a reminder, each action will satisfy some subset of specifications in this profile. The consistent-evaluating function, defined on this behavioral profile, will evaluate actions based on which specifications they satisfy, giving priority to actions with the highest number of highest-valued specifications as described in more detail

in Chapter 1. The action with the highest value is then selected as the action the agent intends to takes.

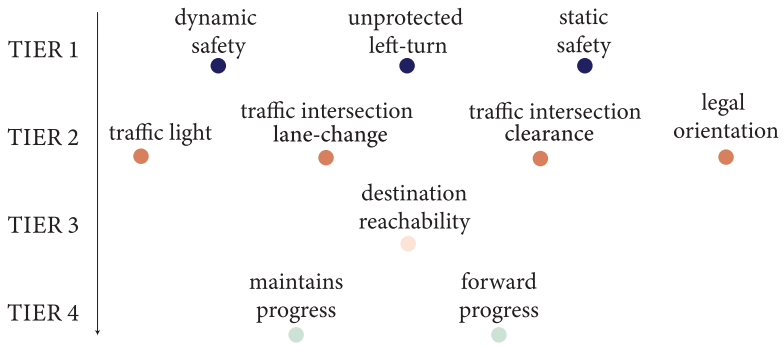

Figure 3.6: Assume-guarantee profile that shows ordering of specifications, where specifications on the same tier are incomparable to one another. Here, the tiers are equivalent to the ranked maximal antichains in the profiles defined in Chapter 2. Tier 1 is the highest priority tier. W.l.o.g. more oracles can be added with priority less than the oracles in the fourth tier.

For this work, the behavioral profile is used to define the agent's intended action a_i. Note, $O_{\text{Ag,dynamic safety}}(s, a, u)$ is not included in the selection of the intended action a_i—otherwise an agent might never propose a lane-change action (since it would require other agents to yield in order for the lane-change action to be safe). The profile is also used to define the agent's best straight action a_{st} which is defined later in the subsection on Action Selection Strategy. The best straight action can intuitively be thought of as the highest-ranked action that is a straight maneuver.

Conflict-Cluster Resolution

At every time-step t, each agent will know when to take its turn based on its local precedence assignment algorithm. Before taking its turn, the agent will have selected an intended action a_i using the behavioral profile. When it is the agent's turn to select an action, it must choose whether or not to take its intended action a_i. When the intended actions of multiple agents conflict, the conflict-cluster resolution is a token-based querying method used to help agents determine which agent has priority in taking its action.

Under the assumption that agents have access to the intentions of other agents within a local region as defined in Section 3.3, each agent can use the following criteria to define when it conflicts with another agent.

Definition 12 (Agent-Action Conflict)**.** Let us consider that an agent Ag is currently at state $s \in S_{\text{Ag}}$ and wants to take action $a \in \rho_{Ag}$ and an agent Ag′ at state $s' \in S_{\text{Ag}'}$ that wants to take action $a' \in \rho_{Ag'}$. We say that an *agent-action conflict* between Ag and Ag′ for a and a' occurs and write $(\text{Ag}, s, a) \dagger (\text{Ag}', s', a')$ if each of the agents taking their respective intended actions which will cause them to overlap in occupancy grid points or end up in a configuration where the agent behind does not have a valid safe backup plan action (i.e. if the lead agent executes its safety backup plan action, the following agent is far enough behind that it can safely execute its own safety backup plan action).

In the case that an agent's action is in conflict with another agents' action, the agent must send a conflict request that ultimately serves as a bid the agent is making to take its intended action. It cannot, however, send requests to just any agent (e.g. agents in front of it). The following criteria are used to determine the properties that must hold in order for an agent Ag to send a conflict request to agent Ag′:

Criteria that Must Hold for Agent Ag to Send Conflict Request to Agent Ag'

- Ag's intended action a_i is a lane-change action (i.e. $\gamma_{Ag} \in \{\texttt{left-lane change}, \texttt{right-lane change}\}$).
- $\text{Ag}' \in \mathcal{B}_{Ag}(s)$, i.e. Ag′ is in agent Ag's bubble.
- $\text{Ag}' \precsim Ag$, i.e. Ag has equivalent or higher precedence than Ag′.
- $s.\theta_{Ag} = s.\theta_{Ag'}$, i.e. the agents have the same heading.
- $(Ag, a_i) \dagger (Ag', a_i')$: agents' intended actions are in conflict with one another.
- $\mathcal{F}_{\text{Ag}}(u, a_i) = \texttt{F}$, where $\mathcal{F}_{\text{Ag}}(u, a_i)$ is the maximum-yielding-not-enough flag and is defined below.

Definition 13 (maximum-yielding-not-enough flag)**.** The *maximum-yielding-not-enough flag* $\mathcal{F}_{\text{Ag}} : \mathcal{U} \times Act_{\text{Ag}} \to \mathbb{B}$ that is set to T when Ag is in a configuration where if Ag did a lane-change, even if Ag' applied its maximum-yielding action, it would still violate the safety of Ag′'s backup plan action.

We note that if $\mathcal{F}_{\text{Ag}}(u, a_i)$ is set, Ag cannot send a conflict request by the last condition. Even though Ag does not send a request, it must use the information that the flag has been set in the agent's Action Selection Strategy. After a complete

exchange of conflict requests, each agent will be a part of a cluster of agents. The agent is bidding for its priority to take its intended actions among the agents that make up this cluster. These clusters of agents are defined as follows:

Definition 14 (Conflict Cluster)**.** A *conflict cluster* for an agent Ag is defined as $C_{Ag} = \{Ag' \in \mathfrak{A} \mid Ag \ \texttt{send} \ \text{Ag}' \text{ or } Ag' \ \texttt{send} \ Ag\}$, where Ag `send` Ag' implies Ag has sent a conflict request to Ag′. An agents' conflict cluster defines the set of agents in its bubble that an agent is in conflict with.

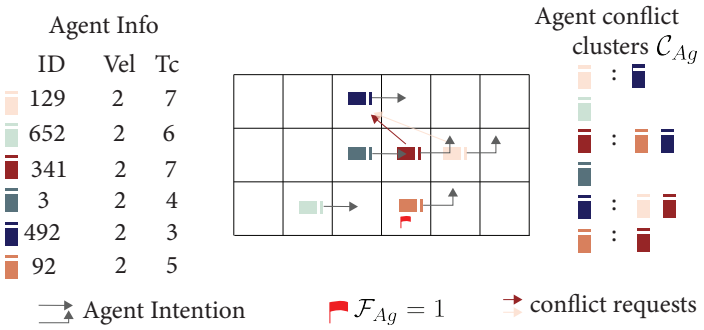

Figure 3.7: An example scenario with agents in a given configuration of agents, their intended actions, and their respective conflict clusters.

By the rules agents must follow to send conflict cluster requests, all agents in one agent's cluster must be a part of that agent's bubble and vice versa. Fig. 3.7 shows an example scenario and each agents' conflict clusters. Once the conflict requests have been sent and an agent can thereby identify the other agents in its conflict cluster, it needs to establish whether or not the conflict resolution has resolved in its favor.

Token Resolution

Once an agent has determined which agents are in its conflict cluster, it must determine whether or not it has the priority to take its intended action. The token resolution scheme is the way in which agents determine whether they have this 'right.'

The conflict resolution strategy must be designed to be fair, meaning each agent always eventually wins a conflict resolution and gets priority. The resolution is therefore based on the agents' token counts `Tc`, which is updated by agents to represent how many times an agent has been unable to take a forward progress action thus far. In particular, token counts are updated as follows:

Definition 15 (Token Count Update)**.** *The token count updates according to the agent's chosen action. In particular, if Ag selects action a: if* $O_{Ag,\,\textit{forward progress}}(s, a, u) = T$*, the agent's token count resets to 0, otherwise it increases by 1.*

Then, a fair strategy would be to give agents with the highest amount of tokens precedence. Formally, this can be written as follows: for each $\mathrm{Ag} \in \mathfrak{A}$, let $\mathcal{W}_{\mathrm{Ag}} \in \mathbb{B}$ be an indicator variable for whether or not the agent has won in its conflict cluster. Let $\mathtt{Tc}_{Ag}$ represent the token count of the agent when it has sent its request. Let $\mathtt{Id}_{Ag}$ represent a unique ID number of an agent. The conflict cluster resolution indicator variable $\mathcal{W}_{\mathrm{Ag}}$ is determined as follows:

$$\mathcal{W}_{\mathrm{Ag}} \triangleq \forall \mathrm{Ag}' \in \mathcal{B}_{\mathrm{Ag}}(s) :$$
$$(\mathtt{Tc}_{\mathrm{Ag}'} < \mathtt{Tc}_{\mathrm{Ag}}) \vee ((\mathtt{Tc}_{\mathrm{Ag}'} = \mathtt{Tc}_{\mathrm{Ag}}) \wedge \mathtt{Id}_{\mathrm{Ag}'} < \mathtt{Id}_{\mathrm{Ag}}).$$

The agent with the highest token count is defined as the winner of the agents' conflict cluster and any ties are broken via an agent ID comparison. The following lemmas, which come from the definition of the conflict-cluster resolution scheme, are helpful for proving safety of the agent protocol.

The first lemma states that an agent cannot send (or receive) a conflict request to (from) an agent outside its bubble.

Lemma 3. *Let us consider agent Ag at state s and agent Ag′ at state s′.*
Ag `send` $Ag' \Rightarrow Ag \in \mathcal{B}_{Ag'}(s')$.

Proof: If A `send` B, this means that all of the conditions specified in Section 3.5 must hold. Further, it means $(A, a_i) \dagger (B, a_i')$. This condition is only valid if $\mathrm{proj}_G s \in \mathcal{G}_{F,B}(B)$ or $\mathrm{proj}_G s \in \mathcal{G}_{F,BP}(B)$ hold. Membership of Agent A's state in either of these sets implies $A \in \mathcal{B}(B)$.

The following lemma follows from the lemma above.

Lemma 4. *At most one agent will win in each agent's conflict cluster.*

Proof: W.l.o.g., let us consider an agent Ag and its respective conflict cluster $C(\mathrm{Ag})$. It follows from Lemma 3 that $\forall \mathrm{Ag}'$ s.t. Ag `send` Ag′, then $\mathrm{Ag}' \in \mathcal{B}_{Ag}(s)$ and $\mathrm{Ag} \in \mathcal{B}_{Ag'}(s')$. It also follows that $\forall \mathrm{Ag}'$ s.t. Ag `send` Ag′, then $\mathrm{Ag} \in \mathcal{B}_{\mathrm{Ag}'}(s')$ and $\mathrm{Ag}' \in \mathcal{B}_{\mathrm{Ag}}(s)$. This means that an agent has access to all token counts and IDs of all agents in its conflict cluster, and all agents in its conflict cluster have access to

the agent's token count and ID. The conflict resolution implies that all agent edges are incident to the winning agent, where edges point to the agent they cede to. This implies that at most one agent can be the winner of each cluster. Less than one winner (per conflict cluster) will occur when an agent that is in the intersection of more than one conflict cluster wins.

The next section defines how each agent uses information from the conflict cluster resolution scheme to ultimately select an action.

Action-Selection Strategy

The purpose of the agent Action Selection Strategy is to define whether or not an agent is allowed to take its intended action a_i and if it is not, which alternative action it should take. The action-selection strategy is defined to coordinate agents so that lane-change maneuvers can be performed safely.

In the case where an agent is not allowed to take a_i, the agent is restricted to take either: the best straight action a_{st}, which is defined in Definition 16, or its backup plan action a_{bp}. The action-selection process that determines which of the three actions an agent Ag will choose is determined by the following five conditions:

1. a_i, the agent's and other agents' (in its bubble) intended actions, which have been selected via the behavioral profile and consistent evaluating function defined in Section 3.5.
2. Ag's role in conflict request cluster being:
   - A conflict request sender ($\exists \text{Ag}' \in \mathcal{B}_{\text{Ag}}(s) : \text{Ag}$ `send` Ag').
   - A conflict request receiver ($\exists \text{Ag}' \in \mathcal{B}_{\text{Ag}}(s) : \text{Ag}'$ `send` Ag).
   - Both a sender and a receiver of conflict requests.
   - Neither a conflict request sender nor receiver.
3. The agent's conflict cluster resolution $\mathcal{W}_{\text{Ag}}$.
4. Evaluation of $O_{\text{Ag,dynamic safety}}(s, a_i, u)$.
5. $\mathcal{F}_{\text{Ag}}(u, a_i)$ for Ag is raised, where $\mathcal{F}_{\text{Ag}}(u, a_i)$ is the maximal-yielding-not-enough flag defined in Section 3.5.

The Action Selection Strategy decision tree, shown in Fig. 3.8, defines how agents should select which action to take based on the five different conditions.

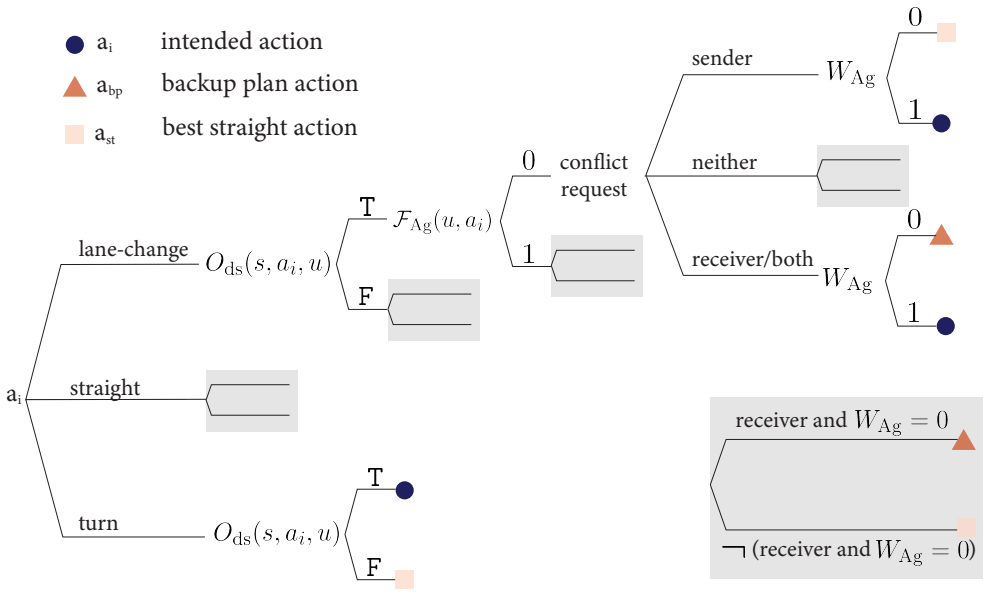

Figure 3.8: Agent action selection strategy.

The best straight action a_{st}, one of the three allowable actions the agent can take according to the action-selection strategy, is defined as follows:

Definition 16 (Best Straight Action)**.** Let us consider Ag and its associated action set $\rho_{\text{Ag}}(s)$. The *best straight action* is the action $a \in \rho_{\text{Ag}(s)}$ that is the highest-ranked action according to the consistent-evaluating function defined on the profile in Fig. 3.6, among the set of all actions for which γ_{Ag} = straight. In this case, the specification for dynamic safety $O_{\text{Ag},\, t,\, \text{dynamic safety}}(s, a, u)$ is included.

The action-selection strategy thus defines the conditions in which an agent is able to take its intended action a_i, its best straight action a_{st}, or when it is expected to yield and take its backup plan action a_{bp}. For instance, as one example, when an agent is both a receiver of a conflict request and a loser in its conflict cluster, the action-selection strategy dictates that the agent must yield (choose its backup plan action a_{bp}) to the agent that it received the conflict request from.

The agent protocol, as described in the above sections, has been designed in a way that formal guarantees on the safety and liveness of all agents can be proven if all agents are taking actions according to it. The proofs of safety and liveness are given in the following sections.

3.6 Safety Guarantees

Safety is guaranteed when agents do not collide with one another. An agent causes collision when it takes an action that satisfies the following conditions.

Definition 17 (Collision)**.** An agent Ag that takes an action $a \in Act_{\text{Ag}}$ will cause *collision* if the grid point occupancy of Ag ever overlaps with the grid point occupancy of another agent Ag' or a static obstacle O_{st}.

A strategy where agents simply take actions that avoid collision in the current time-step is insufficient for guaranteeing safety because of the inertial properties of the agent dynamics. The agent protocol has therefore been defined so an agent also avoids violating the safety of its own and any other agent's backup plan action a_{bp} defined in Section 3.3. An agent's backup plan action a_{bp} is evaluated to be safe when the following conditions hold:

Definition 18. *[Safety of a Backup Plan Action]* Let us define *the safety of an agent's backup plan action* $S_{\text{Ag},bp} : \mathcal{U} \rightarrow \mathbb{B}$, where $\mathbb{B} = \{\texttt{T}, \texttt{F}\}$ is an indicator variable that determines whether an agent's backup plan action is safe or not. It is defined as follows:

$$S_{Ag,bp}(u) = \wedge_{o \in O} o(s, a_{bp}, u),$$

where the set O is the set of all oracles in the top three tiers of the behavioral profile defined in Fig. 3.6.

An agent Ag takes an action $a \in Act_{\text{Ag}}$ that violates the safety backup plan action of another agent Ag$'$ when the following conditions hold:

Definition 19 (Safety Backup Plan Violation Action)**.** Let us consider an agent Ag that is taking an action $a \in Act_{\text{Ag}}$, and another agent Ag$'$. The action $(\text{Ag}, a) \perp \text{Ag}'$, i.e. agent Ag violates the safety backup plan of an agent Ag$'$ when by taking an action a, then $\mathcal{S}_{\text{Ag}',bp}(u') = \texttt{F}$, where u' is the state of the game after Ag has taken its action. In other words, by taking the action, the agent has ended in a state such that it violates the safety of its own or another agents' backup plan action.

The safety proof is based on the premise that all agents only take actions that do not collide with other agents and maintain the invariance of the safety of their own *and* other agents' safety backup plan actions. The safety theorem statement and the proof sketch are as follows.

We can treat the quasi-simultaneous game as a program, where each of the agents are separate concurrent processes. A safety property for a program has the form $P \Rightarrow \Box Q$, where P and Q are immediate assertions. This means if the program starts with P true, then Q is always true throughout its execution [64].

Theorem 5 (Safety Guarantee)**.** *Given all agents $Ag \in \mathfrak{A}$ in the quasi-simultaneous game select actions in accordance to the agent protocol specified in Section 3.5, we can show the safety property $P \Rightarrow \Box Q$, where the assertion P is an assertion that the state of the game is such that $\forall Ag, S_{Ag,bp}(s,u) = T$, i.e. each agent has a backup plan action that is safe, as defined in Definition 18. We denote P_t as the assertion over the state of the game at the beginning of the time-step t, before agents take their respective actions. Q is the assertion that the agents never occupy the same grid point in the same time-step implying that collision never occurs when agents take their respective actions during that time-step. We denote Q_t as the assertion for the agent states/actions taken at time-step t.*

The following is a proof sketch. Note, the full proof can be found in Appendix B.

Proof: To prove an assertion of this form, we need to find an invariant assertion I for which i) $P \Rightarrow I$, ii) $I \Rightarrow \Box I$, and iii) $I \Rightarrow Q$ hold. We define I_t as the assertion that all agents are taking actions that 1) do not collide with other agents and 2) do not violate the safety backup plan of other agents $\forall \text{Ag}, S_{\text{Ag},bp}(u') = \text{T}$ where $s' = \tau_{\text{Ag}}(s,a)$, and u' is the corresponding global state of the game after each Ag has taken its respective action a.

It suffices to assume:

1. Each $\text{Ag} \in \mathfrak{A}$ has access to the traffic light states.
2. There is no communication error in the conflict requests, token count queries, and the agent intention signals.
3. All intersections in the road network R are governed by traffic lights.
4. The traffic lights are designed to coordinate traffic such that if agents respect the traffic light rules, they will not collide.
5. Agents follow the agent dynamics defined in Section 3.3.
6. For $t = 0$, $\forall \text{Ag} \in \mathfrak{A}$ in the quasi-simultaneous game is initialized to:

- Be located on a distinct grid point on the road network.
- Have a safe backup plan action a_{bp} such that $S_{\text{Ag},bp}(s,u) = \texttt{T}$.

We can prove $P \Rightarrow \Box Q$ by showing the following:

1. $P_t \Rightarrow I_t$. This is equivalent to showing that if all agents are in a state where P is satisfied at time t, then all agents will take actions at time t where the I holds. This can be proven using arguments based on the agent protocol showing that each agent will always take actions that 1) do not collide with other agents and 2) will not violate the safety of its own or other agents' backup plan action.
2. $I \Rightarrow \Box I$. If agents take actions such that at time t, the assertion I_t holds, then by the definition of the assertion I, agents will end up in a state where at time t+1, assertion P holds, meaning $I_t \Rightarrow P_{t+1}$. Since $P_{t+1} \Rightarrow I_{t+1}$ from 1, we get $I \Rightarrow \Box I$.
3. $I \Rightarrow Q$. If all agents take actions according to the assertions in I, then collisions will not occur. This follows from the definition of I.

Proof of safety alone is not sufficient reason to argue for the effectiveness of the protocol, as all agents could simply stop for all time and safety would be guaranteed. A liveness guarantee, i.e. proof that all agents will eventually make it to their final destination, is critical. In the following section, we present liveness guarantees.

3.7 Liveness Guarantees

Note, we introduce the definition of liveness, from [64], as follows:

Definition 20 (Liveness)**.** A *liveness* property asserts that program execution eventually reaches some desirable state.

For this work, the eventual desirable state for each agent is to reach their respective final destinations. Proving fairness, as described in [64], is proving that each action will always terminate, and is fundamental for proving liveness. Additionally for liveness, the absence of 1) deadlocks and 2) collisions also need to be proved. Deadlock occurs when agents indefinitely wait for resources held by other agents [73]. Since the Manhattan grid road network has loops, agents can enter a configuration in which each agent in the loop is indefinitely waiting for a resource held

by another agent. When the density of agents in the road network is high enough, deadlocks along these loops will occur. We can therefore guarantee liveness only when certain assumptions hold on the density of the road network.

Definition 21 (Sparse Traffic Conditions)**.** Let M denote the number of grid points in the smallest loop (defined by legal orientation) of the road network, not including grid points $g \in \mathcal{S}_{\text{intersections}}$. The *sparse traffic conditions* must be such that $N < M - 1$, where N is the number of agents in the road network. The number of agents has to be such that the smallest loop does not become completely saturated, in which deadlock would occur. Note, these sparsity conditions are conservative because they are define based on the worst possible assignment of agents and their destinations.

Now, we introduce the liveness guarantees under these sparse traffic conditions. The proof of liveness is based on the fact that 1) behavioral profile include progress specifications and 2) conflict precedence is resolved by giving priority to the agent that has waited the longest time (a quantity that is reflected by token counts).

Theorem 6 (Liveness Under Sparse Traffic Conditions)**.** *Under the Sparse Traffic Assumption in Definition 21 and given all agents* $Ag \in \mathfrak{A}$ *in the quasi-simultaneous game select actions in accordance to the agent protocol specified in Section 3.5, liveness is guaranteed, meaning all* $Ag \in \mathfrak{A}$ *will always eventually reach their respective goals.*

The following is a proof sketch.
Proof: It suffices to assume:

1. $\forall \text{Ag} \in \mathfrak{A}$, $\forall \text{Ag}' \in \mathbb{B}_{Ag}$ in road segments, and $\forall Ag'$ within a local region around the agent as defined in Section 3.3 at intersections, Ag has access to other agents' state and intended action.
2. Each $\text{Ag} \in \mathfrak{A}$ has access to the traffic light states.
3. There is no communication error in the conflict requests, token count queries, and the agent intention signals.
4. For $t = 0$, $\forall \text{Ag} \in \mathfrak{A}$ in the quasi-simultaneous game is initialized to:
    - Be located on a distinct grid point on the road network.
    - Have a safe backup plan action a_{bp} such that $S_{\text{Ag},bp}(u) = \texttt{T}$.

5. The traffic lights are red a window of time Δt_{tl} such that $t_{\min} < \Delta t_{\text{tl}} < \infty$, where $t_{\min}$ is defined so agents are slowed down long enough so agents that have been waiting at a red light can eventually take a lane-change action. More details can be found in Appendix B.

6. The static obstacles are not on any grid point $g \in G$ where $g.d = 1$.

7. Each Ag treats its respective goal $\texttt{goal}_{\text{Ag}}$ as a static obstacle.

8. Bundles in the road network $\mathfrak{R}$ have no more than 2 lanes.

9. All intersections in the road network $\mathfrak{R}$ are governed by traffic lights.

and prove:

1. The invariance of a no-deadlock state follows from the sparsity assumption and the invariance of safety (no collision) follows from the safety proof.

2. Inductive arguments related to control flow are used to show that all Ag will always eventually take $a \in Act_{\text{Ag}}$ where $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

    a) Let us consider a road segment $rs \in RS$ that contains grid point(s) $g \in \mathcal{S}_{\text{sinks}}$ meaning that the road segment contains grid points with sink nodes. Inductive arguments based on the agents' longitudinal distance to destination grid points are used to show that every Ag $\in r$ will be able to always eventually take $a \in Act_{\text{Ag}}$ for which the forward progress oracle $O_{\text{forward progress}}(s, a, u) = \texttt{T}$.

    b) Let us consider a road segment $rs \in RS$. Let us assume $\forall rs' \in RS$ for which $(rs, rs') \in G_{\text{dep}}$, there is always eventually clearance on rs', meaning all road segments that agents on rs depend on to have clearance always eventually have it. We can use inductive arguments based on agents' longitudinal distance to the front of the intersection to show any Ag on rs will always eventually take $a \in Act_{\text{Ag}}$ where the forward progress oracle $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

    c) For any $\mathfrak{R}$ where the dependency graph G_{dep} (as defined in Definition 11) is a directed-acyclic-graph (DAG), inductive arguments based on the linear ordering of road segments $rs \in G_{\text{dep}}$ are used to prove that all Ag $\in \mathfrak{A}$ will always eventually take $a \in Act_{\text{Ag}}$ for which the forward progress oracle $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

d) When the graph G_{dep} is cyclic, the Sparsity Assumption (in Definition 21) breaks the cyclic dependency and allows for the similar induction arguments in 2c to apply.

3. By the above inductive arguments and the definition of the forward progress oracle $O_{\text{Ag, forward progress}}(s, a, u)$, all Ag will always eventually take actions that allow them to make progress towards their respective destinations.

Features of the agent protocol, like fairness from the conflict-cluster resolution and eventual satisfaction of all oracles in the behavioral profile are used for the arguments in the proof. The full proof can be found in Appendix B.

3.8 Simulation Environment

In order to streamline discrete-time multi-agent simulations, we have built a traffic game simulation platform called Road Scenario Emulator (RoSE). This emulator offers an easy-to-use, simple, modular interface. We use RoSE to generate different game scenarios and simulate how agents will all behave if they each follow the agent strategy protocol introduced in this work.

A road network environment, complete with legal lane orientations, intersections, and traffic lights, can be specified via a CSV file. The specified (by the user) road network environment forms a map data structure graph, which decomposes the roads into bundles, mentioned in Section 3.5.

The map will automatically parse the boundaries and lane directions of the road network to define where agents can either spawn from or exit the road network. In each game scenario, agents will randomly spawn according to a specified spawn rate. Each agent has the attributes described in Section 3.3 (i.e. goal destination, ID, dynamic parameters, etc.). The specific dynamic parameters are described in the following section.

Each game scenario is comprised of the road network graph and a set of agents (constantly changing over time as new agents spawn and old agents reach their goals and leave). The game is simulated forward for a specified number of time steps and the traces from the simulation are saved. The animation module in RoSE animates the traces from the simulated game.

Simulation Results

We simulate the game with randomized initialization of spawning agents at the source nodes for three different road network environments: 1) the straight road segment, 2) the small city blocks grid, and 3) the large city blocks grid. A snapshot of a small city blocks grid simulation is shown in Fig. 3.9, 3.10, and 3.11, respectively. The agent attributes are as follows: $v_{\min} = 0$, $v_{\max} = 3$, $a_{\min} = -1$, and $a_{\max} = 1$. For each road network environment, we simulate the game 100 times for $t = 250$ time-steps. During each time-step, agents will spontaneously spawn with some defined probability p at the source nodes. Each agent that spawns onto the road network is randomly assigned a sink node as its desired destination.

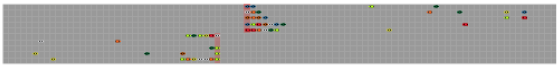

Figure 3.9: Straight road map environment.

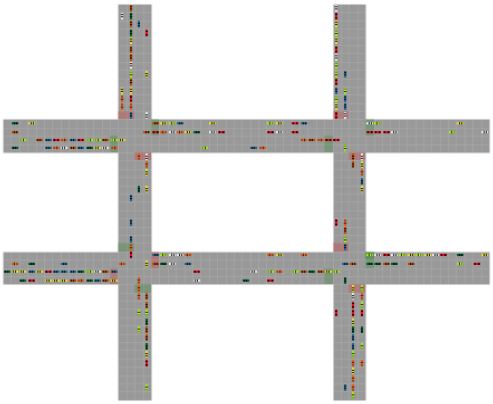

Figure 3.10: Simulation

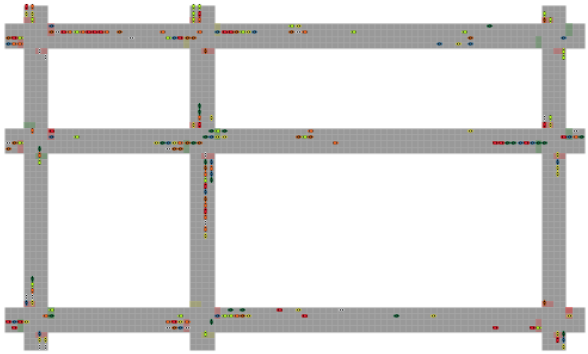

Figure 3.11: City blocks map environment.

Agents that make it to their respective sink node destinations will exit the map. For all simulation trials, collision does not occur. Although liveness is only guaranteed in sparse traffic conditions, we simulate for a number of agents N for which $N > M-1$. Even with a number of agents much greater than the number specified in the sparsity assumption, deadlock still does not occur. In particular, over the 100 trials for each of the maps, on average 77%, 36%, and 43% of agents made it to their respective destinations on the respective maps by the end of the 250 time-steps.

3.9 Conclusion

In this work, we have defined a comprehensive framework for introducing behavioral protocols for autonomous agents to follow. In particular, we introduced a quasi-simultaneous game, which is a turn-based game where the turns are defined based on agent states. The road network environment includes any grid-road network with traffic intersections. The specific agent class has specific attributes like limited perception and communication capabilities, a token count, and specific dynamic maneuvers. The behavioral protocol is then the process agents use to select which action to take in their given state. In this process, the agent uses its behavioral profile to select an intended action, conflict-cluster resolution to resolve conflicts, and the action selection strategy to ultimately select an action to take. If all agents are selecting actions according to the protocol, safety and progress of all agents can be guaranteed. Proofs of the safety and liveness properties are presented and verified in simulation.

Chapter 4

PERCEPTION: SEMANTIC ESTIMATION

4.1 Introduction

Conventional simultaneous localization and mapping (SLAM) algorithms rely on geometric measurements and require loop-closure detections to correct for drift accumulated over a vehicle trajectory. Semantic measurements can add measurement redundancy and provide an alternative form of loop closure.

In this chapter, we propose two different estimation algorithms that incorporate semantic measurements provided by vision-based object classifiers. The work presented in this chapter has been published in [14] and was done in joint collaboration with Alexei Harvard.

We assume we have an *a-priori* map of regions where the objects can be detected. The first estimation framework is posed as a maximum likelihood problem, where the likelihood function for semantic measurements is derived from the confusion matrices of the object classifiers. In this case, the semantic measurements improve the performance of conventional SLAM algorithms by incorporating measurements that provide additional information about the vehicle state. This method is described in further depth in Section 4.2. The second estimation framework is a hierarchical formulation that serves a dual purpose because it 1) leverages semantic measurements to improve the accuracy of state estimation and 2) uses state estimation to improve the certainty of object detection events. In particular, it is comprised of two parts: 1) a continuous-state estimation formulation that includes semantic measurements as a form of state constraints and 2) a discrete-state estimation formulation used to compute the certainty of object detection measurements using a Hidden Markov Model (HMM). The details of this technique can be found in Section 4.3.

Before introducing the two different methods, we introduce the problem formulation. Consider the traditional localization and mapping algorithm, where the goal is to simultaneously estimate the vehicle poses $\mathcal{X} \triangleq \{x_t\}_{t=0}^{T}$ and the position of a set of landmarks in the environment denoted by $\mathcal{L} \triangleq \{l_m\}_{m=1}^{M}$, given a set of continuous measurements $\mathcal{Z}_{\mathrm{c}} \triangleq \{z_{\mathrm{c},t}\}_{t=0}^{T}$. The landmarks are features in the environment that can easily be recognized, and the continuous measurements are range measurements to those landmarks. Note that $x_t \in \mathbb{R}^n$, where n is the dimension of the vehicle state,

and M is the number of landmarks in the environment.

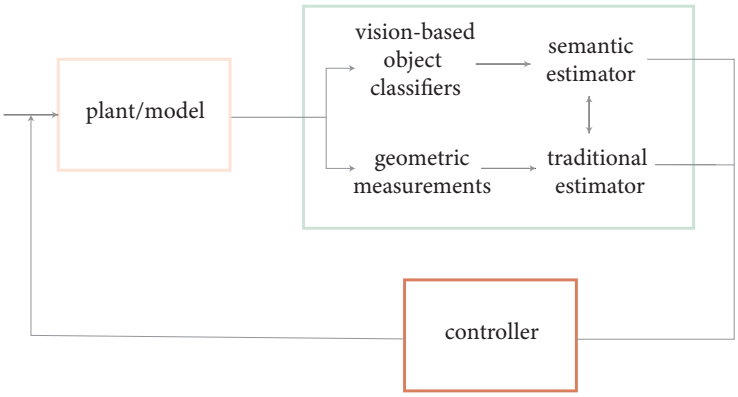

Figure 4.1: Model for the inclusion of semantic measurements in a traditional plant-controller system.

For this model, we assume that we have a set of odometry measurements given by $\mathscr{B} \triangleq \{b_t\}_{t=0}^{T}$ to approximate the vehicle dynamics, where b_t gives the vehicle translation and rotation between discrete-time points of the vehicle trajectory. These odometry measurements can be given by methods like the iterative closest point (ICP) algorithm. This estimation problem is typically formulated as the following maximum likelihood problem:

$$\hat{\mathscr{X}}, \hat{\mathscr{L}} = \text{argmax}_{\mathscr{X},\mathscr{L}} \log p(\mathscr{Z}_c | \mathscr{X}, \mathscr{L}). \tag{4.1}$$

In our problem, we consider a vision-based object classification algorithm that can detect K different objects given by $\mathscr{O} \triangleq \{o_k\}_{k=1}^{K}$. We assume that we have an *a priori* map that defines the positions of each object o_k and the corresponding region R_k where the object can be detected. We define the set of semantic measurements as $\mathscr{Z}_s \triangleq \{z_{s,t}\}_{t=0}^{T}$, where $z_{s,t} \in \mathbb{B}^K$. The measurement corresponding to the object detector of object o_k can be represented as a binary variable $z_{s,t}^{k} \in \{1, 0\}$, where a measurement of 1 indicates that the object o_k has been detected and 0 indicates that it has not.

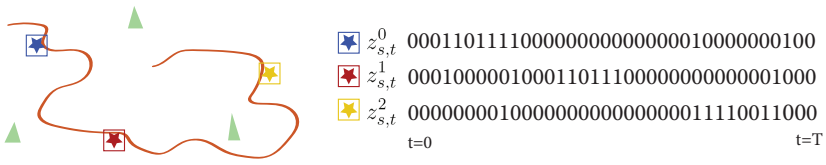

Figure 4.2: Object classification measurements modeled as binary measurements.

Each object detection measurement has a corresponding confusion matrix $C^k \in \mathbb{R}^{2\times 2}$ that captures the ratio of false positive and negative detections. Let the variable $v_t^k = g(x_t, o_k)$ be an indicator variable representing whether the object o_k is in the field of view of the camera and can be detected at time t that depends on the vehicle state x_t. The elements of the confusion matrix are defined as: $c_{iv^k}^k = p(z_{\mathrm{s},t}^k = i|x_t)$. We assume that these error statistics can be computed offline. In the case of a perfect classification algorithm, the confusion matrix would be the identity matrix. In the next section, we encode these semantic measurements into the traditional maximum likelihood formulation to add redundancy and robustness to the estimation algorithm.

4.2 Maximum Likelihood Formulation with Semantic Measurements

The first method we introduce for incorporating semantic measurements into SLAM algorithms is based on the maximum likelihood formulation. The estimation problem with semantic measurements can be formulated as a maximum likelihood problem:

$$\hat{\mathscr{X}}, \hat{\mathscr{L}} = \mathrm{argmax}_{\mathscr{X},\mathscr{L}} \log p(\mathscr{Z}_c, \mathscr{Z}_\mathrm{s}|\mathscr{X}, \mathscr{L}). \tag{4.2}$$

Assuming that the semantic measurements are independent from the continuous range measurements, the maximization problem can be rewritten as the following minimization problem:

$$\hat{\mathscr{X}}, \hat{\mathscr{L}} = \mathrm{argmin}_{\mathscr{X},\mathscr{L}} \sum_{t=0}^{T} \sum_{k=1}^{K} -\log(p(z_{\mathrm{s},t}^k|x_t)) - \log(p(\mathscr{Z}_c|\mathscr{X}, \mathscr{L})) - \log(p(\mathscr{B}|\mathscr{X})), \tag{4.3}$$

where the first term in the formulation corresponds to the likelihood function of the semantic measurements, and the second and third terms represent the nonlinear least-squares terms associated with the continuous measurements and odometry measurements, respectively. The likelihood function for the semantic measurements from object detector k can be derived from the object detector's confusion matrix C^k as follows:

$$p(z_{\mathrm{s},t}^k|x_t) = \prod_{a=\{0,1\}} \prod_{b=\{0,1\}} c_{a,b}^{\mathbb{1}(z_{\mathrm{s},t}^k=a)\mathbb{1}(v_t^k=b)}, \tag{4.4}$$

where v_t^k is a function of x_t since it represents whether the object o^k is in the field of view of the vehicle at the state x_t. This likelihood function selects each element of the confusion matrix based on the semantic measurement $z_{\mathrm{s},t}^k$ and v_t^k, which indicates whether the object is in the field of view.

Depending on whether the measurement $z^k_{s,t}$ is a 1 or a 0, the likelihood function takes on a different shape. The likelihood function shown in Fig. 4.3 assumes that the object is in the field of view when inside the region R_k associated with object o_k. Under this assumption, we can see how the likelihood function promotes the region corresponding to the detected object when the measurement $z^k_{s,t} = 1$ and demotes the region when $z^k_{s,t} = 0$.

For the likelihood estimation formulation, where we solve for Eqn. 4.3, we only include measurements where $z^k_{s,t} = 1$. The positive detection events are the measurements that give the most information, since they promote a very small region when they occur. Further, a measurement where $z^k_{s,t} = 0$ does not imply the vehicle is not in the region associated with the object. Instead, the vehicle could simply not have the object in its field of view, but still be in the region R_k.

Note that the likelihood function is a discrete, nonlinear function that must be approximated by a smooth function in order to be implemented in any factor-graph estimation algorithms like gtsam [47], which relies on gradient-based methods for solving the optimization problem. The details of this approximation are given in Appendix C.

Although this model improves the robustness of the estimation algorithm, the likelihood function does not take into consideration higher-level details about the measurements like their persistence over time. In the next section, we therefore introduce an alternative formulation for including object detection events as nonlinear factors that impose state constraints similar to loop closure detections.

4.3 Hierarchical Formulation with Semantic Measurements

In this section, we present a hierarchical estimation framework that is intended to serve two purposes: 1) use semantic measurements from object classifiers to improve the accuracy of estimation methods in a more robust way than the maximum likelihood method and 2) to use state estimates to improve the certainty of object classification events. The inner loop of this algorithm and outer loop are described in the remainder of this section, and the system architecture for the overall algorithm is defined in Section 4.4.

Inner-Loop: Continuous-State Estimation Process

The inner-loop continuous-state estimation process is very similar to the maximum-likelihood formulation, except with a slight modification to the factors associated

with semantic measurements and the introduction of switch variables.

In particular, we treat each object detection measurement $z^k_{s,t} = 1$ as a state constraint that is modeled to reflect the same shape as the factors in the maximum likelihood formulation. We do not consider measurements when $z^k_{s,t} = 0$ in our factor-graph formulation for the same reasons we excluded them in the maximum likelihood formulation. Since the standard factor-graph formulation does not allow for explicit state constraints, we introduce a relaxation, and use the following nonlinear least-squares factor to represent the constraint imposed by a positive object detection measurement:

$$f(z^k_{s,t}, x_t) = z^k_{s,t} f^k_1(x_t). \tag{4.5}$$

This semantic factor is defined to reflect the same properties as the discrete likelihood function described in Section 4.2. The comparison between the factors derived for the likelihood function and the factor derived here can be seen in Fig. 4.3. The factor $f_1(x_t)$ is defined as the following piecewise function:

$$f^k_1(x_t) = \begin{cases} 0 & d^k_h(x_t) = 0 \\ \alpha \exp(-\frac{\beta}{d^k_h(x_t)}) & d^k_h(x_t) > 0 \end{cases}, \tag{4.6}$$

where $d_h(x_t)$ is the shortest distance from x_t to the boundary of the region corresponding to object k given by R_k. Although the factor representing the likelihood function and the customized factor are similar, the customized factor improves the estimation accuracy further because of properties of its gradient.

The gradient of the function $f^k_1(x_t)$ is nonzero even when the estimate is far from the object detection region, so it still acts towards improving the estimate each time a positive object detection measurement occurs. Further, the parameters α and β can be modified to change the scale and rate of the inverse exponential functions, respectively. With these additional features, the new formulation with semantic measurements becomes the following minimization problem:

$$\hat{\mathscr{X}}, \hat{\mathscr{L}} = \text{argmin}_{\mathscr{X},\mathscr{L}} \sum_{t=0}^{T} \sum_{k=1}^{K} \|f(z^k_{s,t}, x_t)\| - \log(p(\mathscr{Z}_c|\mathscr{X}, \mathscr{L})) - \log(p(\mathscr{B}|\mathscr{X})). \tag{4.7}$$

False positive measurements will cause the wrong state-constraint factors to be imposed and will result in poor estimation results. We therefore introduce switchable constraints, taken from the loop-closure literature, to account for the possibility of bad measurements.

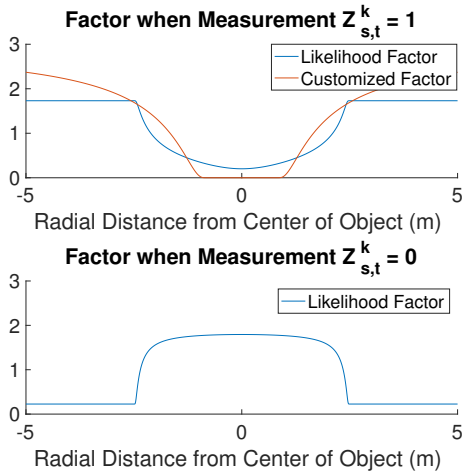

Figure 4.3: The nonlinear least squares factors added to the graph corresponding the semantic measurement $z^k_{\mathrm{s},t} = 1$ is in the top figure and $z^k_{\mathrm{s},t} = 0$ on the bottom. The factor added corresponding to the negative log likelihood function described in (C.1) and the customized factors formed by piecewise inverse exponential functions correspond to the blue and red plots, respectively. The parameter α for the inverse exponential factor is chosen to be 3.5 to approximate the same properties as the factor derived from the likelihood function.

In traditional SLAM algorithms, switchable constraints are introduced into the optimization formulation to improve the algorithm's performance when false positive data associations or loop-closure detections occur [87]. We propose the following addition of switchable constraints to improve the robustness of our algorithm to false semantic measurements:

$$\hat{\mathscr{X}}, \hat{\mathscr{L}}, \hat{\Gamma} = \operatorname{argmin}_{\mathscr{X},\mathscr{L},\Gamma} \sum_{t=0}^{T} \sum_{k=1}^{K} \Big(\|\psi(\gamma_t^k) f(z^k_{\mathrm{s},t}, x_t)\|_\Sigma + \|\gamma_t^k - \bar{\gamma}_t^k\|_\Lambda \Big) - \log(p(\mathscr{Z}_c|\mathscr{X}, \mathscr{L})) - \log(p(\mathscr{B}|\mathscr{X})) \quad (4.8)$$

where $\Gamma \triangleq \{\gamma_t\}_{t=0}^T$ is the set of all switch variables, $\gamma_t \in \mathbb{R}^K$, and $\bar{\Gamma} \triangleq \{\bar{\gamma}_t\}_{t=0}^T$ is the set of priors on the switch variables. The variables Σ and Λ are optimization hyperparameters that determine the weight of the factors corresponding to the state constraints and the switch variable priors. The function $\Psi : \mathbb{R} \mapsto [0, 1]$ is a function which takes a real value and maps it to the closed interval between 0 and 1. We choose $\Psi(\gamma_t^k) = \gamma_t^k$ to be a linear function of the switch variables and to constrain the switch variables between $0 \leq \gamma_t^k \leq 1$ since these choices have been empirically shown to work well [87].

Each switch variable γ_t^k quantifies the certainty of its associated semantic measurement $z^k_{\mathrm{s},t}$. When the switch variable γ_t^k is set to 0, the certainty in the measurement

is extremely low, and the influence of the state-constraint factor associated with the measurement $z_{s,t}^k$ gets disregarded. Probabilistically, the switch variable is modifying the information matrix associated with the semantic factor such that $\hat{\Sigma}^{-1} = \Psi(\gamma_t^k)^2\Sigma^{-1}$ [87]. This means the covariance of the semantic measurement is unchanged from Σ when $\Psi(\gamma_t^k) = 1$ and the certainty in the measurement is high, but scales with $\Psi(\gamma_t^k)^2$ when $\Psi(\gamma_t^k) < 1$. For typical object classifiers, since the rate of false positive measurements is relatively low, we can default to trusting the measurements, so the switch priors $\bar{\gamma}_t^k$ are set to 1. Thus, both the certainty of each object detection measurement and the pose estimates are optimized for in this framework.

Although the formulation in Eqn. 4.8 is more robust to semantic measurement errors, setting the prior on all switch variables to 1 will sometimes cause the optimization to converge to the wrong solution. If we can compute the certainty of each object detection measurement by leveraging both the error statistics of the object classifier algorithms and the vehicle dynamics, we can construct a more accurate prior on the switch variables. In the next section, we propose a formulation where we use an HMM to compute the marginal probabilities of object detection events. These marginal probabilities reflect a new certainty of object detection events, and improve the certainty of individual detection measurements. Furthermore, these marginal probabilities can be used to set the prior on the switch variables in the optimization framework.

Outer-Loop: Discrete-State Estimation Process

Higher-level properties of the estimation formulation, like the persistence of semantic measurements over time and the relative vehicle dynamics, can be used to improve the certainty of object detection measurements. In this section, we propose a discrete-state estimation framework.

In particular, we consider a discrete-state representation of the vehicle trajectory in terms of object detection events. The vehicle trajectory can be parsed into different object detection events based on the persistence of semantic measurements in $\mathcal{Z}_s$ over time. The continuous-state estimation, which we refer to as the lower-layer estimation framework, occurs on the time-scale of t whereas the discrete-state estimation, which we refer to as the higher-layer estimation framework, occurs on the time-scale of τ. This is also shown more clearly in Fig. 4.4. Once an object detection event has been detected, we represent the detection event with a discrete state s_τ. This discrete state s_τ has a time interval in the continuous-state estimation time

domain given by $t_\tau = [t_{\tau,i}, t_{\tau,f}]$ and a set of semantic measurements $Y_\tau \triangleq \{z_{s,t}\}_{t=t_{\tau,i}}^{t_{\tau,f}}$ that occur over the time interval t_τ. The semantic measurements associated with the object o_k during this time interval are defined as: $Y_\tau^k \triangleq \{z_{s,t}^k\}_{t=t_{\tau,i}}^{t_{\tau,f}}$.
The vehicle trajectory can then be represented as a sequence of states $\mathcal{S} \triangleq \{s_\tau\}_{\tau=1}^{Q}$, where Q is the number of object detection events that occur over the trajectory and $s_\tau = \{o_1, o_2, ..., o_k\}$. The notation $s_\tau = o_i$ means that object o_i has been seen during the detection event s_τ.

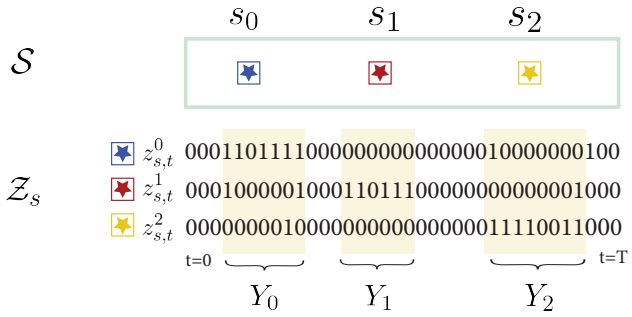

Figure 4.4: This figure shows the relation between the discrete-state HMM and the continuous-state estimation. The semantic measurements $\mathcal{Z}_s$ are illustrated by the grid, where each row represents the measurements received by an object classifier. The grey boxes represent $z_{s,t}^k = 1$ and the white boxes represent $z_{s,t}^k = 0$. We can see that an object detection event, which is represented as a discrete state in the HMM, corresponds to a time interval in the continuous-state estimation framework.

The time-sequence of object detection events can be modeled as an HMM since the memoryless-Markov property holds, i.e. $p(s_\tau|s_{0:\tau-1}) = p(s_\tau|s_{\tau-1})$. The HMM estimation formulation and the algorithms for computing the switch prior, which are used to improve the optimization formulation in Eqn. 4.8, are described in the following sections.

Parsing Trajectory into Object Detection Events

In this model, we assume that only one object detection event occurs at a given time. The trajectory $\mathcal{X}$ can be parsed into different object detection events based on the semantic measurements $\mathcal{Z}_s$. Each object classifier is associated with a sequence of measurements given by $\{z_{s,t}^k\}_{t=0}^{T}$ and a confusion matrix C^k that describes the algorithm's error statistics. In the event that the object o_k is visible, the frequency of nonzero measurements can be approximated by $C_{11}^k = p(z_{s,t}^k = 1|v_t^k = 1)$. We therefore define an object detection event for the object o_k as occurring when the proportion of nonzero measurements over a minimum time interval exceeds the

threshold value set by $C_{11}^k - \epsilon$. The value of ϵ is set to a value that depends on the certainty of the statistics given by the confusion matrix. The object detection event is terminated when the proportion of nonzero measurements decreases to less than the threshold value of $C_{11}^k - \epsilon$.

Transition and Observation Matrices

HMMs are typically defined by a single transition matrix and a single observation matrix. The hybrid nature of our estimation formulation means continuous states have been traversed between the discrete states representing object detection events, and that a set of semantic measurements, denoted by Y_τ, have elapsed during each object detection event. Since we want to incorporate continuous-state pose estimates and semantic measurements into our HMM formulation, the transition and observation matrices are time-varying and dependent on the lower-layer estimates and measurements. In particular, the transition matrices are a function of the pose-estimates between discrete states representing object detection events, and the observation matrices are a function of Y_τ, the semantic measurements that have elapsed during the time interval t_τ corresponding to the discrete state s_τ. Each element of the transition matrix between discrete states $s_{\tau-1}$ and s_τ is defined as follows:

$$\begin{aligned} A(s_{\tau-1}, s_\tau)_{ij} &\triangleq p(s_\tau = o_j | s_{\tau-1} = o_i, \hat{x}_{t_{\tau,i}}, \hat{x}_{t_{\tau-1,f}}) \\ &\propto \exp(-\frac{1}{2}\|d_{ij} - \hat{d}_{\tau-1,\tau}\|), \end{aligned} \tag{4.9}$$

where $d_{ij} = \|p_i - p_j\|^{\frac{1}{2}}$, and p_i and p_j denote the positions of the center of mass (COM) of the objects o_i and o_j respectively. The distance $\hat{d}_{\tau-1,\tau} \triangleq \|\hat{x}_{t_{\tau,i}} - \hat{x}_{t_{\tau-1,f}}\|^{\frac{1}{2}}$ is the estimated distance traveled between object detection events. The rows of the transition probability matrix are normalized to sum to one. The transition probability is defined by the error between the actual distance of two objects from each other and the estimated distance traveled from one object detection event to another. The definition of the transition matrix would have to be modified to accommodate for the objects whose regions are not centralized around the objects' COM, because the distance traveled between object detection events could vary considerably. Examples of such objects include sidewalk or road detectors. This will be considered in future work.

Each element of the observation matrix for the discrete state s_τ is defined as follows:

$$\begin{aligned} O(\tau)_{ij} &\triangleq p(y_\tau = o_i, Y_\tau | s_\tau = o_j) \\ &= p(y_\tau = o_i | Y_\tau) p(Y_\tau | s_\tau = o_j). \end{aligned} \tag{4.10}$$

The probability $p(Y_\tau|s_\tau = o_j)$ is the likelihood of a sequence of semantic measurements over the time interval t_τ given that the object detection event corresponds to object o_j. When conditioned on $s_\tau = o_j$, each measurement in the sequence Y_τ^j can be modeled as a Bernoulli random variable with the probability of a nonzero measurement given by C_{11}^j. Thus, the probability $p(Y_\tau|s_\tau = o_j)$ can be approximated by how well the sequence of measurements Y_τ^j fits a Bernoulli distribution with parameter $p = C_{11}^j$. This probability can be computed with a Chi-squared goodness of fit test [7].

Since there is no discrete-state observation of the system, we define y_τ to be a function of the sequence of measurements Y_τ. The discrete-state observation is the object that corresponds to the maximum likelihood for the sequence of semantic observations, which means $\bar{y}_\tau = \text{argmax}_{o_k} p(Y_\tau|s_\tau = o_k)$ for $k = 1, .., K$. Thus, the probability of a discrete-time measurement conditioned on the continuous-state observations becomes defined as follows:

$$p(y_\tau = o_i|Y_\tau) = \begin{cases} 1 & \text{if } o_i = \bar{y}_\tau, \\ 0 & \text{if } o_i \neq \bar{y}_\tau. \end{cases} \tag{4.11}$$

Therefore, an observation matrix for each discrete-state s_τ can be derived for the HMM as a function of the sequence of semantic measurements Y_τ.

Computing Marginal Probabilities

The Viterbi algorithm can be used to determine the most probable sequence of states given a set of observations. We use a modified version of the Viterbi formulation given as follows:

$$\hat{\mathcal{S}} = \text{argmax}_{\mathcal{S}} \sum_{\tau=1}^{\tau,f} \log(p(y_\tau, Y_\tau|s_\tau)) + \log(p(s_\tau|s_{\tau-1}, \hat{d}_{\tau-1,\tau})). \tag{4.12}$$

This equation accounts for the dependencies of the transition and observation matrices on the time-varying lower-layer estimates and measurements. Once the most-probable sequence of states for the HMM are derived from the Viterbi algorithm in Eqn. 4.12, the marginal probabilities $p(s_\tau = o_i|Y_{0:\tau,f})$ can be computed with dynamic programming using a modified version of the forward-backward algorithm. The variable $Y_{0:\tau,f}$ denotes all the measurement sequences corresponding to object detection events that have been observed. Details of this computation can be found in Appendix C.

The forward-backward algorithm therefore defines a certainty associated with the most-probable sequence of object detection events. The switch prior associated with the semantic measurements that occur over the time-intervals corresponding to these object detection events are computed in the following section.

Switch Prior Derivation

The switch prior γ_t^k associated with a semantic measurement $z_{\mathrm{s},t}^k$ quantifies the reliableness of the measurement. When an object detection event corresponding to object o_k occurs, the marginal probability from the forward-backward algorithm gives us $p(s_\tau = o_k | Y_{0:\tau,f})$, which is a metric for the certainty that object o_k for the time interval t_τ associated with the detection event. This means that over the time interval t_τ, the measurements where $z_{\mathrm{s},t}^k = 1$ should be proportional to the certainty of the detection event. Thus, we define the switch priors for the time interval t_τ where $s_\tau = o_k$ for every measurement for which $z_{\mathrm{s},t}^k = 1$ as follows:

$$\gamma_t^k = p(s_\tau = o_k | Y_{0:\tau,f}, y_{0:\tau,f}). \tag{4.13}$$

In the case where the certainty in the object detection event $s_\tau = o_k$ is high, the switch priors corresponding to $z_{\mathrm{s},t}^k = 1$ will be very close to 1.

The HMM formulation only gives a certainty metric for positive object detection events. The switch prior must also be derived for any semantic measurements where $z_{\mathrm{s},t}^k = 1$ and the measurement does not occur during a time interval specified by an object detection event. These measurements occur during a non-detection event which occurs over the time interval in the continuous-state estimation time domain given by $t_{\tau\emptyset} = [t_{\tau\emptyset,i}, t_{\tau\emptyset,f}]$, and has semantic measurements given by $Y_{\tau\emptyset}^k = \{z_{\mathrm{s},t}^k\}_{t=\tau\emptyset,i}^{T_f=\tau\emptyset,f}$. To compute the probability that all measurements during the time interval correspond to a non-detection event, which we define as $p(s_{\tau\emptyset} = \emptyset | Y_{\tau\emptyset}^k)$, we can compute how well the sequence of measurements in $Y_{\tau\emptyset}^k$ fits to a Bernoulli distribution with parameter C_{10}^k. The switch prior associated with the measurements outside the object detection are set to have a certainty proportional to $1 - p(s_{\tau\emptyset} = \emptyset | Y_{\tau\emptyset}^k)$. Thus, when the certainty that the non-object detection event has occurred is high, the switch priors corresponding to $z_{s,t}^k = 1$ will be very close to 0.

4.4 Hierarchical System Architecture

In this section, we summarize the hierarchical architecture. There are two estimation processes that are occurring on different time-scales: the continuous-state estimation process with switchable constraints and the discrete-state estimation of

object detection events. Each process is iteratively improving the other, and the dependencies of the two processes can be seen in Fig. 4.5.

Switch variables are introduced into the continuous-state estimation framework to improve the robustness of the algorithm to erroneous measurements [87]. The continuous-state optimization problem with semantic measurements and switchable constraints can be formulated as follows:

$$\hat{\mathcal{X}}, \hat{\mathcal{L}}, \hat{\Gamma} = \text{argmax}_{\mathcal{X},\mathcal{L},\Gamma} \log(p(\mathcal{Z}_s, \mathcal{Z}_c|\mathcal{X}, \mathcal{L}, \Gamma_0)). \tag{4.14}$$

Note, the switch variables are variables that represent the certainty of semantic measurements. We define the discrete-state estimation framework to define the priors on these switch variables, where the switch priors should be representative of the confidence we have in the semantic measurements.

The discrete-state estimation framework operates on the time scale of object detection events given by τ. Once an object-detection event has been classified, as described in Section 4.3, the pose-estimates and semantic measurements from Eqn. 4.14 are used to derive the transition and observation matrices of the HMM. This HMM is used to model the discrete-state representation of the trajectory. The most probable sequence of object detection events, represented by the set of discrete states $\mathcal{S}$, is then solved as follows:

$$\hat{\mathcal{S}} = \text{argmax}_{\mathcal{S}} \log(p(\mathcal{Z}_s, \mathcal{Z}_c|\mathcal{S}, \hat{\mathcal{X}})). \tag{4.15}$$

The forward-backward algorithm is then used to compute the marginal probabilities associated with the maximum likelihood sequence of object detection events.

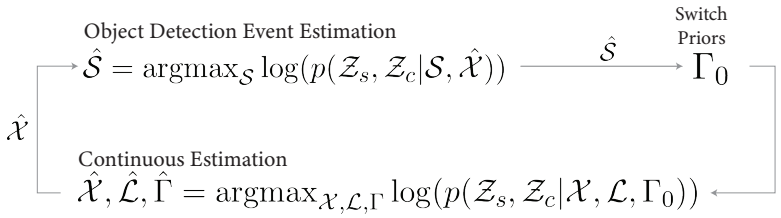

Figure 4.5: The system architecture is comprised of two layers: the lower layer shown in the lower box represents the factor-graph formulation with switchable constraints and updates at every time step t, whereas the higher layer shown in the top box represents the HMM estimation framework, and computes switch prior variables after every object detection event. The switch variables are denoted by Γ and the switch prior is given by Γ_0.

These marginal probabilities are used to compute priors on the switch variables associated with the semantic measurements in Eqn. 4.14. Therefore, the higher and lower-layer estimation processes are simultaneously improving the pose estimate and the certainty of object detection events.

4.5 Simulation Results

We investigate the performance of our algorithms in simulation. We consider an object classifier that can detect three different objects, each of which is associated with a known radial region in a 2-D map that is shown in Fig. 4.6, and each of which has a confusion matrix whose parameters are defined in Table 4.1.

Table 4.1: Confusion Matrix Parameters

	o_1	o_2	o_3
$p(z^k_{s,t} = 1 \mid v^k)$	0.02	0.03	0.05
$p(z^k_{s,t} = 0 \mid v^k)$	0.2	0.15	0.1

There are four landmarks that provide the vehicle with range measurements when detected. The data association problem for the landmarks are assumed to be solved in this formulation. The vehicle traverses an ellipsoid trajectory and goes through each of the object detection regions during its path. We introduce noisy odometry measurements in our simulation. The range measurements to the landmarks mitigate the estimation error when included in the traditional SLAM algorithms. We investigate how the introduction of semantic measurements improves the estimation even further.

In our simulations, we run three different algorithms to estimate the vehicle trajectory. First, we run the algorithm with range measurements to the four different landmarks, which is what is conventionally available in the SLAM community. In the simulation figures, this is referred to as 'landmarks only.' Second, we use the maximum likelihood formulation with the smoothed likelihood function described in Section 4.2. In the simulation figures, this is referred to as 'likelihood.' Finally, we test the hierarchical formulation which was described in Section 4.3. In the simulation figures, this is referred to as 'switch with HMM.' In our simulations, the estimation frameworks are implemented using gtsam, a factor-graph formulation commonly used for solving pose-graph estimation problems [22, 47].

For the different formulations, the noise model must be chosen for the factors

corresponding to the state constraints imposed by the semantic measurements. In the algorithm involving the likelihood function, we use the identity matrix to define the covariance on the likelihood factor so that the Bayesian representation of the likelihood function is preserved.

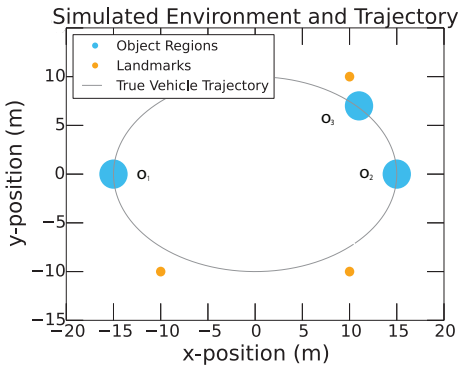

Figure 4.6: The objects and their corresponding regions of detections R_k are shown in blue and the positions of the landmarks that are visible during the vehicle trajectory are shown in orange. The gray line represents the true vehicle path.

For the formulation with switch variables, the noise on the prior for the switch variables, which is given by Λ in Eqn. 4.8, is chosen to be 0.01 when the switch prior values are chosen by the HMM and 10 when the switch prior is set to the default value of 1. This choice reflects our increase in certainty on the switch prior when we use the HMM formulation. The noise on the inverse-exponential factors, which is represented by Σ in Eqn. 4.8, is chosen to be 0.5.

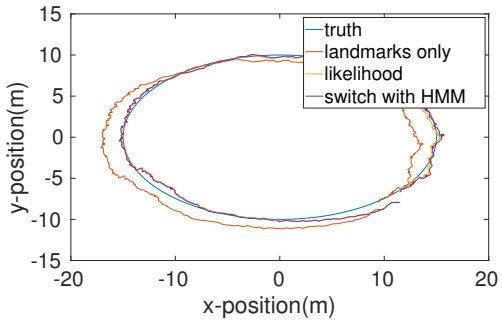

Figure 4.7: The true trajectory can be compared to the estimated trajectory using three different algorithms. The algorithms that incorporate the semantic measurements improve the estimate significantly.

The estimated trajectory resulting from the different estimation algorithms are shown in Fig. 4.7. The squared error between the estimated trajectories and the true

trajectory is shown in Fig. 4.8. We can see that the two algorithms that incorporate semantic measurements outperform the traditional SLAM algorithm. We also see that the estimation framework with the smooth approximation of the likelihood function and the estimation framework with the switch variables and HMM-derived switch priors converge to very similar local minima.

Since different noise on the odometry measurements will contribute to different estimation results, we compare our estimation algorithms on a set of randomly generated odometry measurements. This way, we can evaluate the performance of the different algorithms over many different trials.

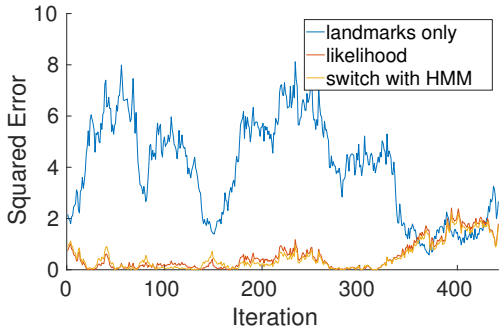

Figure 4.8: The squared error between the true trajectory and the estimated trajectories from the three different estimation estimation algorithms.

The comparison of the different estimation algorithms is captured in Fig. 4.9 and Fig. 4.10. We see how the mean of the squared error over the entire trajectory for all the trials is notably smaller when the semantic measurements use either the likelihood algorithm or the hierarchical algorithm.

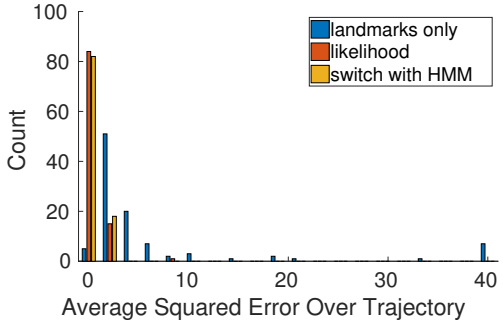

Figure 4.9: The mean squared error over the trajectory path is computed for each of the different algorithms and the results from the trials can be compared in this histogram. The likelihood estimation algorithm and the two-layer estimation algorithm perform similarly, and both perform significantly better than the algorithm without semantic measurements.

The estimation framework with the likelihood function has more trials with very low average-squared errors, but is less consistent than the two-layer estimation framework, since its distribution has higher variance. If we look at the squared error of the final position estimate, which is the value we are iteratively estimating, Fig. 4.10 shows that the HMM algorithm performs marginally better than the likelihood algorithm.

The main advantage of using the HMM formulation, however, is not to improve the performance of the likelihood formulation. Rather, it is to leverage information from the persistence of the semantic measurements as well as vehicle dynamics (i.e. distance traveled between object detection events) to improve the certainty of the measurements corresponding to object detection events. In this simulation, the sequence of events corresponding to the maximum likelihood given by the Viterbi algorithm was given as follows: $s_0 = o_2$, $s_1 = o_3$, and $s_2 = o_1$, meaning object 2 as detected first, followed by object 3, and then object 1. The corresponding marginal probabilities of these object detection events are $p(s_0 = o_2|Y_0) = 0.943$, $p(s_1 = o_3|Y_1) = 0.994$, and $p(s_2 = o_1|Y_2) = 0.997$. This shows certainty levels of object detection events that are much greater than the accuracy guaranteed by the c_{11}^k element in the confusion matrix for each object detector which were approximately 0.8 for each of the object classifiers in our simulation. These results highlight how the additional state information (how much distance was traveled between object detection events) can be leveraged to increase the confidence that an object detection measurement was correctly triggered.

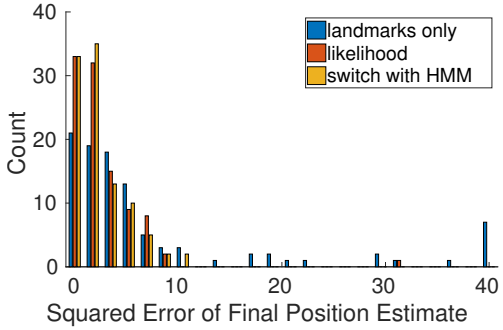

Figure 4.10: For every trial corresponding to different odometry noise measurements, we use the three algorithms to estimate the trajectory. The mean squared error of the final position estimate is computed. The final position estimate is marginally better using the two-layer estimation framework.

The effectiveness of semantic measurements will depend on the frequency at which

these objects are detected, but when objects have been detected, these estimation architectures proposed in this paper provide a robust way to incorporate these semantic measurements.

4.6 Conclusion

In summary, we have presented two methods of augmenting existing localization methods with semantic information from object classifiers. In particular, we have introduced a maximum likelihood approach where we model semantic measurements as binary measurements with probabilistic error based on the confusion matrix. In the second approach, we introduced a hierarchical formulation where the certainty of object detection events is computed. Then, this certainty is used to better reject false positive measurements in the localization estimation layer. The improvement of robustness and accuracy of localization is demonstrated in simulation.

Chapter 5

CONCLUSION AND FUTURE WORK

Designing safe and interpretable algorithms for autonomous vehicles is paramount to their successful integration into society. In this thesis, we have presented new frameworks for designing safe and interpretable algorithms—particularly in the decision-making and perception modules of autonomous vehicles.

5.1 Decision-Making

Designing the behavioral module for autonomous vehicles is especially challenging because of the complex coupling among many agents at once in highly-interactive settings. Without any baseline assumptions about how all agents are operating, we cannot provide any formal guarantees of the entire system. We therefore propose a top-down design approach for the decision-making module where all agents are expected to behave according to a behavioral contract.

The first questions to answer in designing this behavioral contract include: 1) what specifications (or rules) should each agent follow and 2) what prioritization structure should exist on these rules so they behave "correctly" (i.e. are safe) and it is clear why an agent selects a particular action. In Chapter 2, we have specified a methodology for ordering specifications that govern the high-level behaviors of autonomous vehicles. If specifications are hierarchically ordered into a profile with the mathematical properties that we proposed, the actions agents are selecting from can be compared to one another in a consistent manner. In this way, there is a unique line of reasoning for agents to use when selecting actions. An assume-guarantee contract says if the environment can be assumed to behave according to some class of profiles, then the self-driving car can guarantee that it will behave according to its own profile. Blame is defined as the case where an agent does not act according to its assumed profile. Finally, we provide some examples of how cars following these assume-guarantee profiles might behave in game-theoretic experiment settings. We show, through these simple game-theoretic scenarios that behavioral profiles are insufficient for the successful and safe coordination of multiple agents. They also fail to guarantee the progress of the agents.

In Chapter 3, we propose a solution to address these limitations. In particular,

we define a comprehensive framework that defines 1) a new quasi-simultaneous game paradigm where turn-order is based on agent states, 2) the road network environment agents are assumed to be operating on, and 3) an extended version of the assume-guarantee profile introduced in Chapter 2 that additionally constrains agent behaviors. The assume-guarantee contract (protocol) is defined for a single class of agents with specific agent attributes. The protocol definition includes the region an agent must reason over (i.e. the bubble), how the agent chooses its intended action (via the assume-guarantee profile), and how it ultimately selects which action to take. The action selection process depends on a conflict-resolution scheme that depends on the querying of other agent intentions and tokens. With this protocol, we formally guarantee safety and progress (under sparse traffic conditions) for all agents. We validate the safety and liveness guarantees in a randomized simulation environment.

5.2 Perception

With regard to perception, we introduce a robust estimation framework for incorporating probabilistic binary measurements, which are used to model data from vision-based object classification algorithms. We first introduce a formulation for solving a maximum likelihood problem with the semantic measurement likelihood function modeled after the confusion matrix of object classifiers. We then derive a two-part estimation framework where the lower-layer is formulated as a factor-graph estimation problem, with each measurement corresponding to a state-constraining factor modeled after the discrete likelihood function and a switchable constraint. We also present a higher-layer estimation framework that takes into account measurement persistence and vehicle dynamics to compute the certainty of sets of semantic measurements corresponding to object detection events. These certainties capture which measurements are false positives, and are used to compute the switch priors in the lower-layer estimation algorithm. The advantage of including the higher-layer estimation framework is demonstrated in the presented numerical simulation. We show in simulation how the addition of semantic measurements in this framework improves the robustness and accuracy of the estimated trajectory.

5.3 Future Work

Robust Behavioral Contracts

The current framework relies on strong assumptions about the agents and the road network in order for the safety and liveness properties to hold. Real-life systems

have multiple types of agents and are also especially prone to both unintentional and intentional agent error, especially when humans are in the loop. In order to make the proposed behavioral contract paradigm relevant to deployment in the real world, the framework must be extended to accommodate for a heterogeneous set of agents and be robust to errors. One approach would be to introduce some probabilistic modeling over the agent behaviors and dynamics, and designing the protocol to ensure some probabilistic lower bounds on safety and performance guarantees.

A separate approach for dealing with errors specifically could involve defining different classes of agent error (i.e. perception error, communication error, and violations of agent protocols). Then some type of perturbation analysis could be used to determine the protocol's sensitivity to these different classes of errors. Perhaps some notion of compositionality could be used to establish overlap among these different classes of errors. The sensitivity analysis would hopefully highlight how the system should be modified to be more robust to errors in the system.

Data-Informed, Human-Robot Behavioral Contracts

It is safe to assume that humans will never strictly act in accordance with a behavioral protocol like the one defined in the thesis. Further, human intentions are not always perfectly communicated. These are just a few of many reasons why behavioral contract design with humans in-the-loop will be extremely challenging. Luckily, a tremendous amount of traffic and human driving data has recently become readily available. A future direction for this would be defining a way to leverage this data to define a human-robot behavioral contract that guarantees correct agent behavior. This relaxed version of the fully-autonomous agent protocol would depend on inferring human internal states and understanding the behaviors associated with those states (learned from data). An effective approach for defining these contracts would allow us to define probabilistic lower bounds on safety and performance guarantees.

Robust System-Level Design

Designing any module in the autonomy stack (like the behavioral module) in isolation from the other modules (perception, controls, etc.) will invariably encourage sub-optimality and failure. In this thesis, the decision-making module and the perception modules were studied in isolation. For the decision-making module, the perception module was abstracted away and vice versa (for the perception module). We would also like to design modules in a way that is complementary (where infor-

mation passed between modules are thought to more actively inform one another). This framework could potentially offer significantly more robustness in the case of system failures. Simple examples of this methodology could include designing motion-planning schemes that enhance an agent's perceptual understanding of its surroundings (i.e. certainty objects that might be hidden by occlusions, certainty of object classifications). Along the same vein, an agents' semantic context and understanding of its environment can be leveraged to gain certainty over how it should behave (i.e. if an agent has a notion of what objects are possibly occluded, then it can motion-plan more conservatively).

BIBLIOGRAPHY

[1] E. Ackerman. *Fatal Tesla Self-Driving Car Crash Reminds Us That Robots Aren't Perfect*. 2016. URL: https://spectrum.ieee.org/cars-that-think/transportation/self-driving/fatal-tesla-autopilot-crash-reminds-us-that-robots-arent-perfect (visited on 07/01/2016).

[2] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. "Robust map optimization using dynamic covariance scaling." In: *2013 IEEE International Conference on Robotics and Automation*. May 2013, pp. 62–69. DOI: 10.1109/ICRA.2013.6630557.

[3] N. Arechiga. "Specifying safety of autonomous vehicles in signal temporal logic." In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. Paris, France: IEEE, 2019, pp. 58–63.

[4] N. Atanasov, B. Sankaran, J. Le Ny, G. J. Pappas, and K. Daniilidis. "Nonmyopic view planning for active object classification and pose estimation." In: *IEEE Transactions on Robotics* 30.5 (Oct. 2014), pp. 1078–1090. ISSN: 1552-3098. DOI: 10.1109/TRO.2014.2320795.

[5] N. Atanasov, M. Zhu, K. Daniilidis, and G. J. Pappas. "Semantic localization via the matrix permanent." In: *Proc. of Robotics and Science Systems*. 2014.

[6] C. Baier and J-P. Katoen. *Principles of Model Checking*. Cambridge, Massachussetts: MIT Press, 2008.

[7] N. Balakrishnan, V. Voinov, and M. S. Nikulin. *Chi-Squared Goodness of Fit Tests with Applications*. Academic Press, 2013.

[8] D. P. Bertsekas. "The auction algorithm: A distributed relaxation method for the assignment problem." In: *Annals of operations research* 14.1 (1988), pp. 105–123.

[9] C. Boutilier. "Planning, learning and coordination in multiagent decision processes." In: *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*. 1996, pp. 195–210.

[10] Craig Boutilier. "Sequential optimality and coordination in multiagent systems." In: *IJCAI*. 1999, pp. 478–485.

[11] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. "Probabilistic data association for semantic SLAM." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 1722–1729. DOI: 10.1109/ICRA.2017.7989203.

[12] L. Buşoniu, R. Babuška, and B. De Schutter. "Multi-agent reinforcement learning: An overview." In: *Innovations in Multi-Agent Systems and Applications-1*. Springer, 2010, pp. 183–221.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age." In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[14] K. X. Cai, A. Harvard, R. M. Murray, and S-J. Chung. "Robust Estimation Framework with Semantic Measurements." In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 3809–3816. URL: https://ieeexplore.ieee.org/document/8814793.

[15] K. X. Cai, T. Phan-Minh, S-J. Chung, and R. M. Murray. "Rules of the Road: Safety and Liveness Guarantees for Autonomous Vehicles." In: *arXiv preprint arXiv:2011.14148* (2021). URL: https://arxiv.org/pdf/2011.14148v2.pdf.

[16] L. Carlone, A. Censi, and F. Dellaert. "Selecting good measurements via l1relaxation: A convex approach for robust estimation over graphs." In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2014, pp. 2667–2674. DOI: 10.1109/IROS.2014.6942927.

[17] A. Censi, S. Bolognani, J. G. Zilly, S. S. Mousavi, and E. Frazzoli. "Today me, tomorrow thee: efficient resource allocation in competitive settings using karma games." In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand: IEEE, 2019, pp. 686–693.

[18] A. Censi, K. Slutsky, T. Wongpiromsarn, D. Yershov, S. Pendleton, J. Fu, and E. Frazzoli. "Liability, ethics, and culture-aware behavior specification using rulebooks." In: *arXiv e-prints* (Feb. 2019), arXiv:1902.09355.

[19] K. M. Chandy and J. Misra. "The drinking philosophers problem." In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 6.4 (1984), pp. 632–646.

[20] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, Z. Liu, H. I. Christensen, and F. Dellaert. "Multi robot object-based SLAM." In: *2016 International Symposium on Experimental Robotics*. Springer International Publishing, 2017, pp. 729–741.

[21] J. Civera, D. Gálvex-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel. "Towards semantic SLAM using a monocular camera." In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sept. 2011, pp. 1277–1284. DOI: 10.1109/IROS.2011.6094648.

[22] F. Dellaert and M. Kaess. *Factor Graphs for Robot Perception*. Vol. 6. Now Publishers, Jan. 2017, pp. 1–139.

[23] W. van Der Hoek, M. Roberts, and M. Wooldridge. "Social laws in alternating time: Effectiveness, feasibility, and synthesis." In: *Synthese* 156.1 (2007), pp. 1–19.

[24] N. E. Du Toit and J. W. Burdick. "Robot motion planning in dynamic, uncertain environments." In: *IEEE Transactions on Robotics* 28.1 (2011), pp. 101–115.

[25] M. Duering and P. Pascheka. "Cooperative decentralized decision making for conflict resolution among autonomous agents." In: *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*. IEEE. 2014, pp. 154–161.

[26] S. Ekvall, P. Jensfelt, and D. Kragic. "Integrating active mobile robot object recognition and SLAM in natural environments." In: *IEEE Intl. Conf. on Intelligent Robots and Systems*. 2006.

[27] I. Filippidis. "Decomposing formal specifications into assume-guarantee contracts for hierarchical system design." PhD thesis. California Institute of Technology, 2019.

[28] C. Finn, S. Levine, and P. Abbeel. "Guided cost learning: deep inverse optimal control via policy optimization." In: *International Conference on Machine Learning*. New York, New York, USA: JMLR, 2016, pp. 49–58.

[29] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan. "Hierarchical game-theoretic planning for autonomous vehicles." In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9590–9596.

[30] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[31] A. Glover, W. Maddern, M. Warren, S. Reid, M. Milford, and G. Wyeth. "OpenFABMAP: An open source toolbox for appearance-based loop closure detection." In: *2012 IEEE International Conference on Robotics and Automation*. May 2012, pp. 4730–4735. DOI: 10.1109/ICRA.2012.6224843.

[32] P. J. Gmytrasiewicz and P. Doshi. "A framework for sequential planning in multi-agent settings." In: *Journal of Artificial Intelligence Research* 24 (2005), pp. 49–79.

[33] N. A. Greenblatt. "Self-driving cars and the law." In: *IEEE Spectrum* 53.2 (Feb. 2016), pp. 46–51. ISSN: 0018-9235. DOI: 10.1109/MSPEC.2016.7419800.

[34] C. Guestrin, D. Koller, and R. Parr. "Multiagent planning with factored MDPs." In: *Advances in Neural Information Processing Systems* 14 (2001), pp. 1523–1530.

[35] C. Guestrin, S. Venkataraman, and D. Koller. "Context-specific multiagent coordination and planning with factored MDPs." In: *AAAI/IAAI*. 2002, pp. 253–259.

[36] David Guichard. "An introduction to combinatorics and graph theory." In: *Whitman College-Creative Commons* (2017).

[37] J. Y. Halpern and M. Kleiman-Weiner. "Towards formal definitions of blameworthiness, intention, and moral responsibility." In: *arXiv e-prints*, arXiv:1810.05903 (Oct. 2018).

[38] J. Hardy and M. Campbell. "Contingency planning over probabilistic obstacle predictions for autonomous road vehicles." In: *IEEE Transactions on Robotics* 29.4 (2013), pp. 913–929.

[39] M. Harris. *NTSB Investigation Into Deadly Uber Self-Driving Car Crash Reveals Lax Attitude Toward Safety*. 2019. URL: https://spectrum.ieee.org/cars-that-think/transportation/self-driving/ntsb-investigation-into-deadly-uber-selfdriving-car-crash-reveals-lax-attitude-toward-safety (visited on 11/07/2019).

[40] M. Herman, V. Fischer, T. Gindele, and W. Burgard. "Inverse reinforcement learning of behavioral models for online-adapting navigation strategies." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3215–3222.

[41] M. W. Hofbaur and B. C. Williams. "Mode estimation of probabilistic hybrid systems." In: *International Workshop on Hybrid Systems: Computation and Control*. 2002.

[42] C. A. Holt and A. E. Roth. "The Nash equilibrium: A perspective." In: *Proceedings of the National Academy of Sciences* 101.12 (2004), pp. 3999–4002.

[43] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller. "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles." In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 1671–1678.

[44] D. Isele, A. Nakhaei, and K. Fujimura. "Safe reinforcement learning on autonomous vehicles." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1–6.

[45] R. Ivanov, N. Atanasov, M. Pajic, G. Pappas, and I. Lee. "Robust estimation using context-aware filtering." In: *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. Sept. 2015, pp. 590–597. DOI: 10.1109/ALLERTON.2015.7447058.

[46] Radoslav Ivanov, Nikolay Atanasov, Miroslav Pajic, James Weimer, George J Pappas, and Insup Lee. "Continuous estimation using context-dependent discrete measurements." In: *IEEE Transactions on Automatic Control* 64.1 (2018), pp. 238–253.

[47] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. "iSAM2: Incremental smoothing and mapping using the Bayes tree." In: *The International Journal of Robotics Research* 31.2 (2012), pp. 216–235.

[48] M. Kuderer, S. Gulati, and W. Burgard. "Learning driving styles for autonomous vehicles from demonstration." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2641–2646.

[49] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g 2 o: A general framework for graph optimization." In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 3607–3613.

[50] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. "Real-time motion planning with applications to autonomous urban driving." In: *IEEE Transactions on Control Systems Technology* 17.5 (2009), pp. 1105–1118.

[51] M. Kwiatkowska. "Cognitive reasoning and trust in human-robot interactions." In: *TAMC 2017*. Springer, 2017.

[52] M. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: verification of probabilistic real-time systems." In: *CAV*. Vol. 6806. LNCS. Springer, 2011, pp. 585–591.

[53] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P.H.S. Torr, and M. Chandraker. "Desire: Distant future prediction in dynamic scenes with interacting agents." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 336–345.

[54] S. Lefèvre, D. Vasquez, and C. Laugier. "A survey on motion prediction and risk assessment for intelligent vehicles." In: *ROBOMECH* 1.1 (2014), p. 1.

[55] D. Lenz, F. Diehl, M. T. Le, and A. Knoll. "Deep neural networks for Markovian interactive scene prediction in highway scenarios." In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2017, pp. 685–692.

[56] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard. "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems." In: *IEEE Transactions on Control Systems Technology* 26.5 (2017), pp. 1782–1797.

[57] W. Li, D. Sadigh, S. S. Sastry, and S. A. Seshia. "Synthesis for human-in-the-loop control systems." In: *TACAS*. Springer Berlin Heidelberg, 2014, pp. 470–484. ISBN: 978-3-642-54862-8.

[58] A. Lomuscio, H. Qu, and F. Raimondi. "MCMAS: An open-source model checker for the verification of multi-agent systems." In: *International Journal on Software Tools for Technology Transfer* 19.1 (2017), pp. 9–30.

[59] O. Madani, S. Hanks, and A. Condon. "On the undecidability of probabilistic planning and related stochastic optimization problems." In: *Artificial Intelligence* 147.1-2 (2003), pp. 5–34.

[60] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. "FastSLAM: A factored solution to the simultaneous localization and mapping problem." In: *AAAI/IAAI* 593598 (2002).

[61] D. Morgan, G. P. Subramanian, S-J. Chung, and F. Y. Hadaegh. "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming." In: *The International Journal of Robotics Research* 35.10 (2016), pp. 1261–1285.

[62] S. M. Oh, S. Tariq, B. N. Walker, and F. Dellaert. "Map-based priors for localization." In: *IEEE Conf. on Intelligent Robots and Systems*. 2004.

[63] E. Ohn-Bar and M. M. Trivedi. "Looking at humans in the age of self-driving and highly automated vehicles." In: *T-IV* 1.1 (Mar. 2016), pp. 90–104. ISSN: 2379-8904. DOI: 10.1109/TIV.2016.2571067.

[64] S. Owicki and L. Lamport. "Proving liveness properties of concurrent programs." In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.3 (1982), pp. 455–495.

[65] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli. "A survey of motion planning and control techniques for self-driving urban vehicles." In: *IEEE Transactions on Intelligent Vehicles* 1.1 (Mar. 2016), pp. 33–55.

[66] C. H. Papadimitriou and J. N. Tsitsiklis. "The complexity of Markov decision processes." In: *Mathematics of Operations Research* 12.3 (1987), pp. 441–450.

[67] D. C. Parkes and J. Shneidman. "Distributed implementations of vickrey-clarke-groves mechanisms." In: *Proceedings of the Third Joint Conference on Autonomous and Multiagent Systems*. IEEE. 2004.

[68] T. Patten, M. Zillich, R. Fitch, M. Vincze, and S. Sukkarieh. "Viewpoint evaluation for online 3-D active object classification." In: *IEEE Robotics and Automation Letters* 1.1 (Jan. 2016), pp. 73–81. ISSN: 2377-3766. DOI: 10.1109/LRA.2015.2506901.

[69] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Elsevier, 2014.

[70] Hans Peters. *Game theory: A multi-leveled approach*. Springer, 2015.

[71] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart. "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 2096–2101.

[72] T. Phan-Minh, K. X. Cai, and R. M. Murray. "Towards assume-guarantee profiles for autonomous vehicles." In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. Nice, France: IEEE, 2019, pp. 2788–2795. URL: https://ieeexplore.ieee.org/abstract/document/9030068.

[73] S. A. Reveliotis and E. Roszkowska. "On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems." In: *IEEE Transactions on Automatic Control* 55.7 (2010), pp. 1646–1651.

[74] P. Robbel and D. Roy. "Exploiting feature dynamics for active object recognition." In: *2010 11th International Conference on Control Automation Robotics Vision*. Dec. 2010, pp. 2102–2108. DOI: 10.1109/ICARCV.2010.5707373.

[75] D. Sadigh, A. D. Dragan, S. Sastry, and S. A. Seshia. "Active preference-based learning of reward functions." In: *Robotics: Science and Systems (RSS)*. Berlin, Germany, 2013.

[76] D. Sadigh, K. Driggs-Campbell, A. Puggelli, W. Li, V. Shia, R. Bajcsy, A. Sangiovanni-Vincentelli, S. S. Sastry, and S. Seshia. "Data-driven probabilistic modeling and verification of human driver behavior." In: *AAAI Spring Symposium Series*. 2014.

[77] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan. "Planning for autonomous cars that leverage effects on human actions." In: *Robotics: Science and Systems (RSS)*. Vol. 2. Ann Arbor, MI, USA, 2016.

[78] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan. "Information gathering actions over human internal state." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 66–73.

[79] Y. E. Sahin and N. Ozay. "From drinking philosophers to wandering robots." In: *arXiv preprint arXiv:2001.00440* (2020).

[80] W. Schwarting, J. Alonso-Mora, and D. Rus. "Planning and decision-making for autonomous vehicles." In: *Annual Review of Control, Robotics, and Autonomous Systems* (2018).

[81] W. Schwarting and P. Pascheka. "Recursive conflict resolution for cooperative motion planning in dynamic highway traffic." In: *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2014, pp. 1039–1044.

[82] M. F. Shakun. "Unbounded rationality." In: *Group Decision and Negotiation* 10.2 (2001), pp. 97–118.

[83] S. Shalev-Shwartz, S. Shammah, and A. Shashua. "On a formal model of safe and scalable self-driving Cars." In: *arXiv e-prints*, arXiv:1708.06374 (Aug. 2017), arXiv:1708.06374. arXiv: 1708.06374 [cs.RO].

[84] S. Shalev-Shwartz, S. Shammah, and A. Shashua. "Safe, multi-agent, reinforcement learning for autonomous driving." In: *arXiv preprint arXiv:1610.03295* (2016).

[85] Y. Shoham and M. Tennenholtz. "On social laws for artificial agent societies: off-line design." In: *Artificial Intelligence* 73.1-2 (1995), pp. 231–252.

[86] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

[87] N. Sünderhauf and P. Protzel. "Switchable constraints for robust pose graph SLAM." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Oct. 2012, pp. 1879–1884. DOI: 10.1109/IROS.2012.6385590.

[88] S. Thrun. "Probabilistic robotics." In: *Communications of the ACM* 45.3 (2002), pp. 52–57.

[89] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. "Context-based vision system for place and object recognition." In: *Computer Vision, IEEE International Conference on*. Vol. 2. IEEE Computer Society. 2003, pp. 273–273.

[90] P. Trautman and A. Krause. "Unfreezing the robot: Navigation in dense, interacting crowds." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2010, pp. 797–803.

[91] P. Trautman, J. Ma, R. M. Murray, and A. Krause. "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation." In: *The International Journal of Robotics Research* 34.3 (2015), pp. 335–356.

[92] J. Tumova, G. C. Hall, S. Karaman, E. Frazzoli, and D. Rus. "Least-violating control strategy synthesis with safety rules." In: *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*. Philadelphia, Pennsylvania, USA: ACM, 2013, pp. 1–10.

[93] C-I. Vasile, Jana Tumova, S. Karaman, C. Belta, and D. 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.

[94] N. Vlassis. "A concise introduction to multiagent systems and distributed artificial intelligence." In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 1.1 (2007), pp. 1–71.

[95] T. Wongpiromsarn and E. Frazzoli. "Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications." In: *CDC*. Dec. 2012, pp. 7644–7651. DOI: 10.1109/CDC.2012.6426524.

[96] T. Wongpiromsarn, S. Karaman, and E. Frazzoli. "Synthesis of provably correct controllers for autonomous vehicles in urban environments." In: *ITSC*. Oct. 2011, pp. 1168–1173.

[97] T. Wongpiromsarn, K. Slutsky, E. Frazzoli, and U. Topcu. "Minimum-violation planning for autonomous systems: theoretical and practical considerations." In: *arXiv preprint arXiv:2009.11954* (2020).

[98] T. Wongpiromsarn, A. Ulusoy, C. Belta, E. Frazzoli, and D. Rus. "Incremental synthesis of control policies for heterogeneous multi-agent systems with linear temporal logic specifications." In: *ICRA*. May 2013, pp. 5011–5018.

[99] M. Wooldridge and W. Van Der Hoek. “On obligations and normative ability: Towards a logical analysis of the social contract.” In: *Journal of Applied Logic* 3.3-4 (2005), pp. 396–420.

Appendix A

DECISION-MAKING: BEHAVIORAL PROFILES

A.1 Examples of Consistent Evaluators

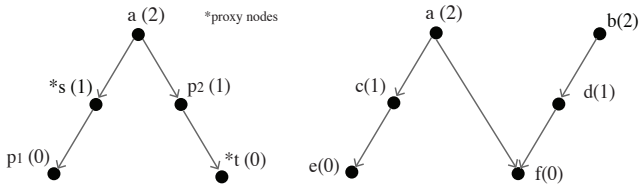

Figure A.1: Both posets admit consistent evaluators. The value the consistent evaluator assigns on each singleton that consists of the node is written in parentheses. Note that the poset on the left has the additional graded property while the one on the right does not.

Consider the posets in Fig. A.1. Any evaluator f that assigns the values according to the values in the parentheses, shown in the figure, can easily be verified to have properties 1-4 of consistent evaluation. On the left poset, we can also see that property 5 is also satisfied since for every pair of nodes such that $f(\{p_1\}) < f(\{p_2\})$ implies that there exist proxy nodes s and t such that $f(\{p_1\}) = f(\{t\})$, $f(\{p_2\}) = f(\{s\})$ and $p_1 \prec s$ and $p_2 \prec t$. This relation can be easily seen for the nodes p_1 and p_2 in Fig. A.1. The same statement applies to the poset on the right, which is not graded, but also happens to be consistently evaluable!

A.2 Adding Nodes to a Specification Structure

Since car-manufacturers will want to refine a given root specification structure, it is important to define the ways nodes (or specifications) can be added and still preserve the graded property of the specification structure. In Fig. A.2, we see some simple examples of how nodes can be added and the maximal chain property defined in Lemma 1 is maintained. When a node is added to an anti-chain with a given rank r, it must be valued less than at least one node with rank $r + 1$ and greater than at least one node of rank $r - 1$ if such nodes exit. This is clearly displayed in the top-left example and bottom-right example in Fig. A.2. When a specification is added in-between existing rankings to create a new ranking, the node must be

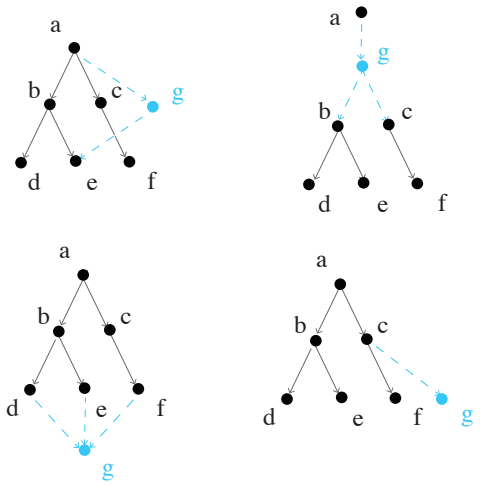

Figure A.2: Different ways a dimensional property can be added to a graded poset and still preserve the graded property.

compared to all nodes in the rank above and the rank below in a manner that is consistent with the existing partial order. This can be seen in the top right and left bottom examples in Fig. A.2. Oftentimes comparisons of a new node with existing nodes in a specification structure will result in a poset that is no longer graded. When the resulting poset is no longer graded, we introduce a way to make minimal modifications to the poset such that it regains its graded property. We mean that the modifications are minimal in the sense that they do not significantly redefine the existing relationships between nodes. In the particular scenario where a node

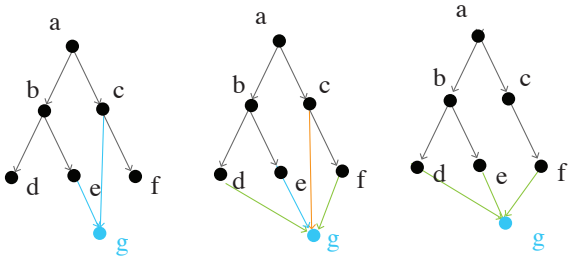

Figure A.3: The steps taken to add the node g such that $g < e$ and $g < c$. The orange edges are deleted and the green edges are added to minimally change the poset to a graded poset.

is added such that it has a lower value than two nodes of ranks with a difference of one as shown in Fig. A.3, it is best to preserve the edge with the node in the poset with smaller rank, remove the edge of the node with higher rank, and redefine that comparison via a proxy node. We can see it can be burdensome to exhaustively

define all the different ways a node could be compared to existing nodes in a graded poset. Here are some general guidelines to follow when trying to add or remove edges to regain the graded property of the poset:

1. If an edge is redundant (i.e. the comparison is already defined via another node), then remove it.

2. Add edges between incomparable nodes of the poset.

A.3 Adding Edges to a Specification Structure

The same guidelines for resolving improperly added nodes applies to edges as well. Note that the only incomparable nodes are ones of the same rank.

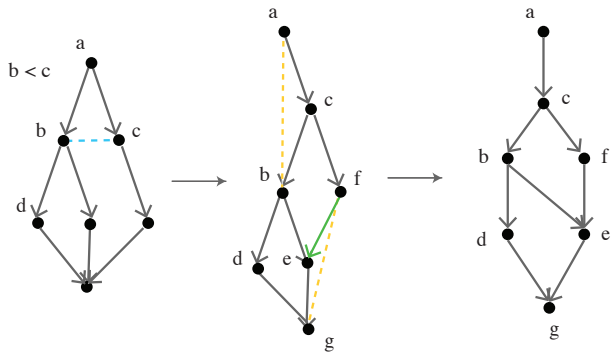

Figure A.4: This shows the sequence of steps to add in the edge $b < c$ into the existing poset on the left. Orange edges are removed and green edges are added.

The example shown in Fig. A.4 shows how adding a comparison between the nodes b and c causes the set to no longer have the graded property. By adding edges between incomparable nodes and also removing redundant edges, the gradedness of the poset returns.

Appendix B

DECISION-MAKING: BEHAVIORAL CONTRACTS

B.1 Road Network

The following defines the set of properties that grid points can have.

Grid Point Properties

The set of properties $\mathcal{P}_g = \{p, d, \texttt{lo}\}$ of each grid point $g \in G$. $p \in \mathbb{Z}^2$ denotes the Cartesian coordinate of the grid point, $d \in \{0, 1\}$ is an indicator variale that defines whether or not the grid point is drivable, `lo` is the legal orientation, where the legal orientation is an element of the set $\{\texttt{north}, \texttt{east}, \texttt{south}, \texttt{west}\}$. The set `lo` may be empty when the grid point is not drivable.

The road network is hierarchically decomposed into lanes and bundles, which are defined informally as follows:

- Lanes: Let lane $La(g)$ denote a set of grid points that contains all grid points that are in the same 'lane' as g. $La(g) = \{g' | \text{proj}_x(g'.p) = \text{proj}_x(g.p)$ or $\text{proj}_y(g.p) = \text{proj}_y(g.p)$, $g'.\phi_l = g.\phi_l$, $g.\texttt{drivable} = g'.\text{drivable} = 1\}$.
- Bundles: First, we define the set of adjacent lanes to lane $La(g)$ as $\texttt{adj}(La(g)) = \{La(g') \mid \exists e = (\hat{g}, \hat{g}') \in \mathfrak{R}$ s.t. $(\hat{g} \in La(g), \hat{g}' \in La(g'))$ and $\hat{g}.\phi_l = \hat{g}'.\phi_l\}$. This represents the set of lanes $La(g)$ in the same direction that the lane is adjacent to. Let $N(g) = \texttt{adj}(\text{La(g)})$. Let bundle $Bu(g)$ denote a set of lanes that are all connected to one another and is defined recursively as follows:

$$Bu(g) = \begin{cases} La(g) \cup N(g) & \text{if } N(g) \neq \emptyset \\ La(g) & \text{otherwise.} \end{cases}$$

B.2 Bubble Construction

In order to define the bubble for the agent dynamics specified in Section 3.3, we present some preliminary definitions. We first introduce the backup plan node set (which is defined recursively) as follows:

Definition 22 (Backup Plan Node Set)**.** *Let $Ag \in \mathfrak{A}$ and $s_0 \in S_{Ag}$. The backup plan grid point set $BP_{Ag}(s_0)$ is all the grid points agent Ag occupies as it applies*

maximum deceleration to come to a complete stop. $BP_{Ag}(s_0) = \mathcal{G}_{Ag}(s_0, a_{bp}) \cup BP_{Ag}(\tau_{Ag}(s_0, a_{bp}))$ *if* $\tau_{Ag}(s_0, a_{bp}).v \neq 0$ *and* $BP_{Ag}(s_0) = \mathcal{G}_{Ag}(\tau(s_0, a_{bp}))$ *otherwise, where* a_{min} *is the agent's action of applying maximal deceleration while keeping the steering wheel at the neutral position.*

Definition 23 (Forward/Backward Reachable States)**.** *The (1-step) forward reachable state set of agent Ag denoted* $\mathcal{R}_{Ag}(s_0)$ *represents the set of all states reachable by Ag from the state* s_0*. The forward reachable set is defined as* $\mathcal{R}_{Ag}(s_0) \triangleq \{s \in S_{Ag} \mid \exists a \in \rho_{Ag}(s_0).s = \tau(s_0, a)\}$*. Similarly, we define the (1-step) backward reachable state set* $\mathcal{R}^{-1}_{Ag}(s_0)$ *as the set of all states from which the state* s_0 *can be reached by Ag. Formally,* $\mathcal{R}^{-1}_{Ag}(s_0) \triangleq \{s \in S_{Ag} \mid \exists s \in S_{Ag}.\exists a \in \rho_{Ag}(s).s_0 = \tau(s, a)\}$.

Definition 24 (Forward Reachable Nodes)**.** *We denote by* $\mathcal{G}^{\mathcal{R}}_{Ag}(s_0)$ *the forward reachable node set, namely, the set of all grid points that can be occupied upon taking the actions that brings the agent Ag from its current state* s_0 *to a state in* $\mathcal{R}_{Ag}(s_0)$*. Specifically,*

$$\mathcal{G}^{\mathcal{R}}_{Ag}(s_0) \triangleq \bigcup_{a \in \rho_{Ag}(s_0)} \mathcal{G}_{Ag}(s_0, a)$$

.

This set represents all the possible grid points that can be occupied by an agent in the next time step.

Definition 25 (Occupancy Preimage)**.** *For* $n \in G$*, where G are the nodes in the road network graph* $\mathfrak{R}$*, the occupancy preimage* $\mathcal{G}^{\mathcal{R}^{-1}}_{Ag}(n)$ *is the set of states of agent Ag from which there is an action that causes n to be occupied in the next time step. Formally,*

$$\mathcal{G}^{\mathcal{R}^{-1}}_{Ag}(n) = \{s \in S_{Ag} \mid \exists a \in \rho_{Ag}(s).n \in \mathcal{G}_{Ag}(s, a)\}$$

.

In the next section, we define several different sets of grid points that are defined to represent the locations where two agents may possibly interfere with one another, which are shown in Fig. B.1. The bubble is defined to be the union of these sets of grid points.

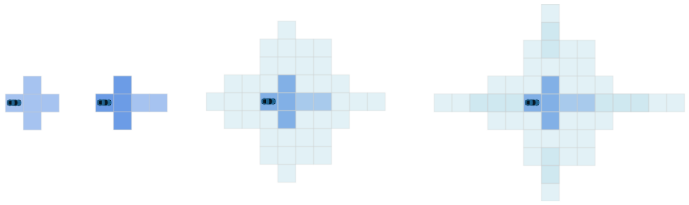

Figure B.1: Bubble if all Ag $\in \mathfrak{A}$ have the Agent Dynamics specified in Section 3.3. Construction of this set defined in the Appendix.

We begin by considering the ego agent whose bubble we are defining. In particular, let us again consider an agent Ag at state $s_0 \in S_{Ag}$. The corresponding grid point set $\mathcal{G}^{\mathcal{R}}_{\text{Ag}}(s_0)$ is shown in the left-most figure in Fig. B.1. The grid points an agent occupies when executing its backup plan from a state in the agent's forward reachable set $\mathcal{R}_{\text{Ag}}(s_0)$ is given by:

$$\mathcal{G}^{\mathcal{R},BP}_{\text{Ag}}(s_0) \triangleq \bigcup_{s \in \mathcal{R}_{\text{Ag}}(s_0)} BP_{\text{Ag}}(s).$$

These grid points are shown in the second from the left sub-figure in Fig. B.1. The set-valued map

$$\mathcal{Z}_{\text{Ag}}(s_0) \triangleq \mathcal{G}^{\mathcal{R}}_{\text{Ag}}(s_0) \cup \mathcal{G}^{\mathcal{R},BP}_{\text{Ag}}(s_0)$$

represents all the grid points an agent can possibly reach in the next state or in the following time step were it to execute its backup plan. Let Ag$' \in \mathfrak{A}$ and Ag$' \neq$ Ag. The set:

$$\mathcal{S}^{\mathcal{R}}_{\text{Ag}'}(\text{Ag}, s_0) \triangleq \bigcup_{n \in \mathcal{Z}_{\text{Ag}}(s_0)} \mathcal{G}^{\mathcal{R}^{-1}}_{\text{Ag}'}(n)$$

defines the set of all states in which another agent Ag$'$ can reach any grid point in the other agents' forward reachable grid points $\mathcal{Z}_{\text{Ag}}(s_0)$. Let us define the grid point projection of these states as

$$\mathcal{G}^{\mathcal{R}}_{\text{Ag}'}(\text{Ag}, s_0) \triangleq \{\mathcal{G}_{\text{Ag}'}(s) \mid s \in \mathcal{S}^{\mathcal{R}}_{\text{Ag}'}(\text{Ag}, s_0)\}$$

These grid points are defined in the third from the left subfigure in Fig. B.1.

The bubble also needs to include any state where an agent Ag$'$ where the agent has so much momentum it cannot stop fast enough to avoid collision with the agent Ag. To define the set of states from which this might occur, let us define the set:

$$\mathcal{S}^{BP}_{Ag'}(Ag, s_0) = \{s \in S_{Ag'} \mid BP_{Ag'}(s) \cap \mathcal{Z}_{Ag}(s_0) \neq \emptyset\}.$$

If another agent Ag′ occupies a state in this set, then execution of that agent's backup plan will cause it to intersect with the set of grid points that are in agents set $\mathcal{Z}_{\text{Ag}}(s_0)$. Let

$$\mathcal{S}^{\mathcal{R},BP}_{\text{Ag}'}(\text{Ag}, s_0) = \bigcup_{s \in \mathcal{S}^{BP}_{\text{Ag}'}(\text{Ag})} \mathcal{R}^{-1}_{\text{Ag}'}(s).$$

This is the set of all states backward reachable to the states in $\mathcal{S}^{BP}_{\text{Ag}'}(\text{Ag}, s_0)$. If an agent Ag′ occupies any of these states, it will end up in a state where its backup plan will intersect with agent Ag's potential grid points that are defined in $\mathcal{Z}_{\text{Ag}}$. We project this set of states to a set of grid points as

$$\mathcal{G}^{\mathcal{R},BP}_{\text{Ag}'}(\text{Ag}, s_0) = \{\mathcal{G}_{Ag'}(s) \mid s \in \mathcal{S}^{BP}_{Ag'}(Ag, s_0)\}.$$

Note, this set of grid points is shown in the right-most subfigure in Fig. B.1. The bubble is then defined as the union of all the sets of grid points specified above.

Definition 26 (Bubble)**.** *Let us consider an agent Ag with state $s_0 \in S_{Ag}$ and agent Ag′ be another agent. Then the bubble of Ag with respect to agents of the same type as Ag′ is given by*

$$\mathcal{B}_{Ag/Ag'}(s_0) \triangleq \mathcal{Z}_{Ag}(s_0) \cup \mathcal{G}^{\mathcal{R}}_{Ag'}(Ag, s_0) \cup \mathcal{G}^{\mathcal{R},BP}_{Ag'}(Ag, s_0).$$

Note that under almost all circumstances, we should have

$$\mathcal{Z}_{\text{Ag}}(s_0) \subseteq \mathcal{G}^{\mathcal{R}}_{\text{Ag}'}(Ag, s_0) \subseteq \mathcal{G}^{\mathcal{R},BP}_{\text{Ag}'}(\text{Ag}, s_0)$$

so $\mathcal{B}_{\text{Ag}}(s_0)$ is simply equal to $\mathcal{G}^{\mathcal{R},BP}_{\text{Ag}'}(\text{Ag}, s_0)$. This holds true for the abstract dynamics we consider in this paper. This means the bubble contains any grid points in which another agent Ag′ occupying those grid points can interfere (via its own forward reachable states or the backup plan it would use in any of its forward reachable states) with at least one of agent Ag's next possible actions and the backup plan it would use if it were to take any one of those next actions.

B.3 Safety Lemmas

The following lemma states that if all $\text{Ag} \in \mathfrak{A}$ are following the agent protocol, an agent Ag will not take an action that will cause it to 1) collide with or 2) violate the safety backup plan of another agent outside its bubble $\mathcal{B}_{\text{Ag}}(s)$.

Lemma 5. *If Ag is following the agent protocol, and $S_{Ag,bp}(u) = T$, Ag will only choose an action $a \in Act_{Ag}$ for which the following two conditions hold: 1) $\mathcal{G}_{Ag}(s, a) \cap (\cup_{Ag' \in S} \mathcal{G}_{Ag'}(s', a')) = \emptyset$ and 2) $\forall Ag' \in S$, $\neg((Ag, a) \perp Ag')$, where the set $S \triangleq \{Ag' | Ag' \notin \mathcal{B}_{Ag}(s) \wedge ((Ag' \sim Ag) \vee (Ag' \prec Ag) \vee (Ag \prec Ag'))\}$.*

Proof: This follows from the definition of the agent bubble, whose construction is defined in Section B.2.

The following lemma states that an agent Ag following the agent protocol will not take an action for which it violates the safety of its own backup plan.

Lemma 6. *If Ag is following the agent protocol, and $S_{Ag,bp}(u) = \mathtt{T}$, Ag will only choose an action $a \in Act_{Ag}$ for which the following condition holds: $\forall Ag' \in S$, $\neg((Ag, a) \perp Ag')$, where $S = \{Ag\}$.*

Proof: We prove this by using definitions of elements in the agent protocol.

1. Let us first show that any action $a \in Act_{\mathrm{Ag}}$ that Ag takes will satisfy the oracles in the top two tiers (safety and traffic rules) of Ag's profile defined in Section. 3.5.

    a) According to the Action Selection Strategy defined in Section 3.5, Ag will choose one of three actions: the agent's intended action a_i, the best straight action a_{st}, or its backup plan action a_{bp}.

    b) Let us consider the actions a_i and a_{st}.

        i. Both a_i and a_{st} are selected via the behavioral profile and consistent-function evaluator defined in Section 3.5.

        ii. Since $S_{\mathrm{Ag},bp}(u) = \mathtt{T}$, the agent will have at least one action (a_{bp}) for which the top two tiers of specifications are satisfied.

        iii. By definition of the behavioral profile and the consistent evaluator function, if $S_{\mathrm{Ag},bp}(u) = \mathtt{T}$, the safety backup plan action a_{bp} will always be chosen over an action where any of the specifications in the top two tiers of the profile are not satisfied.

        iv. By 1(b)ii and 1(b)iii, Ag will have $a \in Act_{Ag}$ and will choose an action for which the top two tiers of the behavioral profile are satisfied and thus a_i and a_{st} are actions where all oracles in the top two tiers of the profile are satisfied.

    c) Let us consider the action a_{bp}.

        i. This follows from the assumption that $S_{\mathrm{Ag},bp}(u) = \mathtt{T}$ and the definition of $S_{\mathrm{Ag},bp}(u)$.

2. If the oracles in the top two tiers are satisfied by an action a, by the definition of the oracles in Section 3.5, this implies that the action a will take Ag to a state s' and the system will be in a new global state u' where $S_{Ag,bp}(u') = \mathtt{T}$.

3. $S_{Ag,bp}(u') = T$ means Ag will end up in a state where a_{bp} will be an action that satisfies traffic rules, avoids inevitable collision with static obstacles, and thus will not violate its own safety backup plan action $\neg((Ag, a_i)\perp\mathrm{Ag})$.

The following lemma states that if all $\mathrm{Ag} \in \mathfrak{A}$ are following the agent protocol, any agent Ag will not take an action for which it collides with or violates the safety backup plan of any agent with higher precedence.

Lemma 7. *If Ag is following the agent protocol, and $S_{Ag,bp}(u) = \mathtt{T}$, Ag will only choose an action $a \in Act_{Ag}$ for which the following two conditions hold: 1) $\mathcal{G}_{Ag}(s, a) \cap (\cup_{Ag' \in S}\mathcal{G}_{Ag'}(s', a')) = \emptyset$ and 2) $\forall Ag' \in S$, $\neg((Ag, a)\perp Ag')$, where the set $S \triangleq \{Ag' | Ag \prec Ag'\}$, i.e. agents with higher precedence than Ag.*

Proof: We prove this by using arguments based on the definition of precedence, the agent protocol, and agent dynamics.

1. Let us first consider all Ag′ where $\mathrm{Ag} \prec \mathrm{Ag}'$ and $Ag' \notin \mathcal{B}_{\mathrm{Ag}}(s)$.

    a) Proof by Lemma 5.

2. Now, let us consider all Ag′ where $\mathrm{Ag} \prec \mathrm{Ag}'$ and $\mathrm{Ag}' \in \mathcal{B}_{\mathrm{Ag}}(s)$.

3. According to Lemma 6, Ag will only take an action that satisfies all oracles in the top two tiers, including
$O_{\mathrm{Ag,\ dynamic\ safety}}(s, a, u)$.

4. Since a is such that $O_{\mathrm{Ag,\ dynamic\ safety}}(s, a, u) = \mathtt{T}$, by definition of the oracle, Ag will not cause collision with any $Ag' \in \mathcal{B}_{\mathrm{Ag}}(s)$.

5. For any $\mathrm{Ag} \prec \mathrm{Ag}'$, where Ag′ has higher precedence than Ag, then $\mathrm{proj}_{\mathrm{long}}(\mathrm{Ag}) < \mathrm{proj}_{\mathrm{long}}(\mathrm{Ag}')$, i.e. Ag′ is longitudinally ahead of Ag.

6. In order for $(\mathrm{Ag}, a)\perp\mathrm{Ag}'$, the action a would have to be such that $s_f = \tau_{Ag}(s, a)$, and $La(s_f) = La(s')$ and $\mathrm{proj}_{\mathrm{long}}(\mathrm{Ag}) > \mathrm{proj}_{\mathrm{long}}(\mathrm{Ag}')$, where Ag is directly in front of Ag′.

7. Because of the agent dynamics defined in Section 3.3, any a such that $(\text{Ag}, a) \perp \text{Ag}'$ will require $\mathcal{G}(\text{Ag}, a) \cap \mathcal{G}(\text{Ag}') \neq \emptyset$.

8. Thus, any such action a will not satisfy the oracle $O_{\text{Ag, dynamic safety}}(s, a, u)$.

9. Since $S_{\text{Ag},bp}(u) = \texttt{T}$, by Assumption 6 in Section B.4, the agent will have at least one action a_{bp} for which $O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$.

10. Since the agent will only choose an action for which $O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$ and it always has at least one action a_{bp} that satisfies the oracle, the agent will always choose an action for which $O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$ and thus will take an action such that $\forall Ag' \in S \neg((\text{Ag}, a) \perp \text{Ag}')$.

The following lemma states that if all $\text{Ag} \in \mathfrak{A}$ are following the agent protocol, any agent Ag will not take an action for which it collides with or violates the safety backup plan of any agent with lower precedence.

Lemma 8. *If Ag is following the agent protocol, and $S_{Ag,bp}(u) = T$, Ag will only choose an action $a \in Act_{Ag}$ for which the following two conditions hold: 1) $\mathcal{G}_{Ag}(s, a) \cap (\cup_{Ag' \in S} \mathcal{G}_{Ag'}(s', a')) = \emptyset$ and 2) $\forall Ag' \in S$, $\neg((Ag, a) \perp Ag')$, where the set $S \triangleq \{Ag' | Ag' \prec Ag\}$, i.e. agents with lower precedence than Ag.*

Proof: We prove this by using arguments based on the definition of precedence, the agent protocol, and agent dynamics.

1. Let us first consider all Ag' where $\text{Ag} \prec \text{Ag}'$ and $\text{Ag}' \notin \mathcal{B}_{\text{Ag}}(s)$.

    a) Proof by Lemma 5.

2. Now, let us consider all Ag' where $\text{Ag} \prec \text{Ag}'$ and $\text{Ag}' \in \mathcal{B}_{\text{Ag}}(s)$.

3. According to 3, Ag will only take an action that satisfies all oracles in the top two tiers, including $O_{\text{Ag, dynamic safety}}(s, a, u)$.

4. Since a is such that $O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$, by definition of the oracle, Ag will not cause collision with any $Ag' \in \mathcal{B}_{\text{Ag}}(s)$.

5. According to the Action Selection Strategy defined in Section 3.5, Ag will choose one of three actions: the agent's intended action a_i, the best straight action a_{st}, or its backup plan action a_{bp}.

6. Let us consider the backup plan action a_{bp}.

   a) By violation of safety backup plan, $((\text{Ag}, a_{bp}) \perp \text{Ag}')$ only if $La(\text{Ag}) = La(\text{Ag}')$.

   b) W.l.o.g., let us consider Ag′ that is directly behind Ag.

   c) Since $S_{\text{Ag}',bp}(s, u) = \texttt{T}$, by Assumption 6 in Section B.4, $O_{\text{Ag, dynamic safety}}(s, a_{bp}, u) = \texttt{T}$, meaning Ag′ will be far enough behind Ag so that if Ag executes its backup plan action a_{bp}, Ag′ can safely execute its own backup plan action.

   d) Thus, by Definition 19, $\neg((\text{Ag}, a_{bp}) \perp \text{Ag}')$.

7. Let us consider the best straight action a_{st}.

   a) This follows from the arguments made in 6, since a_{st} is a less severe action than a_{bp}.

8. Let us consider the intended action a_i.

   a) Let us consider when $\gamma_{\text{Ag}} = \{\texttt{straight}\}$.

      i. This follows from 6.

   b) Let us consider when $\gamma_{\text{Ag}} \in \{\texttt{right-turn}, \texttt{left-turn}\}$.

      i. If Ag takes such an action, Ag will end up in a state where $Bu(\text{Ag}') \neq Bu(\text{Ag})$ and from Definition 19, agents in different bundles cannot violate each others' backup plans.

   c) Let us consider when $\gamma_{\text{Ag}} \in \{\texttt{right-lane change}, \texttt{left-lane change}\}$.

      i. $(\text{Ag}, a_i) \perp \text{Ag}'$ when a_i is a lane change and the agents Ag and Ag′ are at a state such that $s_f = \tau(s, a_i)$ and $s'_f = \tau(s', a_{bp})$, respectively, where $d(s_f, s'_f) < gap_{req}$, where $d(s_f, s'_f)$ is the l_2 distance between s_f and s'_f.

      ii. When this condition holds, the agent's max-yielding-not-enough flag $\mathcal{F}_{\text{Ag}}(u, a_i)$ defined in Section 13 will be set.

      iii. According to the action-selection strategy, Ag will only take a_i when $\mathcal{F}_{\text{Ag}}(u, a_i) = \texttt{F}$.

iv. Thus, Ag will only take a_i when $\neg((Ag, a_i) \perp Ag')$.

The following lemma states that if all Ag $\in \mathfrak{A}$ are following the agent protocol, any agent Ag will not take an action for which it collides with or violates the safety backup plan of any agent with equal precedence.

Lemma 9. *If Ag is following the agent protocol, and $S_{Ag,bp}(u) = T$, Ag will only choose an action $a \in Act_{Ag}$ for which the following two conditions hold: 1) $\mathcal{G}_{Ag}(s, a) \cap (\cup_{Ag' \in S} \mathcal{G}_{Ag'}(s', a')) = \emptyset$ and 2) $\forall Ag' \in S$, $\neg((Ag, a) \perp Ag')$, where the set $S \triangleq \{Ag' | Ag' \sim Ag\}$, i.e. agents with equivalent precedence as the agent.*

Proof: We prove this by using arguments based on the definition of precedence, Agent Dynamics, and the agent protocol.

1. Let us first consider all Ag′ where Ag $\prec$ Ag′ and Ag′ $\notin \mathcal{B}_{\text{Ag}}(s)$.
    a) Proof by Lemma 5.
2. Now, let us consider all Ag′ where Ag $\prec$ Ag′ and Ag′ $\in \mathcal{B}_{\text{Ag}}(s)$.
3. Let us first consider the agent itself, since an agent has equivalent precedence to itself.
    a) This is true by Lemma 6.
4. This can be proven for any other agents of equivalent precedence that is not the agent itself as follows.
5. Agents with equal precedence take actions simultaneously so $O_{\text{Ag, dynamic safety}}(s, a, u)$ does not guarantee no collision.
6. According to the Action Selection Strategy defined in Section 3.5, Ag will choose one of three actions: the agent's intended action a_i, the best straight action a_{st}, or its backup plan action a_{bp}.
7. By definition of precedence assignment, any Ag′ for which Ag′ $\sim$ Ag will be such that $La(\text{Ag}) \neq La(\text{Ag}')$.
8. Let us show if Ag selects a_{bp}, it will 1) not collide with any Ag′ $\in S$ and 2) $\neg((\text{Ag}, a_{bp}) \perp \text{Ag}')$.

a) W.l.o.g., let us consider Ag′ where Ag′ ∼ Ag.

b) The flag $\mathcal{F}_{\text{Ag'}}(u, a_i) = \texttt{T}$ if Ag′s intended action a_i causes collision with Ag or $(Ag', a_i) \perp Ag$, i.e. it collides with or violates the safety of Ag's backup plan action.

c) By the action-selection-strategy, Ag′ will not take the action a_i when $\mathcal{F}_{\text{Ag'}}(u, a_i) = \texttt{T}$, so this guarantees Ag will not collide with Ag′ when Ag takes a_{bp}.

d) By the Agent Dynamics, Ag's backup plan action cannot cause Ag to end up in a position where it can violate Ag′'s backup plan without colliding with it–for which Ag′'s flag $\mathcal{F}_{\text{Ag}}(u, a_i)$ would be set.

9. Let us show that Ag will only choose an a_{st} if it will 1) not collide with Ag′ ∈ S and 2) $\neg((\text{Ag}, a_{st}) \perp \text{Ag}')$.

a) When $a_{st} = a_{bp}$, then the arguments in 8 hold.

b) Ag selects an a_{st} that is not a_{bp} only when 1) its conflict cluster is empty (i.e. $C_{\text{Ag}} = \emptyset$) or 2) when it has received a conflict request from another agent and it has won its conflict cluster resolution (i.e. $W_{\text{Ag}} = \texttt{T}$).

c) If $C_{\text{Ag}} = \emptyset$, by definition of how conflict clusters are defined in Section 14, the agent's action a_{st} will not cause Ag to collide with any Ag′ ∈ S, and $\forall \text{Ag}' \in S, \neg((\text{Ag}, a_{st}) \perp \text{Ag}')$.

d) In the case Ag has received a conflict request and has won W_{Ag}, by Lemma 3, if $W_{\text{Ag}} = \texttt{T}$, it will be the only agent in its conflict cluster that has won.

e) By definition of the conflict cluster, any Ag′ ∈ C_{Ag} where Ag ∼ Ag′ will take a straight action.

f) Since agents of equivalent precedence are initially in separate lanes by 7 and any Ag′ ∈ S will take a straight action, then $La(s_{\text{Ag},t+1}) \neq La(s_{\text{Ag'},t+1})$ when Ag takes a_{st}.

g) Thus, by definition of agent dynamics and Definition 19, the action will not cause Ag to collide with any Ag′ ∈ S, and $\forall \text{Ag}' \in S, \neg((\text{Ag}, a_{st}) \perp \text{Ag}')$.

10. Let us show that Ag will only choose an a_i if it will 1) not collide with any Ag′ ∈ S and 2) $\neg((\text{Ag}, a_i) \perp \text{Ag}')$.

a) Let us consider when $\gamma_{\text{Ag}} = \texttt{straight}$ for a_i.

i. This follows from the same arguments presented in 9.

b) Let us consider when $\gamma_{\text{Ag}} \in \{\texttt{right-turn}, \texttt{left-turn}\}$ for a_i.

i. This follows from the fact that all other agents are following the agent protocol and will not take a lane-change action in the intersection, and because of the definition of the Agent Dynamics and Road Network.

c) Let us consider when $\gamma_{\text{Ag}} \in \{\texttt{right-lane change}, \texttt{left-lane change}\}$.

i. Ag will only take its intended action a_i if the flag $\mathcal{F}_{\text{Ag}}(u, a_i) = \texttt{F}$, and in the case that it is part of a conflict cluster, it is the winner of the conflict cluster resolution, i.e. $\mathcal{W}_{\text{Ag}} = \texttt{T}$.

ii. By definition of $\mathcal{F}_{\text{Ag}}(u, a_i)$, the agent will not take a_i when a_i causes Ag to collide with any agent $\text{Ag}' \in S$ or when it causes Ag to violate the safety of the back up plan of another agent Ag', i.e. $\exists Ag'$ s.t. $(\text{Ag}, a_i) \perp \text{Ag}'$.

iii. In the case the agent has received a conflict request and has won $\mathcal{W}_{\text{Ag}}$, by Lemma 3, if $\mathcal{W}_{\text{Ag}} = \texttt{T}$, it will be the only agent in its conflict cluster that has won.

iv. By definition of the conflict cluster, any $\text{Ag}' \in C_{\text{Ag}}$ where $\text{Ag} \sim \text{Ag}'$ will take its backup plan action a_{bp}, and thus $s_f = \tau(s, a_{st})$, and $s'_f = \tau(s, a_{bp})$, where $d(s_f, s'_f) \geq gap_{\text{req}}$.

v. Thus, a_i will only be selected when a_i does not cause Ag to collide with any $\text{Ag}' \in S$ and $\forall \text{Ag}' \in S, \neg((\text{Ag}, a_i) \perp \text{Ag}')$.

The following lemma states that if all $\text{Ag} \in \mathfrak{A}$ are following the agent protocol, any agent Ag will not take an action for which it collides with or violates the safety backup plan of any agent with incomparable precedence to it.

Lemma 10. *If Ag is following the agent protocol, and $S_{Ag,bp}(u) = \texttt{T}$, Ag will only choose an action $a \in Act_{Ag}$ for which the following two conditions hold: 1) $\mathcal{G}_{Ag}(s, a) \cap (\cup_{Ag' \in S} \mathcal{G}_{Ag'}(s', a')) = \emptyset$ and 2) $\forall Ag' \in S$, $\neg((Ag, a) \perp Ag')$, where the set $S \triangleq \{Ag' | Ag' \nsucc Ag\}$, i.e. agents with precedence incomparable to the agent.*

Proof: We prove this by using arguments based on the definition of precedence, Agent Dynamics, and the agent protocol.

1. Let us show when Ag chooses a_{bp}, it will 1) not collide with any $\text{Ag}' \in S$ and 2) $\neg((\text{Ag}, a_{bp}) \perp \text{Ag}')$.

    a) Since $S_{\text{Ag},bp}(u) = \texttt{T}$, the agent will have at least one action (a_{bp}) for which the top two tiers of specifications are satisfied.

    b) By 1a, the action a_{bp} will only take Ag into the intersection if the traffic light is green.

    c) By Assumption 4, all traffic lights are coordinated so if agents respect traffic light rules, they will not collide.

    d) By the assumption that all other $\text{Ag}' \in \mathfrak{G}$ are obeying the same protocol, each agent will only take actions that satisfy the top two tiers of their profile.

    e) Any Ag' in a perpendicular bundle will not enter the intersection since they have a red light.

    f) Thus, Ag cannot collide or violate the backup plan of agents in perpendicular bundles.

    g) Any Ag' in an oncoming traffic bundle must only take an unprotected left-turn when it satisfies
    $O_{\text{Ag, unprotected left-turn}}(s, a, u)$.

    h) Thus Ag will not collide or violate the backup plan of agents in bundles of oncoming traffic.

2. Let us show that when Ag chooses a_{st}, it will 1) not collide with any $\text{Ag}' \in S$ and 2) $\neg((\text{Ag}, a_{st}) \perp \text{Ag}')$.

    a) Since a_{st} is chosen according to the behavioral profile, it will only be a straight action that is not a_{bp} as long as it satisfies the top-two tiers of the profile and more.

    b) Thus, a_{st} will only take Ag into intersection if traffic light is green.

    c) By the same arguments in 1, this holds.

3. Let us show that when Ag chooses a_i, it will 1) not collide with any $\text{Ag}' \in S$ and 2) $\neg((\text{Ag}, a_i) \perp \text{Ag}')$.

    a) Let us consider when a_i is such that $\gamma_{Ag} = \texttt{straight}$.

        i. This follows from the same arguments presented in 2.

b) Let us consider when a_i is such that $\gamma_{Ag} \in$ {`left-lane change`,`right-lane change`}.

   i. Ag will never select such an action at an intersection since $O_{\text{Ag, intersection lane-change}}(s, a, u)$ will evaluate to `F`.

c) Let us consider when a_i is such that $\gamma_{Ag} \in$ {`left-turn, right-turn`}.

   i. By the assumption that all other agents are following the agent protocol, all Ag′ that are in bundle perpendicular to $Bu(Ag)$ will not be in the intersection and will not collide with Ag.

   ii. Further, the agent will only take $\gamma_{\text{Ag}} =$ `right-turn` when $O_{\text{Ag, dynamic safety}}(s, a, u) =$ `T` and $O_{\text{Ag, traffic light}}(s, a, u) =$ `T`. Thus, $\neg((\text{Ag}, a_i) \perp \text{Ag}')$.

   iii. For an action a_i where $\gamma_{\text{Ag}} =$ `left-turn`, Ag will only take a_i if $O_{\text{Ag, traffic-light}}(s, a, u) =$ `T` and $O_{\text{Ag, unprotected left-turn}}(s, a, u) =$ `T`.

   iv. Since all agents are following the traffic laws based on Proof B.4, $O_{Ag,\text{traffic light}}(s, a, u) =$ `T` means action will not cause the agent to collide with or violate the safety of the backup plan in perpendicular bundles.

   v. By the definition of the unprotected-left-turn oracle, Ag will only take the left-turn action when it does not violate the safety of the backup plan of agents in oncoming traffic.

B.4 Safety Proof

Theorem 7. *Given all agents $Ag \in \mathfrak{A}$ in the quasi-simultaneous game select actions in accordance to the agent protocol specified in Section 3.5, we can show the safety property $P \Rightarrow \Box Q$, where the assertion P is an assertion that the state of the game is such that $\forall Ag, S_{Ag,bp}(s, u) =$ `T`, i.e. each agent has a backup plan action that is safe, as defined in Definition 18. We denote P_t as the assertion over the state of the game at the beginning of the time-step t, before agents take their respective actions. Q_t is the assertion that the agents never occupy the same grid point in the same time-step t (i.e. collision never occurs when agents take their respective actions during that time-step).*

Proof: To prove an assertion of this form, we need to find an invariant assertion I for which i) $P \Rightarrow I$, ii) $I \Rightarrow \Box I$, and iii) $I \Rightarrow Q$ hold. We define I to be the assertion

that holds on the actions that agents select to take at a time-step. We denote I_t to be the assertion on the actions agents take at time t such that $\forall Ag$, Ag takes $a \in Act_{\text{Ag}}$ where 1) it does not collide with other agents and 2) $\forall$Ag, $S_{\text{Ag},bp}(u') = \texttt{T}$ where $s' = \tau_{\text{Ag}}(s, a)$, and u' is the corresponding global state of the game after Ag has taken its action a.

It suffices to assume:

1. Each Ag $\in \mathfrak{A}$ has access to the traffic light states.
2. There is no communication error in the conflict requests, token count queries, and the agent intention signals.
3. All intersections in the road network $\mathcal{R}$ are governed by traffic lights.
4. The traffic lights are designed to coordinate traffic such that if agents respect the traffic light rules, they will not collide.
5. Agents follow the agent dynamics defined in Section 3.3.
6. For $t = 0$, $\forall$Ag $\in \mathfrak{A}$ in the quasi-simultaneous game is initialized to:
   - Be located on a distinct grid point on the road network.
   - Have a safe backup plan action a_{bp} such that $S_{\text{Ag},bp}(s, u) = \texttt{T}$.

We can prove $P \Rightarrow \Box Q$ by showing the following:

1. $P_t \Rightarrow I_t$. This is equivalent to showing that if all agents are in a state where P is satisfied at time t, then all agents will take actions at time t where the I holds.
   a) In the case that the assertion P_t holds, let us show that Ag will only choose an action $a \in Act_{Ag}$ for which the following two conditions hold: 1) $\mathcal{G}_{Ag}(s, a) \cap (\cup_{Ag' \in S} \mathcal{G}_{\text{Ag}'}(s', a')) = \emptyset$ and 2) $\forall Ag' \in S$, $\neg((\text{Ag}, a) \perp \text{Ag}')$, where the set S is:
      i. The set $S \triangleq \{\text{Ag}' | \text{Ag} \prec \text{Ag}'\}$, i.e. agents with higher precedence than Ag. Proof by Lemma 7.
      ii. $S \triangleq \{\text{Ag}' | \text{Ag}' \prec \text{Ag}\}$, i.e. agents with lower precedence than Ag. Proof by Lemma 8.

iii. $S \triangleq \{\text{Ag}' | \text{Ag}' \sim \text{Ag}\}$, i.e. agents with equal precedence than the agent. Proof by Lemma 9.

iv. $S \triangleq \{\text{Ag}' | \text{Ag}' \nsim \text{Ag}\}$, i.e. agents with precedence incomparable to the agent. Proof by Lemma 10.

b) The set of all agents, agents with lower precedence, higher precedence, equal precedence, and incomparable precedence, is complete and includes all agents.

c) By 1-1(a)iv and 1b, an agent will not take an action that will cause collision with any other agents (including itself) or violate the safety of the safety backup plan of all other agents, and thus any action taken by any agent will be such that following the action, the assertion P still holds.

2. $I \Rightarrow \Box I$. If agents take actions at time t such that the assertion I_t holds, then by the definition of the assertion I, agents will end up in a state where at time t+1, assertion P holds, meaning $I_t \Rightarrow P_{t+1}$. Since $P_{t+1} \Rightarrow I_{t+1}$, from 1, we get $I \Rightarrow \Box I$.

3. $I \Rightarrow Q$. This is equivalent to showing that if all agents take actions according to the assertions in I, then collisions will not occur. This follows from the invariant assertion that agents are taking actions that do not cause collision, and the fact that all Ag have a safe backup plan action a_{bp} to choose from, and thus will always be able to (and will) take an action from which it can avoid collision in future time steps.

B.5 Liveness Lemmas

Lemma 11. *If the only $a \in Act_{Ag}$ for an agent Ag for which $O_{Ag,\text{ destination reachability}}(s,a,u) = \texttt{T}$ and $O_{Ag,\text{ forward progress}}(s,a,u) = \texttt{T}$ is an action such that: $\gamma_{Ag} \in \{\texttt{right-turn, left-turn}\}$ and the grid-point $s_f = \tau_{Ag}(s,a)$ is unoccupied (for a left-turn, where a is the final action of the left-turn maneuver), Ag will always eventually take a.*

Proof: W.l.o.g., let us consider agent $\text{Ag} \in \mathfrak{A}$ in the quasi-simultaneous game $\mathfrak{G}$. We prove this by showing that all criteria required by the agent protocol are always eventually satisfied, thereby allowing Ag to take action a.

1. By the definition of $\mathfrak{R}$ and the agent dynamics, when Ag is in a position where only $\gamma_{\text{Ag}} \in \{\texttt{right-turn}, \texttt{left-turn}\}$, it will neither send nor receive requests from other agents and $\mathcal{F}_{\text{Ag}}(u, a_i)$ will never be set to $\texttt{T}$.

2. In accordance with the Action Selection Strategy, for Ag to take action a, all the oracles in the behavioral profile must be simultaneously satisfied (so it will be selected over any other $a' \in Act_{\text{Ag}}$). Thus, we show:

    a) The following oracle evaluations will always hold when Ag is in this state:
    $O_{\text{Ag, traffic intersection lane-change}}(s, a, u) = \texttt{T}$
    $O_{\text{Ag, legal orientation}}(s, a, u) = \texttt{T}$, $O_{\text{Ag, static safety}}(s, a, u) = \texttt{T}$ and
    $O_{\text{Ag, traffic intersection clearance}}(s, a, u) = \texttt{T}$.

        i. The first oracle is true vacuously and the following are true by the road network constraints and agent dynamics, assumptions 6, and the Assumption in the lemma statement that $s_f = \tau(s, a)$ is unoccupied respectively.

    b) To show that the following oracles will always eventually simultaneously hold true, let us first consider when $\gamma_{\text{Ag}} = \{\texttt{right-turn}\}$.

        i. By the assumption, the traffic light is red for a finite time, and when the traffic light is green, $O_{\text{Ag, traffic light}}(s, a, u) = \texttt{T}$.

        ii. $O_{\text{unprotected left-turn}}(s, a, u)$ is vacuously true for a right-turn action.

        iii. Since $O_{\text{Ag, traffic intersection clearance}}(s, a, u) = \texttt{T}$ and by the safety proof B.4, all Ag are only taking actions in accordance with traffic laws so there will never be any $\text{Ag}' \in \mathfrak{A}$ blocking the intersection, making $O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$.

        iv. Thus, all oracles are always eventually simultaneously satisfied and Ag can take a where $\gamma_{\text{Ag}} = \{\texttt{right-turn}\}$

    c) Let us consider when $\gamma_{\text{Ag}} = \{\texttt{left-turn}\}$.

        i. By Assumption 5, traffic lights are green for a finite time.

        ii. By the safety proof B.4, all Ag are only taking actions in accordance with traffic laws so there will never be any $\text{Ag}' \in \mathfrak{A}$ blocking the intersection.

        iii. When $\gamma_{\text{Ag}} = \texttt{left-turn}$, by definition of the unprotected left-turn oracle, $\Box\Diamond O_{\text{Ag, unprotected left-turn}}(s, a, u)$, specifically when the traffic

light switches from green to red and Ag has been waiting at the traffic light.

iv. Thus, $\square\lozenge O_{\text{Ag, unprotected left-turn}}(s, a, u)$ after the light turns from green to red.

v. Further, $O_{\text{Ag, unprotected left-turn}}(s, a, u) = \texttt{T}$ combined with $O_{\text{Ag, traffic intersection clearance}}(s, a, u) = \texttt{T}$ implies $O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$.

vi. Thus, all oracles are always eventually simultaneously satisfied and Ag can take a where $\gamma = \{\texttt{left-turn}\}$.

3. Thus, we have shown all oracles in the behavioral profile will always eventually be satisfied, and Ag will take a such that $O_{\text{Ag, destination reachability}}(s, a, u) = \texttt{T}$ and $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

Lemma 12. *If the only $a \in Act_{Ag}$ for which $O_{Ag,\ destination\ reachability}(s, a, u) = T$ and $O_{Ag,\ forward\ progress}(s, a, u) = T$ is when a has $\gamma_{Ag} \in \{$`right-lane change`, `left-lane change`$\}$ and the grid-point(s) $\mathcal{G}(s, a)$ is (are) either unoccupied or agents that occupy these grid points will always eventually clear these grid points, Ag will always eventually take this action a.*

Proof: W.l.o.g., let us consider agent Ag $\in \mathfrak{A}$ in the quasi-simultaneous game $\mathfrak{G}$. We prove this by showing that all criteria required by the agent protocol are always eventually satisfied, thereby allowing Ag to take its action a.

1. Let us consider Case A, when a is such that $s_f = \tau_{\text{Ag}}(s, a) = \texttt{Goal}_{\text{Ag}}$, i.e. the action takes the agent to its goal, and let us show that Ag will always eventually be able to take a.

2. In accordance with the Action Selection Strategy, for Ag to take a is that 1) all the oracles in the behavioral profile must be simultaneously satisfied (so the action a is chosen over any other $a' \in Act_{\text{Ag}}$, 2) $\mathcal{F}_{\text{Ag}}(u, a_i) = 0$, and 3) $W_{\text{Ag}} = \texttt{T}$.

3. We first show all the oracles for Ag will always be simultaneously satisfied:

   a) When Ag is in this state, the following oracle evaluations always hold: $O_{\text{Ag, traffic light}}(s, a, u) = \texttt{T}$, $O_{\text{Ag, traffic intersection lane-change}}(s, a, u) = \texttt{T}$,

$O_{\text{Ag, unprotected left turn}}(s, a, u) = \texttt{T}$,
$O_{\text{Ag, traffic intersection clearance}}(s, a, u)$, $O_{\text{Ag, static safety}}(s, a, u) = \texttt{T}$,
$O_{\text{Ag, traffic orientation}}(s, a, u) = \texttt{T}$.

i. The first four hold vacuously, the others hold by Assumption 6, and the last holds by agent dynamics and the Road Network.

b) $O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$.

i. By the definition Road Network $\mathfrak{R}$, agent dynamics in Section 3.3, and the condition that $\forall \text{Ag} \in \mathfrak{A}$ will leave $\mathfrak{R}$ (i.e. Ag does not occupy any grid point on $\mathfrak{R}$ when it reaches its respective goal $\texttt{Goal}_{\text{Ag}}$). Thus, $O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$ whenever an agent is in this state.

4. In accordance with the action selection strategy, for Ag to take a, it must be that $\mathcal{F}_{\text{Ag}}(u, a_i) = 0$, i.e. the max-yielding-flag-not-enough must not be set. Let us show that this is always true.

a) The only Ag′ that can cause the $\mathcal{F}_{\text{Ag}}(u, a_i) = 1$ of Ag is when an agent Ag′ is in a state where $La(Ag') = \texttt{Goal}_{\text{Ag}}$.

b) W.l.o.g. let us consider such an Ag′. By liveness Assumption 7, upon approaching the goal, the agent Ag′ must be in a state where Ag′ backup plan action a_{bp} will allow it to come to a complete stop before reaching its goal.

c) By 4b, Ag′ will always be in a state for which the max-yielding-not-enough flag for Ag is $\mathcal{F}_{\text{Ag}}(u, a_i) = 0$.

5. In order for Ag to take a, it must be that $W_{Ag} = 1$. Let us show that this is always eventually true.

a) In the case that Ag has the maximum number of tokens, $\mathcal{W}_{Ag} = 1$ and Ag will be able to take its forward action since all criteria are satisfied.

b) Any $\text{Ag}' \in C_{Ag}$ will be of equal or lower precedence than Ag.

c) Any Ag′ with the maximum number of tokens will move to its goal since $\mathcal{W}_{\text{Ag}} = 1$ and all the other criteria required for that agent to take its action will be true.

d) By definition of the Action Selection Strategy in Section 3.5, any agent $\hat{\text{Ag}}$ that replaces Ag′ will have taken a forward progress action and its respective token count will reset to 0.

e) Thus, any Ag′ will be allowed to take its action before Ag, but Ag's token count $\texttt{Tc}_{\text{Ag}}$ will increase by one for every time-step this occurs.

f) Thus, by 5d and by 5e, Ag will always eventually have the highest token count in its conflict cluster such that $W_{\text{Ag}} = 1$.

g) Since conditions 3 and 4 are always true, and 5 is always eventually true, then all conditions will simultaneously always eventually be true and the Ag will always eventually take the action a.

6. Let us consider Case B, when a is the final action to take for an agent to reach its sub-goal (i.e. a critical left-turn or right-turn tile), and let us show Ag will always eventually be able to take a forward progress action where $\gamma_{\text{Ag}} \in \{\texttt{left-lane change}, \texttt{right-lane change}\}$.

7. In accordance with the Action Selection Strategy, for Ag to take a is that 1) $W_{\text{Ag}} = 1$, 2) $\mathcal{F}_{\text{Ag}}(u, a_i) = 0$, i.e. the max-yielding-flag-not-enough must not be set and 3) all the oracles in the behavioral profile must be simultaneously satisfied.

8. Let us first consider when $W_{\text{Ag}} = 1$, then $\Box W_{\text{Ag}}$ until Ag takes its forward progress action a because by definition of W_{Ag}, Ag has the highest token count in its conflict cluster, $\text{Ag.}\texttt{tc} = \text{Ag.}\texttt{tc} + 1$, while Ag does not select a (and thus does not make forward progress) and any Ag that newly enters Ag's conflict cluster will have a token count of 0.

9. All the oracles are either vacuously or trivially satisfied by the assumptions except for $O_{\text{Ag, dynamic safety}}(s, a, u)$.

10. By the lemma assumption that all Ag' occupying grid points will always eventually take their respective forward progress actions, $\Box\Diamond O_{\text{Ag, dynamic safety}}(s, a, u)$.

11. By the Assumption 5, the traffic light will always cycle through red-to-green and green-to-red at the intersection Ag is located at.

12. By the Assumption on the minimum duration of the red traffic light, all Ag′ will be in a state such that $\mathcal{F}_{\text{Ag}}(u, a_i) = 0$.

13. Thus, all criteria for which Ag can take its forward progress action a will be simultaneously satisfied.

14. When $W_{\text{Ag}} = 0$, we must show $\Box\Diamond W_{\text{Ag}}$.

a) For Ag, all agents in its conflict cluster have equal or lower precedence and are not in the same lane as Ag.

b) For any such Ag′ with equal precedence, Ag′ will always eventually take its forward progress action by the arguments in 8-14 if Ag′ intends to make a lane-change.

c) By the lemma assumption, any agents Ag' occupying the grid points that Ag needs to take its action will always eventually take its forward progress action so
$\Box\Diamond O_{\text{Ag, dynamic safety}}(s, a, u)$.

d) Any $\hat{\text{Ag}}$ with lower precedence and higher token count that Ag will take Ag′'s position and in doing so will have a token count of 0 and any Ag that replaces any agents with higher token count than Ag and is in Ag's conflict cluster will have token count 0.

e) Thus $\Box\Diamond W_{\text{Ag}}$.

Lemma 13. *Let us consider a road segment $rs \in RS$ where there exist grid points $g \in \mathcal{S}_{sinks}$. Every $Ag \in rs$ will always eventually be able to take $a \in Act_{Ag}$ for which $O_{Ag,\, forward\ progress}(s, a, u) = \top$.*

Proof: We prove this by induction. W.l.o.g, let us consider $\text{Ag} \in \mathfrak{A}$. Let $m_{\text{Ag}} = \text{proj}_{\text{long}}(\texttt{Goal}_{\text{Ag}}) - \text{proj}_{\text{long}}(Ag.s)$.

1. Base Case: $m_{\text{Ag}} = 1$, i.e. Ag only requires a single action a to reach its goal $\texttt{Goal}_{\text{Ag}}$.

   a) If a is such that
   $\gamma_{Ag} \in \{\texttt{left-lane change}, \texttt{right-lane change}\}$, then Ag will take always eventually this action by Lemma 12.

   b) If a is such that $\gamma_{\text{Ag}} = \texttt{straight}$:

   c) In accordance with the Action Selection Strategy, for Ag to take a is that 1) all the oracles in the behavioral profile must be simultaneously satisfied (so the action a is chosen over any other $a' \in Act_{\text{Ag}}$, and 2) $W_{\text{Ag}} = 1$.

   d) First, we show that all oracles in the behavioral profile will always be simultaneously satisfied.

i. These all follow from the same arguments presented when $\gamma_{Ag} = \{\texttt{right-lane change}, \texttt{left-lane change}\}$ in Case A in Lemma 12.

e) In accordance with the Action Selection Strategy, we must show that $\Box\Diamond W_{\text{Ag}}$. This is vacuously true since no Ag will be in the agent's conflict cluster when an agent is in this state.

2. Case $m = N$: Let us assume that any $\forall$Ag where $m_{\text{Ag}} = N$ always eventually take $a \in Act_{\text{Ag}}$ for which
$O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

3. Case $m = N + 1$: Let us show $\forall$Ag where $m_{\text{Ag}} = N + 1$ always eventually take a for which
$O_{\text{Ag, forward progress}} = \texttt{T}$.

a) Any Ag for which $m_{\text{Ag}} > 1$ will always have an a where $\gamma_{\text{Ag}} = \texttt{straight}$ such that $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

b) Thus, we show that Ag always eventually will take $\gamma_{\text{Ag}} = \texttt{straight}$ such that $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

c) W.l.o.g., let us consider Ag for which $m_{\text{Ag}} = N + 1$.

d) In accordance with the Action Selection Strategy, for Ag to take a is 1) $W_{\text{Ag}} = 1$ and 2) all the oracles in the behavioral profile must be simultaneously satisfied (so the action a is chosen over any other $a' \in Act_{\text{Ag}}$).

e) In accordance with the Action Selection Strategy, we must show $\Box\Diamond W_{\text{Ag}}$.

i. Any $\text{Ag}' \in C_{\text{Ag}}$ will be an agent of equal or higher precedence and in a separate lane.

ii. Any such agent with higher token count than Ag that is in its conflict cluster will always eventually be able to go by the inductive assumption in 2.

iii. After all such agents take a forward progress action, they will no longer be in Ag's conflict cluster and Ag will have the highest token count since all Ag that newly enter the conflict cluster will have token count of 0.

f) After the assignment $W_{\text{Ag}} = 1$, $\square W_{\text{Ag}}$ until Ag selects a. This is true because by definition of W_{Ag}, Ag has the highest token count in its conflict cluster, $\text{Ag.}\texttt{tc} = Ag.\texttt{tc} + 1$, while Ag does not select a, and any Ag that enters Ag's conflict cluster will have a token count of 0.

g) Let us show that the oracles in the behavioral profile will always evaluate to T.

i. The same arguments hold here as in Lemma 12.1 for all oracles except for $O_{\text{Ag, dynamic safety}}(s, a, u)$, where $\square\Diamond O_{\text{Ag, dynamic safety}}(s, a, u) = \texttt{T}$ by the inductive Assumption 2.

Lemma 14. *Let Ag be on a road segment $rs \in RS$, where RS is the set of nodes in the dependency road network dependency graph $\mathcal{G}_{dep}$. Let rs be a road segment for which $\forall rs' \in RSs.t.\exists e : (rs', rs)$. Each road segment rs' has vacancies in the grid points where Ag $\in rs$ would occupy if it crossed the intersection (i.e. $s_f = \tau_{Ag}(s, a)$), and we show that Ag will always eventually take an action $a \in Act_{Ag}$ where $O_{Ag,\ progress\ oracle}(s, a, u) = \texttt{T}$.*

Proof: We prove this with induction. W.l.o.g., let us consider $\text{Ag} \in \mathfrak{A}$. Let $m_{\text{Ag}} = \text{proj}_{\text{long}}(g_{\text{front of rs}}) - \text{proj}_{\text{long}}(\text{Ag}.s)$, where $g_{\text{front of intersection}}$ represents a grid point at the front of the road segment.

1. Base Case $m_{\text{Ag}} = 0$: Let us consider an Ag whose next action will take will bring Ag to cross into the intersection and show that Ag will always eventually take a for which $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

   a) If the only a where $O_{\text{Ag, forward progress}} = \texttt{T}$ is such that $\gamma_{\text{Ag}} \in \{\texttt{left-turn}, \texttt{right-turn}\}$, proof by Lemma 11.

   b) If the only a where $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$ is such that $\gamma_{\text{Ag}} = \texttt{straight}$.

      i. In accordance with the Action Selection Strategy, for Ag to take a is that 1) all the oracles in the behavioral profile must be simultaneously satisfied (so the action a is chosen over any other $a' \in Act_{\text{Ag}}$, 2) $W_{\text{Ag}} = 1$.

         A. $O_{\text{Ag, unprotected left-turn}}(s, a, u) = \texttt{T}$,
         $O_{\text{Ag, traffic intersection lane-change}}(s, a, u) = \texttt{T}$,
         $O_{\text{Ag, static safety}}(s, a, u) = \texttt{T}$,

$O_{\text{Ag, traffic intersection clearance}}(s, a, u) = \texttt{T}$
$O_{\text{Ag, legal orientation}}(s, a, u) = \texttt{T}$.

B. The first two oracles are true vacuously, followed by Assumption 6, and by agent dynamics and the Road Network $\mathfrak{R}$ definition, respectively, and by the assumption in the lemma statement.

C. $\square\lozenge O_{\text{Ag, traffic light}}(s, a, u)$ by Assumption 5.

D. $O_{\text{Ag, dynamic obstacle}}(s, a, u) = \texttt{T}$ because by the safety proof, all Ag take $a \in Act_{\text{Ag}}$ that satisfy the first top tiers of the behavioral profile so there will be no $\text{Ag}' \in \mathfrak{A}$ that are in the intersection when the traffic light for Ag is green. Thus, whenever $O_{\text{Ag, traffic light}}(s, a, u) = \texttt{T}$, then it $O_{\text{Ag, dynamic obstacle}}(s, a, u) = \texttt{T}$ as well.

ii. $W_{Ag} = 1$ vacuously since neither Ag or any $\text{Ag}' \in \mathfrak{A}$ will send a conflict request at the front of the intersection since all a_i must satisfy $O_{\text{Ag, traffic intersection lane-change}}(s, a, u)$ according to the Safety Proof in Section AB.4.

c) By the safety proof in B.4, Ag will only take $a \in Act_{\text{Ag}}$ that satisfy the top two tiers of the behavioral profile, so Ag will not take an a where $\gamma_{\text{Ag}} \in \{\texttt{left-lane change}, \texttt{right-lane change}\}$ into an intersection.

2. Case $m_{Ag} = N$: Let us assume that Ag with $m_{\text{Ag}} = N$ will always eventually take $a \in Act_{\text{Ag}}$ for which
$O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

3. Case $m_{Ag} = N + 1$: Let us show that any Ag that is at a longitudinal distance of $N + 1$ from the destination will always eventually take a for which $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

a) Let us consider when Ag's only a such that
$O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$ is
$\gamma_{\text{Ag}} \in \{\texttt{right-lane change}, \texttt{left-lane change}\}$.

b) Although Ag may not have priority (since it does not have max tokens in its conflict cluster), any Ag that occupies grid points $\mathcal{G}(s, a, u)$ will always eventually make forward progress by Argument 1.

c) Once these agents have made forward progress, any $\hat{Ag}$ that replace Ag' will have a $\texttt{Tc}_{Ag} = 0$ and since Ag is always increasing its token counts

as it cannot make forward progress, it will always eventually have the max tokens and thus have priority over those grid points.

d) Thus, this can be proven by using Case B in Lemma 12.

e) For all other $a \in Act_{\text{Ag}}$ are actions for which $\gamma_{\text{Ag}} = \texttt{straight}$, and the same arguments as in the proof of straight actions for rs with $g \in \mathcal{S}_{\text{sinks}}$ in 3 hold.

B.6 Liveness Proof

Theorem 8 (Liveness Under Sparse Traffic Conditions)**.** *Under the Sparse Traffic Assumption given by 21 and given all agents Ag* $\in \mathfrak{A}$ *in the quasi-simultaneous game select actions in accordance with the agent protocol specified in Section 3.5, liveness is guaranteed, i.e. all Ag* $\in \mathfrak{A}$ *will always eventually reach their respective goals.*

Proof: It suffices to assume:

1. $\forall \text{Ag} \in \mathfrak{A}$, $\forall \text{Ag}' \in \mathbb{B}_{\text{Ag}}$, Ag knows $\text{Ag}'.s, Ag'.i$, i.e. the other agent's state Ag.s and intended action a_i and all Ag within a region around the intersection defined in Appendix B.

2. Each Ag $\in \mathfrak{A}$ has access to the traffic light states.

3. There is no communication error in the conflict requests, token count queries, and the agent intention signals.

4. For $t = 0$, $\forall \text{Ag} \in \mathfrak{A}$ in the quasi-simultaneous game is initialized to:

   - Be located on a distinct grid point on the road network.
   - Have a safe backup plan action a_{bp} such that $S_{\text{Ag},bp}(u) = \texttt{T}$.

5. The traffic lights are red for some time window Δt_{tl} such that $t_{\min} < \Delta t_{\text{tl}} < \infty$, where $t_{\min}$ is defined in Appendix B in Section B.7.

6. The static obstacles are not on any grid point g where $g.d = 1$.

7. Each Ag treats its respective goal Ag.g as a static obstacle.

8. Bundles in the road network $\mathfrak{R}$ have no more than 2 lanes.

9. The road network R is such that all intersections are governed by traffic lights.

and prove:

1. The invariance of a no-deadlock state follows from the sparsity assumption and the invariance of safety (no collision) follows from the safety proof.

2. For any $\mathfrak{R}$ where the dependency graph G_{dep} (as defined in 11) is a directed-acylcic-graph (DAG), we prove all Ag $\in \mathfrak{A}$ will always eventually take $a \in Act_{Ag}$ for which
   $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$ inductively as follows.

   a) A topological sorting of a directed acyclic graph G = (V, E) is a linear ordering of vertices V such that $(u, v) \in E \rightarrow u$ appears before v in ordering.

   b) If and only if a graph G is a DAG, then G has a topological sorting. Since G_{dep} is a DAG, it has a topological sorting.

   c) We can then use an argument by induction on the linear ordering provided by the topological sorting to show that all Ag always eventually take $a \in Act_{Ag}$ for which $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

      i. Let l denote the linear order associated with the road network dependency graph G_{dep}, where an ordering of $l = 0$ denotes a road segment with source nodes.

      ii. Base Case $l = 0$. This can be proven true by Lemma 13.

      iii. Let us assume this is true for any road segment where $l = N$.

      iv. Under the Inductive Assumption 2(c)iii, there will be clearance in any road segment that agent Ag depends on for Ag to make forward progress to its destination.

      v. Since all Ag are following the traffic laws by the Safety proof in B.4, the clearance spots will be given precedence to Ag $\in rs$ for a positive, finite time, and thus the assumptions required in Lemma 11 and 12 used to prove Lemma 14 will hold.

      vi. Thus, the Lemma 14 can be used to show that all Ag for which $l = N + 1$ always eventually take an action for which
      $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

3. When the graph G_{dep} is cyclic, the Sparsity Assumption (in Defintion 21) can be used to prove all agents always eventually take an action for which $O_{\text{Ag, forward progress}}(s, a, u) = \texttt{T}$.

    a) The sparsity assumption (in Definition 21) ensures that there is at least one vacancy in any map loop.

    b) Let us consider Ag inside a map loop.

        i. Let us consider Ag in the loop for which the vacancy is directly ahead of Ag. If the vacancy is directly ahead of Ag, then if the only forward progress action a keeps Ag in the loop, Ag will always eventually take its action by Lemmas 11, 12 and the arguments in Lemma 14 1b. If the only forward progress action a makes Ag leave the loop, Ag will always eventually take its action by the sparsity assumption (in Definition 21) and the inductive arguments in the Liveness proof argument 2c.

        ii. By 3(b)i, it can then be inductively shown that any Ag in the loop will always eventually have a vacancy for which it can take a forward progress action.

    c) Let us consider Ag on a road segment that is not part of a map loop.

        i. Let us consider an action a that takes Ag into a map loop. If the grid point required by Ag to make forward progress is occupied, by 3(b)ii, it will always eventually be unoccupied. If the only action Ag can take is such that $\gamma_{Ag} = \{\texttt{lane-change}\}$ since all Ag' in the loop are reset when they take forward progress action, Ag will always eventually have the max token count. Thus, the same arguments in Lemma 12 hold. If the only action Ag can take is such that Ag crosses into an intersection, the traffic light rules ensure that Ag has precedence over any Ag in the loop. Thus, Ag will always eventually take a forward progress action by Lemma 11 and Lemma 14 1b.

        ii. For any action a that does not take Ag into a map loop, Ag can take a forward action because of the sparsity assumptions 21 and the inductive arguments in the Liveness proof argument 2c.

4. By the induction arguments and by definition of the forward progress oracle $O_{\text{Ag, forward progress}}(s, a, u)$, all Ag will always eventually take actions that

allow them to make progress to their respective destinations, and liveness is guaranteed.

B.7 Traffic Light Assumptions

A traffic light grid point contains three states $g.s = \{\texttt{red}, \texttt{yellow}, \texttt{green}\}$. The traffic lights at each intersection are coordinated so that if all agents obey the traffic signals, collision will not occur (i.e. the lights for the same intersection will never be simultaneously green) and the lights are both red for long enough such that Ag that entered the intersection when the light was `yellow` will be able to make it across the intersection before the other traffic light turns `green`.

Traffic Light Minimum Time

In order to guarantee that agents will always eventually be able to make a lane-change to a critical tile, the traffic light has to be red for sufficiently long such that any Ag' that may cause $\mathcal{F}_{\text{Ag}}(u, a_i) = \texttt{T}$ is slowed down for long enough such that Ag can take its lane-change action. This can be computed simply once given the dynamics of Ag. Normally a simple heuristic can be used instead of computing this specific lower-bound.

Appendix C

PERCEPTION: SEMANTIC ESTIMATION

C.1 Discrete Likelihood Function

The discrete likelihood function in Section 4.2 is a nonlinear discrete function. There are two separate likelihood functions depending on whether the semantic measurement $z^k_{s,t}$ is 1 or 0. We want to give a smooth approximation for the negative log of the discrete likelihood function. For this paper, we approximate the likelihood function with a bump function parameterized by the probabilities a and b with the following form:

$$f(r, a, b) = \sqrt{(k \exp(-\frac{1}{r_0^2 - r^2}) - \log(b))}, \tag{C.1}$$

where $k = \frac{\log(b)-\log(a)}{\exp(-\frac{1}{r_0^2})}$ and $r = \sqrt{(x_1^2 + x_2^2)}$, where x_1 and x_2 are the first and second coordinates of the state x_t. For the bump function corresponding to $z^k_{s,t} = 1$, the bump function parameters are set so $a = C^k_{11}$ and $b = C^k_{10}$ and for the measurement $z^k_{s,t} = 0$, the parameters are set so $a = C^k_{01}$ and $b = C^k_{00}$.

The nonlinear factor in a factor graph becomes the term $f(r, a, b)R^{-1}f(r, a, b)$, where R is the covariance associated with the measurement. This is why the square root is necessary in order to preserve the Bayesian representation of the likelihood function, and the covariance matrix associated with this factor is chosen to be the identity matrix.

C.2 Forward-backward Algorithm

The modified forward-backward algorithm that can be used to accommodate the time-varying transition and observation matrices can be written as follows:

$$P(s_\tau = o_i, Y_{0:\tau,f}) = \frac{\alpha_{o_i}(\tau)\beta_{o_i}(\tau)}{\sum_{o'_i} \alpha_{o'_i}(\tau)\beta_{o_i}(\tau)}, \tag{C.2}$$

where the values $\alpha_{o_i}(\tau)$ and $\beta_{o_i}(\tau)$ are recursively defined below:

$$\alpha_{o_i}(\tau) = p_\tau(y_\tau|o_i) \sum_{o'_i} \alpha_{o'_i}(\tau - 1)p_{\tau-1,\tau}(o_i|o'_i) \tag{C.3}$$

and

$$\beta_{o_i}(\tau) = \sum_{o_i'} \beta_{o_i'}(\tau+1) p_{\tau-1,\tau}(o_i'|o_i) p_{\tau+1}(y_{\tau+1}|o_i'). \tag{C.4}$$

The dependencies of the transition and observation matrices on the object detection events are denoted by the subscripts of the probabilities in the above equation.