Rules of the Road: Towards Safety and Liveness Guarantees for Autonomous Vehicles

Karena X. Cai, Tung Phan-Minh, Soon-Jo Chung, Richard M. Murray

***Abstract*— The ability to guarantee safety and progress for all vehicles is vital to the success of the autonomous vehicle industry. We present a framework for the distributed control of autonomous vehicles that is safe and guarantees progress for all agents. In this paper, we first introduce a new game paradigm which we term the quasi-simultaneous discrete-time game. We then define an Agent Protocol agents must use to make decisions in this quasi-simultaneous discrete-time game setting. According to the protocol, agents first select an intended action and then each agent determines whether it can take its intended action or not, given its proposed intention and the intentions of nearby agents. The protocol so defined will ensure safety under all traffic conditions and liveness for all agents under 'sparse' traffic conditions. These guarantees, however, are predicated on the premise that *all* agents are operating with the aforementioned protocol. We provide proofs of correctness of the protocol and validate our results in simulation.**

I. INTRODUCTION

A prerequisite for introducing autonomous vehicles into our society is a compelling proof of their safety and efficacy. The current prevailing methodology used for proving safety of these vehicles is simulating and test-driving these vehicles for millions of miles, which is a practice that lacks both formal verifiability and scalability.

Formal methods offers tools for designing provably correct control strategies for complex systems like autonomous vehicles that satisfy high-level behavioral specifications like safety and liveness [2] for each individual vehicle. The algorithms used for synthesizing formally-correct strategies for the vehicles, however, cannot guarantee global safety since they do not make the assumptions that must hold on other vehicle behaviors explicit [26], [5], [24].

Instead of reasoning about safety on the individual agent level, Shoham and Tennenholtz introduce the idea of reasoning about safety as a property of the collective of agents [22]. In particular, they introduce the idea of social laws, which are a set of rules imposed upon all agents in a multiagent system to ensure some desirable global behaviors like safety or progress [22], [25]. The design of social laws is intended to achieve the desirable global behavioral properties in a minimally-restrictive way [22]. The problem of automatically synthesizing useful social laws for a set of agents for a general state space, however, has been shown to be NP-complete [22].

This research supported by the National Science Foundation award CNS-1545126.

Karena X. Cai is a graduate student in Control and Dynamical Systems at the California Institute of Technology kcai at caltech dot edu

Tung Phan-Minh is a graduate student in Mechanical Engineering at the California Institute of Technology tung at caltech dot edu

The Responsibility-Sensitive-Safety (RSS) framework [21] adopts a similar top-down philosophy for guaranteeing safety by providing a set of rules, which if followed by all agents, guarantees no collisions. In the case of an accident, blame can be formally assigned. This framework, however is incomplete since it does not ensure the progress of the vehicles towards their destinations.

The Assume-Guarantee framework for autonomous vehicles introduced in [16] similarly dictates all agents must abide by some behavioral contract according to an ordered set of rules in order to guarantee transparency into the reasoning behind agent decisions. The framework is incomplete, though, as it does not provide safety and liveness guarantees.

The problem of fully guaranteeing safety and liveness of decision-making agents is especially challenging since 1) agents are often competing for the same set of resources (some region of the road network) and 2) agents must reason about highly-coupled and complex interactions with other agents. Historically, interactive partially observable Markov Decision Processes (I-POMDPs) have been proposed to model these complex interactions—but these methods lack scalability since they are computationally expensive [10], [15]. More recently, data-based approaches have been proposed to simplify the agent reasoning process but these approaches require large amounts of data and fail to provide any safety or liveness guarantees [19], [8], [7], [18]. Token-based conflict resolution approaches for pairwise interactions between autonomous agents have been studied in [4] as well.

The process for resolving multiple conflicting processes in a local, decentralized manner is addressed in the Drinking Philosopher problem, which provides a mechanism for resolving issues arising from synchronous decision-making [6]. The solution to the Drinking Philosopher problem is an algorithm that assigns precedence among a set of agents that have conflicting goals. The algorithm preserves acyclicity and fairness in the precedence graph, thereby ensuring consistency and fairness among all agents in the game.

Our work is an adaptation of the Drinking Philosopher Problem to the multi-agent collision-avoidance problem on a road network, where the resource agents compete for is road occupancy. We design a decision-making strategy that defines how agents choose their actions. Minimal communication among agents allows each agent to consistently establish precedence and resolve conflicts in a local, decentralized manner. Unlike previous work by Sahin and Ozay in [20], our framework leverages the structure of the driving road network and takes into account the inertial properties of agents. Furthermore, our work guarantees safety and liveness

if all agents operate according to the specified decision-making strategy.

The main contributions of this paper are as follows:

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 follow in two different contexts on a road network (i.e. road segments and intersections).
3) Safety and liveness proofs when all agents operate according to these local rules.
4) Simulations as proof of concept of the safety and liveness guarantees.

II. Overview

In this overview, we give a high-level presentation of the main contributions in this work. All the terms will be formalized and described in further depth in later sections of the paper.

A. The Quasi-Simultaneous Discrete-Time Game

In this paper, we introduce the quasi-simultaneous discrete-time multi-agent game. The quasi-simultaneous game is a modification of turn-based games so that turn order is defined locally and induced by the agent states as opposed to being fixed and predefined. This new game format leverages the structure of the game environment to assign precedence among agents. In this way, it partially constrains the set of actions an agent can choose from based on the agent environment (via the order the agent gets to take its turn in). The quasi-simultaneous game models the agent's decision-making process in a multi-agent game differently than traditional simultaneous or turn-based games found in [10] and [8], and to the best of the authors' knowledge have not been introduced in the literature before.

B. The Agent Protocol

The Agent Protocol is defined to establish local rules agents must follow while making decisions on the road network.

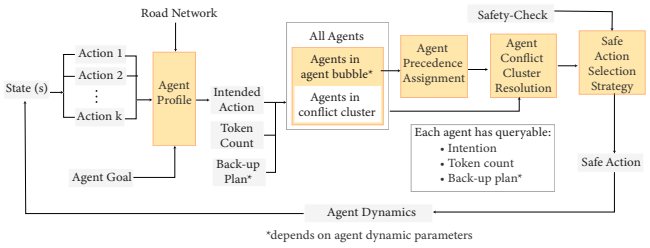

Fig. 1: Agent Protocol Architecture.

The novelty of our work is the introduction of a single backup plan action that all agents rely on. 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. The following is an overview of the Agent Protocol. For each time step of the game, each agent first assigns local precedence according to the rules described in Section IV-D, thereby establishing a consistent turn order (in a local manner). Then, each agent evaluates a set of actions and chooses the best one according to their own Agent Profile (thereby selecting their intended action), described in Section IV-E.

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 described in IV-F, defines how agents should ultimately decide which action to select. 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 a particular set of dynamics. Note that specific assumptions specified in Sections V and VI, must hold on the road network for the guarantees to hold.

III. Quasi-Simultaneous Discrete-Time Multi-Agent Game

We formalize the definition of a quasi-simultaneous discrete-time multi-agent game as follows. 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 (where locality is described in Section IV-C).

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. Let us define some preliminaries before formally defining the game.

We define a quasi-simultaneous game where all agents act in a local, decentralized manner as follows

$$\mathfrak{G} = \langle \mathfrak{A}, \mathscr{Y}, Act_{[\cdot]}, \rho_{[\cdot]}, \tau_{[\cdot]}, P \rangle \tag{1}$$

where

- $\mathfrak{A}$ is the set of all agents in the game $\mathfrak{G}$.
- $\mathscr{Y}$ is the set of all variables in the game $\mathfrak{G}$.
  - Let $\mathscr{U}$ be the set of all $\mathscr{Y}$-states, i.e. all possible assignment of values (states) of the game.
  - Given a subset $Y \subseteq \mathscr{Y}$, denote by $\mathscr{U}\,|_Y$ the projection of $\mathscr{U}$ onto Y, i.e. all possible states of the variables in the set Y.
- For each agent $\mathrm{Ag} \in \mathfrak{A}$, let:
  - S_{Ag} be the set $\mathscr{U}\,|_{\mathscr{Y}_{\mathrm{Ag}}}$ that contains all possible states of Ag.
  - Act_{Ag} be the set of all possible actions Ag can take.
  - $\tau_{\mathrm{Ag}} : S_{\mathrm{Ag}} \times Act_{\mathrm{Ag}} \to S_{\mathrm{Ag}}$. τ_{Ag} be the transition function that defines the state an agent will transition to when taking an action $a \in Act_{\mathrm{Ag}}$ from a given state.

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

- $P : \mathscr{U} \to \text{PolyForest}(\mathfrak{A})$, is the precedence assignment function where PolyForest is an operator that maps a set X to the set of all forests (undirected graphs whose connected components are trees) defined on the set X. The polyforest 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,

$$\forall \text{Ag} \in \mathfrak{A}. \forall s \in S_{\text{Ag}}. \text{Domain}(\tau_{\text{Ag}}(s, \cdot)) = \rho_{\text{Ag}}(s).$$

A. Agent Game Environment

Here we introduce the game environment agents are operating on.

Definition 3.1 (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 $g \in G$ have a set of associated properties $\mathscr{P}$, where $\mathscr{P} = \{p, d, \texttt{lo}\}$ 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 $\phi_l \in \{\texttt{north}, \texttt{east}, \texttt{south}, \texttt{west}\}$.

Let us define $\mathfrak{R}|_{\text{legal}} = (G, E|_{\text{legal}})$ to be a road network where a directed edge $e = (g_1, g_2)$ implies a legal (based on legal orientations) and dynamically-feasible transition (according to ρ_{Ag}) exists between the grid points $g_1, g_2 \in G$.

1) Special Grid Point Sets: Grid points where specific properties hold are given special labels, which can be seen in Fig. 2. These labels and the associated properties are defined as follows:

- $\mathscr{S}_{\text{sources}}$, ($\mathscr{S}_{\text{sinks}}$): A set of grid points designated for Ag to enter (or leave) the road network $\mathfrak{R}$ from. Any $s \in \mathscr{S}_{\text{sources}}$ ($s \in \mathscr{S}_{\text{sinks}}$) must be a grid point with no inbound (outbound) edges in $\mathfrak{R}\,|_{\text{legal}}$.
- $\mathscr{S}_{\text{intersection}}$: A set of grid points that contains all grid points with more than one legal orientation.
- $\mathscr{S}_{\text{traffic light}}$: A set of grid points that represent the traffic light states in the vertical or horizontal direction via its color (for every intersection).

2) Road Network Decomposition: The road network is hierarchically decomposed into lanes and bundles, which are defined as follows:

- Lanes and Bundles: Let lane $La(g)$ define a set of grid points that contains g and all grid points that form a line going through g. 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. Note, the precise mathematical definitions of lanes and bundles can be found in the Appendix.
- Road Segments RS: Each bundle can be decomposed into a set of road segments (separated by intersections). For each bundle, let us define a graph $G_{RS} = (G, E)$ where G is a set of grid points and E is a set of edges between any two grid points g_1, g_2 where $g_1, g_2 \notin \mathscr{S}_{\text{intersection}}$ and $Bu(g_1) = Bu(g_2)$. Road segments $Rs(g)$ is the set of all grid points that are in the same connected component as the grid point g in the graph G_{RS}.

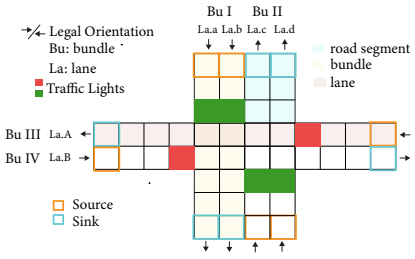

Fig. 2: Road network decomposition where each box represents a grid point.

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

Definition 3.2 (Road Network Dependency Graph): We define a dependency graph $G_{\text{dep}} = (RS, E)$, RS is the set of road segments in $\mathfrak{R}$ and a directed edge $e : (r_1, r_2)$ for $r_1, r_2 \in RS$ implies that an agent Ag whose $x-y$ position is $(s.x_{Ag}, s.y_{Ag}) \in r_1$ depends on the clearance of some agent Ag$'$ whose $x-y$ position $(s.x_{Ag'}, s.y_{Ag'}) \in r_2$, where $s.x_{Ag}$ and $s.y_{Ag}$ are the x and y positions of agent Ag and are defined formally in Section IV-A.

For clarity of the road network decomposition, refer to Fig. 2. The precise set of rules that traffic lights are operating according to can be found in more detail in the Appendix.

IV. Agent Protocol

In the following section, we present the set of attributes agents must have and the set of rules agents must adhere to in order to satisfy our proposed Agent Protocol.

A. Agent Attributes

Each *agent* Ag is characterized by a set of variables $\mathscr{V}_{\text{Ag}}$ such that

$$\{\texttt{Id}_{\text{Ag}}, \texttt{Tc}_{\text{Ag}}, \texttt{Goal}_{\text{Ag}}\} \subseteq \mathscr{V}_{\text{Ag}}$$

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 IV-F.

In this paper, we only consider *car* agents such that if Ag $\in \mathfrak{A}$, then $\mathscr{V}_{\text{Ag}}$ includes x_{Ag}, y_{Ag}, θ_{Ag}, v_{Ag}, namely its absolute coordinates, heading and velocity. $\mathscr{V}_{\text{Ag}}$ also has parameters:

$$a_{\text{min Ag}} \in \mathbb{Z}, a_{\text{max Ag}} \in \mathbb{Z}, v_{\text{min Ag}} \in \mathbb{Z} \text{ and } v_{\text{max Ag}} \in \mathbb{Z}$$

which define the minimum and maximum accelerations and velocities respectively.

The agent control actions are defined by two parameters: 1) an acceleration value acc_{Ag} between $a_{\text{min Ag}}$ and $a_{\text{max Ag}}$

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 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, while taking its maneuver.

Agent grid point occupancy is defined as follows. Note that agents occupy a single grid point at a given time, but when taking an action, they may occupy a set of grid points. More formally:

Definition 4.1 (Grid Point Occupancy): The notion of grid point occupancy is captured by the definitions of the following maps for each $\text{Ag} \in \mathfrak{A}$. To define the grid point an agent is occupying at a given time we use the map: $\mathscr{G}_{\text{Ag},t} : S_{\text{Ag}} \to 2^G$, mapping each agent to the single grid point the agent occupies. By a slight abuse of notation, we let $\mathscr{G}_{Ag,t} : S_{\text{Ag}} \times Act_{\text{Ag}} \to 2^G$ be a function that maps each $s \in S_{\text{Ag}}$ and $a \in \rho_{\text{Ag}}(s)$ 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.

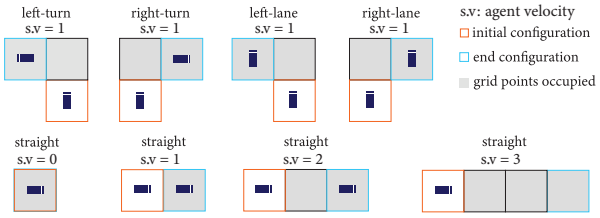

Fig. 3: Shows different grid point occupancy associated with different discrete agent maneuvers.

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.

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 $La(\text{Ag})$.

1) Motion-Planning Algorithm: 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 the liveness guarantees rely on the assumption that rerouting of the agent's motion plan is not supported.

B. 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 if each agent can maintain the ability to safely execute its own backup plan (i.e. keep a far enough distance behind a lead agent), the safety of the collective system safety 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 4.2 (Backup Plan Action): The backup plan action a_{bp} is a control action where $a = \max(a_{\text{min}}, -s.v_{Ag})$. and $\gamma_{\text{Ag}} = \texttt{straight}$. Note, the max is because applying the deceleration (a_{min}) should not push the car velocity below 0. Note a_{min} is less than 0.

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

C. 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.

1) 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 4.3 (Bubble): Let Ag with state $s_0 \in \mathscr{S}_{\text{Ag}}$. Let agent Ag$'$ be another agent. Then the bubble of Ag with respect to agents of the same type as Ag$'$ is given by $\mathscr{B}_{\text{Ag/Ag}'}(s_0)$. 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 Ag considered in this paper have the same attributes, for ease of notation, we refer to the bubble of Ag as $\mathscr{B}_{\text{Ag}}$.

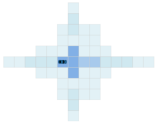

Fig. 4: Bubble if all $\text{Ag} \in \mathfrak{A}$ have the Agent Dynamics specified in Section IV-A.

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 $\text{Ag}' \in \mathscr{B}_{\text{Ag}}(s)$ if $(s.x_{\text{Ag}'}, s.y_{\text{Ag}'})$ is on a grid point in the set $\mathscr{B}_{\text{Ag}}(s)$. The details for the construction of the bubble for an agent with a particular set of attributes and dynamics can be found in the Appendix.

2) Intersections: The locality of information 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 $\text{Ag}' \in \mathfrak{A}$ 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.

D. Precedence Rules

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: $\text{proj}^B_{\text{long}} : \mathfrak{A} \to \mathbb{Z} \cup \{\varnothing\}$ as $\text{proj}^B_{\text{long}}(\text{Ag})$ = the projection of the Ag's state onto the bundle B if Ag is in B and otherwise $\text{proj}^B_{\text{long}}(\text{Ag}) = \varnothing$. In other words, $\text{proj}^B_{\text{long}}(Ag)$ is the mapping from an agent 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.

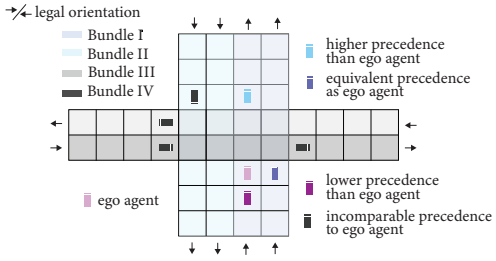

Fig. 5: Rules for precedence assignment.

For every agent Ag', the agent Ag defines the precedence relation between Ag and Ag' using the following set of precedence rules:

1) 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 $\text{Ag} \sim Ag'$ and we say Ag and Ag' are equivalent in precedence.
3) If Ag' and Ag are not in the same bundle, then the two agents are incomparable.

Each agent $\text{Ag} \in \mathfrak{A}$ only assigns precedence according to the above rules locally to agents within its perception region (i.e. bubble on road segments and a slightly larger region at intersections, defined in Section IV-C) when making a decision of which action to take. 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 4.1: If all agents assign precedence according to the local precedence assignment rules to agents in their respective bubbles, then precedence relations will induce a polyforest on $\mathfrak{A}/\sim$ (the quotient set of S by $\sim$).

Proof: (Sketch) This result follows from the total linear ordering assigned by the first local precedence assignment rule. The full proof can be found in the Appendix. ■

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 agents should do either 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, defined in Section IV-F.

E. Assume-Guarantee Profile

An assume-guarantee profile, introduced in [16], is a mechanism for ordering agent specifications into a specific hierarchy so the process for choosing an agent's preferred action is transparent and safe. The concept is related to the concepts of minimum-violation planning [24], [5].

In this work, each agent uses an assume-guarantee profile to *propose* an intended action. From [16], the assume-guarantee profile requires defining a set of agent specifications. For completeness, we define how specifications are evaluated in the game.

Let $r \in R$ denote a specification for an agent and $\text{Ag} \in \mathfrak{A}$. For the specification r, an oracle evaluates whether an agent taking an action in the current joint state game configuration will satisfy the specification. More formally, the oracle is defined as follows $\mathscr{O}_{\text{Ag},t} : R \times S_{\text{Ag}} \times Act_{\text{Ag}} \times \mathscr{U} \to \mathbb{B}$ where $\mathbb{B} = \{\texttt{T}, \texttt{F}\}$ and the subscript t denotes the time-step the oracle is evaluated. These evaluations can easily be refactored to accommodate specifications that are more continuous in nature.

In this work, each agent has a total of ten different specifications, three of whose oracles are defined as follows:

1) $O_{Ag,t,\textbf{dynamic safety}}(s,a,u)$**:** 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) $O_{Ag,t,\textbf{unprotected left-turn safety}}(s,a,u)$ 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 left-turn (while traffic light is green), traffic light turns red, and no agents in oncoming lanes.

3) $O_{Ag,t,\textbf{reachability preservation progress}}(s,a,u)$ returns `T` if the action a from the state s will allow Ag$'s$ planned path to remain reachable.

The definitions of the other oracles shown in Fig. 6 are relatively straight-forward and can be found in the Appendix.

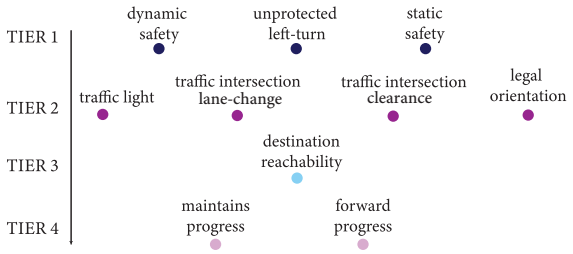

Fig. 6: Assume-guarantee profile that shows ordering of specifications, where specifications on the same tier are incomparable to one another.

The ordering of the specifications that each agent must follow is shown in Fig. 6. In [16], the consistent-evaluating function W evaluates each action based on the number of specifications it satisfies in each tier, where actions satisfying more specifications in the higher tiers are valued more highly. Details can be found in [16]. The action with the highest value is then selected as the action the agent takes.

For this work, the agent profile is used to define the agent's intended action a_i and the agent's best straight action a_{st} which is defined in Section IV-G. Note, the dynamic obstacle safety oracle 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).

F. 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 Agent profile. When it is the agent's turn to select an action, it must choose whether or not to take it's 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 should get to take its action.

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

Definition 4.4 (Agent-Action Conflict): Let us consider an agent Ag 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 either of the following conditions holds:

- $\mathscr{G}_{\text{Ag},t}(s,a) \cap \mathscr{G}_{\text{Ag'},t}(s',a') \neq \emptyset$,
- Let $s_{t+1} = \tau_{\text{Ag}}(s,a)$ and $s'_{t+1} = \tau_{\text{Ag}'}(s',a')$: If $La(s_{t+1}) = La(s'_{t+1})$ and $d(s_{t+1}, s'_{t+1}) \leq gap_{\text{req}}$,

where $d(s_{t+1}, s'_{t+1})$ defines the l_2 distance between two states in the same lane and gap_{req} is the minimum distance between two agents in the same lane so that if the agent in front applies their backup plan, the agent behind will be able to apply their own backup plan without colliding with the former. gap_{req} can easily be computed depending on the agent dynamics.

In the case that an agent's action does conflict with another agent, 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 any agent (i.e. 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$'$:

1) 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. $\gamma_{Ag} \in \{\texttt{left-lane change}, \texttt{right-lane change}\}$.
- Ag$' \in \mathscr{B}_{Ag}(s)$, i.e. Ag$'$ is in agent Ag's bubble.
- Ag$' \precsim Ag$, i.e. Ag has 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.
- $\mathscr{F}_{\text{Ag}}(u, a_i) = \texttt{F}$, where $\mathscr{F}_{\text{Ag}}(u, a_i)$ is the max-yielding-not enough flag and is defined below.

Definition 4.5 (maximum-yielding-not-enough flag): The maximum-yielding-not-enough flag $\mathscr{F}_{\text{Ag}} : \mathscr{U} \times Act_{\text{Ag}} \to \mathbb{B}$ that is defined as follows:

$$\mathscr{F}_{\text{Ag}}(u, a_i) = \begin{cases} \texttt{T} & \text{if } \exists \text{Ag}' \in \mathscr{B}_{Ag}(s) s.t. ((\text{Ag}, a_i) \dagger (\text{Ag}', a_{\text{bp}})) \\ \texttt{F} & \text{otherwise} \end{cases}$$

When the flag is set, it indicates a configuration in which even if Ag$'$ maximally yielded to Ag, if Ag did a lane-change it would violate the safety of Ag$'$'s backup plan action.
We note that if $\mathscr{F}_{\text{Ag}}(u, a_i)$ 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 that define the set of agents it is ultimately bidding for its priority (to take its intended action) over. These clusters of agents are defined as follows:

Definition 4.6 (Conflict Cluster): A conflict cluster for an agent Ag is defined as $\mathscr{C}_{Ag} = \{Ag' \in \mathfrak{A} \mid Ag \texttt{ send } \text{Ag}' \text{ or } Ag' \texttt{ send } Ag\}$, 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.

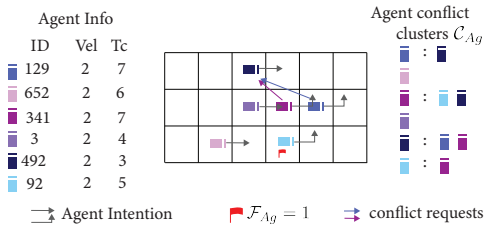

Fig. 7: Conflict clusters

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 it's favor, as shown in Fig. 7.

2) Token Resolution: The conflict resolution strategy must be fair, meaning each agent always eventually wins a conflict resolution. The resolution is based on the agents' token counts $\texttt{Tc}$, which is updated by agents to represent how many times an agent has been unable to take a forward progress action.

Given the assumption that all agents can query the token counts of all other agents, let us define the conflict resolution strategy. For each $\text{Ag} \in \mathfrak{A}$, let $\mathscr{W}_{\text{Ag}} \in \mathbb{B}$ be an indicator variable for whether or not the agent has won in its conflict cluster. Let $\texttt{Tc}_{Ag}$ represent the token count of the agent when it has sent its request. Let $\texttt{Id}_{Ag}$ represent a unique ID number of an agent. The conflict cluster resolution indicator variable $\mathscr{W}_{\text{Ag}}$ is determined as follows:

$$\mathscr{W}_{\text{Ag}} \triangleq \forall \text{Ag}' \in \mathscr{B}_{\text{Ag}}(s) : (\texttt{Tc}_{\text{Ag}'} < \texttt{Tc}_{\text{Ag}}) \vee ((\texttt{Tc}_{\text{Ag}'} = \texttt{Tc}_{\text{Ag}}) \wedge \texttt{Id}_{\text{Ag}'} < \texttt{Id}_{\text{Ag}})$$

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 lemma, which comes from the definition of the conflict-cluster resolution scheme, is a helpful for proving safety of the agent protocol.

Lemma 4.2: At most one agent will win in each agent's conflict cluster.

Proof: (Sketch): This follows from the definition of conflict clusters (i.e. all agents will only send and receive requests from agents within its bubble), and the winner-takes-all conflict-cluster resolution. The full proof can be found in the Appendix. ■

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

G. 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 4.7, 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 agent profile and consistent evaluating function defined in Section IV-E.
2) Ag's role in conflict request cluster being:
   - A conflict request sender ($\exists \text{Ag}' \in \mathscr{B}_{\text{Ag}}(s)$: Ag $\texttt{send}$ Ag').
   - A conflict request receiver ($\exists \text{Ag}' \in \mathscr{B}_{\text{Ag}}(s)$: Ag' $\texttt{send}$ Ag).
   - Both a sender and a receiver of conflict requests.
   - Neither a conflict request sender or receiver.
3) The agent's conflict cluster resolution $\mathscr{W}_{\text{Ag}}$.
4) Evaluation of $O_{\text{Ag},t,\text{dynamic safety}}(s,a_i,u)$.
5) $\mathscr{F}_{\text{Ag}}(u,a_i)$ for Ag is raised, where $\mathscr{F}_{\text{Ag}}(u,a_i)$ is the maximal-yielding-not-enough flag defined in Section IV-F.

The Action Selection Strategy decision tree, shown in Fig. 8, defines how agents should select which action to take based on the five different conditions.

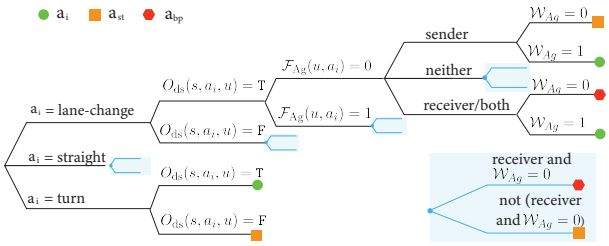

Fig. 8: Agent action selection strategy.

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 4.7 (Best Straight Action): Let us consider Ag and its associated action set $\rho_{\text{Ag}}(s)$. Let us define $\rho_{Ag}(s)\,|_{\text{st}} \triangleq \{a \in \rho_{\text{Ag}}(s) \mid \gamma_{\text{Ag}} = \texttt{straight}\}$, i.e. the set of all allowable straight actions of Ag. The best straight action $a_{st} = \arg\max_{a \in \rho_{\text{Ag}}(s)|_{\text{st}}} W_{\text{Ag}}(a)$, where W is the consistent evaluating function defined with respect to the agent profile in Fig. 6, with they dynamic obstacle safety oracle ($O_{\text{Ag},t,\text{dynamic safety}}(s,a,u)$) included.

When an agent is both a receiver of a conflict request and a loser in its conflict cluster, it must yield to the agent that it received the conflict request from. The Action Selection Strategy requires the agent takes its backup plan action a_{bp} in these scenarios, where the backup plan action a_{bp} is the control action defined in Definition 4.2.

1) Token Count Update: The token count updates according to the agent's chosen action. In particular, if Ag selects action a:

$$\texttt{Tc}_{\text{Ag}} = \begin{cases} \texttt{Tc}_{\text{Ag}} + 1 & \text{if } O_{\text{forward progress}}(s,a,u) = \text{F} \\ 0 & \text{otherwise} \end{cases}$$

.

V. 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 5.1 (Collision): An agent Ag that takes an action $a \in Act_{\text{Ag}}$ will cause collision if either of the following conditions hold:

1) $\mathscr{G}_{\text{Ag},t}(s,a) \cap (\cup_{\text{Ag}'} \mathscr{G}_{\text{Ag}'}(s',a')) \neq \emptyset$.
2) $\mathscr{G}_{\text{Ag},t}(s,a) \cap O_{\text{st}} \neq \emptyset$, where $O_{\text{st}} = \{g \in G \mid g.d = 0\}$, i.e. the set of all undrivable grid points.

In other words, when Ag's action a causes it to overlap in occupancy grid with another agent's occupancy grid or a static obstacle. Note when the agent Ag$'$ is not simultaneously taking an action with Ag, the occupancy set $\mathscr{G}_{\text{Ag}'}(s',\cdot)$ is the singleton set for the agent Ag$'$, representing the single grid point Ag' is occupying.

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 is thus also defined to avoid violating the safety of its own and any other agent's backup plan action a_{bp} defined in Section IV-B. An agent's backup plan action a_{bp} is evaluated to be safe when the following conditions hold:

Definition 5.2: [Safety of a Backup Plan Action] Let us define the safety of an agent's backup plan action $S_{\text{Ag},bp}$: $\mathscr{U} = \mathbb{B}$, where $\mathbb{B} = \{\mathtt{T},\mathtt{F}\}$ 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 agent profile defined in Section IV-E.

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 5.3 (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 where u' is the state of the game after Ag has taken its action, then $S_{\text{Ag}',bp}(u') = \mathtt{F}$.

Note, when $\text{Ag} \neq \text{Ag}'$, then Ag can only violate the backup plan action of the agent Ag$'$ with its action a if the following conditions hold:

1) $\mathscr{G}_{\text{Ag}}(s,a) \cap \mathscr{G}_{\text{Ag}'}(s',a') \neq \emptyset$,
2) Let $s_{t+1} = \tau_{Ag}(s,a)$ and $s'_{t+1} = \tau_{Ag'}(s',a')$: If $\mathscr{G}_{\text{Ag}}(s_{t+1}), \mathscr{G}_{\text{Ag}'}(s_{t+1})$ are in the same lane and if $d(s_{t+1}, s'_{t+1}) < gap_{\text{req}}$,

where $d(s_{t+1}, s'_{t+1})$ and gap_{req} are the same as defined in Section 4.4.

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 [13].

Theorem 5.1 (Safety Guarantee): Given all agents $\text{Ag} \in \mathfrak{A}$ in the quasi-simultaneous game select actions in accordance to the Agent Protocol specified in Section IV, then 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 $\forall Ag, S_{\text{Ag},bp}(s,u) = \mathtt{T}$, i.e. each agent has a backup plan action that is safe, as defined in Section 5.2. 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 (i.e. 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 the Appendix.

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) $\forall \text{Ag}, S_{\text{Ag},bp}(u') = \mathtt{T}$ 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 IV-A.
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 $S_{\text{Ag},bp}(s,u) = \mathtt{T}$.

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 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 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 in the immediate time-step from Condition 1 in, and the fact that all Ag have a safe backup plan action a_{bp} to choose from when Condition 2 holds, and will always be able to (and will) take an action from which it can avoid collision in future time steps.

■

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.

VI. Liveness Guarantees

Note, we introduce the definition of liveness from [13], as follows:

Definition 6.1 (Liveness): A liveness property asserts that program execution eventually reaches some desirable state. For our paper, we describe the eventual desirable state for each agent is to reach their respective final destinations. Proving fairness, as described in [13], 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 [17]. 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 6.2 (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 $g \in \mathscr{S}_{\text{intersections}}$. The sparsity condition must be such that $N < M - 1$, where N is the number of agents in the road network. Note, these sparsity conditions are conservative because it is a bound defined by the worst possible assignment of agents and their destinations (i.e. where all agents enter the smallest loop).

Now, we introduce the liveness guarantees under these sparse traffic conditions. The proof of liveness is based on the fact that 1) agent 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 6.1 (Liveness Under Sparse Traffic Conditions): Under the Sparse Traffic Assumption given by Definition 6.2 and given all agents $\text{Ag} \in \mathfrak{A}$ in the quasi-simultaneous game select actions in accordance to the Agent Protocol specified in Section IV, liveness is guaranteed, i.e. all $\text{Ag} \in \mathfrak{A}$ will always eventually reach their respective goals.

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

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 IV-C 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 $t_{\min} < \Delta t_{\text{tl}} < \infty$, and $t_{\min}$ is defined so that agents are slowed down sufficiently long such that an agent waiting to make a lane-change to a critical tile is such that its max-yielding-flag is not always set to $\texttt{T}$.
6) The static obstacles are not on any grid point g where $g.d = 1$.
7) Each Ag treats its respective goal $\text{Ag}.\texttt{g}$ 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 $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
   a) Let us consider a road segment $r \in RS$ that contains grid point(s) $g \in \mathscr{S}_{\text{sinks}}$, i.e. 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 every $\text{Ag} \in r$ will be able to always eventually take $a \in Act_{\text{Ag}}$ for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
   b) Let us consider a road segment $rs \in RS$. Let us assume $\forall rs \in RS, \exists (rs, rs') \in G_{\text{dep}}$, i.e. the clearance of rs depends on the clearance of all rs'. Inductive arguments based on agents' longitudinal distance to the front of the intersection are used to show that any

Ag on *rs* will always eventually take $a \in Act_{\text{Ag}}$ where $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.

c) For any $\mathfrak{R}$ where the dependency graph G_{dep} (as defined in Section 3.2) 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 all $\text{Ag} \in \mathfrak{A}$ will always eventually take $a \in Act_{\text{Ag}}$ for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.

d) When the graph G_{dep} is cyclic, the Sparsity Assumption 6.2 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 $O_{\text{forward progress}}(s,a,u)$, 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 agent profile are used for the arguments in the proof.

VII. Simulation

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, and 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 paper. 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) small city blocks grid and 3) large city blocks grid. A snapshot of a small city blocks grid simulation is shown in Fig. 9. 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.

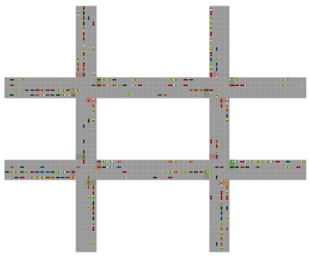

Fig. 9: Simulation

For all simulation trials collision does not occur. Although liveness is only guaranteed in sparse traffic conditions, we simulate for a number of agents $N > M - 1$ specified in the sparsity condition and deadlock does not occur. In particular, for the straight road segment, on average 77%, 36% and 43% made it to their respective destinations on the respective maps by the end of the 250 time-steps.

VIII. Conclusion and Future Work

In this paper, we have proposed a novel paradigm for designing safety-critical decision-making modules for agents whose behavior is extremely complex and highly-coupled with other agents. The main distinction of our proposed architecture from the existing literature, is the shift from thinking of each agents as separate, individual entities, to agents as a collective where all *all* agents adopt a *common* local, decentralized protocol (where additional customization can be built in later). The protocol defines the agent attributes, the region it must reason over (i.e. the bubble), how the agent chooses its intended agent, and how it ultimately selects which action to take. With this protocol, we are able to formally guarantee specifications safety and liveness (under sparse traffic conditions) for all agents. We validate the safety and liveness guarantees in a randomized simulation environment.

The current work still lacks 1) liveness guarantees in all scenarios, 2) robustness to imperfect sensory information and 3) does not account for other agent types like pedestrians and cyclists. Future work will focus on modifying the agent strategy architecture to prevent the occurrence of the loop deadlock introduced in Section VI from occurring. In addition to providing stronger liveness guarantees, the architecture must be modified in a way to effectively accommodate impartial and imperfect information. We also hope to accommodate a diverse, heterogenous set of car agents and also other agent types like pedestrians and cyclists. Although the work needs to be extended to make more applicable to real-life systems, we believe this work is a first step towards defining a comprehensive method for guaranteeing safety and liveness for all agents in an extremely dynamic and complex environment.

IX. Acknowledgments

We would like to acknowledge K. Mani Chandy who provided valuable input to the problem formulation and presentation of ideas in the manuscript and to Giovanna Amorim for her contributions to the simulation code.

X. Author Contributions

K.X.C., R.M.M., and T.P-M. jointly conceived the conceptual framework. K.X.C. and T.P-M. jointly developed the problem formulation and theoretical approach. K.X.C. worked out the main proofs with input from T.P-M. K.X.C. drafted the manuscript and figures with input from T.P-M. S-J.C. and R.M.M. provided guidance on the overall approach and provided feedback on the final manuscript.

References

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

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

[3] Craig Boutilier. Sequential optimality and coordination in multiagent systems. In *IJCAI*, volume 99, pages 478–485, 1999.

[4] 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)*, pages 686–693, Auckland, New Zealand, 2019. IEEE.

[5] 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 *2019 International Conference on Robotics and Automation (ICRA)*, pages 8536–8542, Montreal, QC, Canada, 2019. IEEE.

[6] K. M. Chandy and J. Misra. The drinking philosophers problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 6(4):632–646, 1984.

[7] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58, New York, New York, USA, 2016. JMLR.

[8] 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)*, pages 9590–9596, Montreal, Canada, 2019. IEEE.

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

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

[11] N. A. Greenblatt. Self-driving cars and the law. *IEEE Spectrum*, 53(2):46–51, 2 2016.

[12] W. Li, D. Sadigh, S. S. Sastry, and S. A. Seshia. Synthesis for human-in-the-loop control systems. In *TACAS*, pages 470–484, Grenoble, France, 2014. Springer Berlin Heidelberg.

[13] S. Owicki and L. Lamport. Proving liveness properties of concurrent programs. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):455–495, 1982.

[14] 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. *T-IV*, 1(1):33–55, 3 2016.

[15] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[16] 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)*, pages 2788–2795, Nice, France, 2019. IEEE.

[17] S. A. Reveliotis and E. Roszkowska. On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems. *IEEE Transactions on Automatic Control*, 55(7):1646–1651, 2010.

[18] 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.

[19] 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)*, volume 2, Ann Arbor, MI, USA, 2016.

[20] Y. E. Sahin and N. Ozay. From drinking philosophers to wandering robots. *arXiv preprint arXiv:2001.00440*, 2020.

[21] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a Formal Model of Safe and Scalable Self-driving Cars. *arXiv e-prints*, page arXiv:1708.06374, 8 2017.

[22] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial intelligence*, 73(1-2):231–252, 1995.

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

[24] 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*, pages 1–10, Philadelphia, Pennsylvania, USA, 2013. ACM.

[25] W. van Der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness, feasibility, and synthesis. *Synthese*, 156(1):1–19, 2007.

[26] T. Wongpiromsarn, S. Karaman, and E. Frazzoli. Synthesis of provably correct controllers for autonomous vehicles in urban environments. In *ITSC*, pages 1168–1173, Washington, DC, USA, 10 2011. IEEE.

[27] 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*, pages 5011–5018, Karlsruhe, Germany, 5 2013. IEEE.

[28] M. Wooldridge and W. Van Der Hoek. On obligations and normative ability: Towards a logical analysis of the social contract. *Journal of Applied Logic*, 3(3-4):396–420, 2005.

A. Road Network

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

1) Grid Point Properties: The set of properties $\mathscr{P} = \{p, d, \texttt{lo}\}$ of each grid point $g \in G$, where $p \in \mathbb{Z}^2$ denotes the Cartesian coordinate of the grid point, $d \in \{0,1\}$, which denotes whether a node is drivable or not and `lo` are the legal orientations, which is a set of headings ϕ_l where each $\phi_l \in \{\texttt{north}, \texttt{east}, \texttt{south}, \texttt{west}\}$ that contains the set of legal orientations on the grid point. 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 a 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 and is adjacent to. Let $N(g)$ = adj(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. Bubble Construction

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

Definition 0.1 (Backup Plan Node Set): Let $\text{Ag} \in \mathfrak{A}$ and $s_0 \in S_{\text{Ag}}$. The backup plan grid point set $BP_{\text{Ag}}(s_0)$ is all the grid points agent Ag occupies as it applies maximum deceleration to come to a complete stop. $BP_{\text{Ag}}(s_0) = \mathscr{G}_{\text{Ag}}(s_0, a_{\text{bp}}) \cup BP_{\text{Ag}}(\tau_{\text{Ag}}(s_0, a_{\text{bp}}))$ if $\tau_{\text{Ag}}(s_0, a_{\text{bp}}).v \neq 0$ and $BP_{\text{Ag}}(s_0) = \mathscr{G}_{\text{Ag}}(\tau(s_0, a_{\text{bp}}))$ otherwise, where $a_{\min}$ is the agent's action of applying maximal deceleration while keeping the steering wheel at the neutral position.

Definition 0.2 (Forward/Backward Reachable States): The (1-step) forward reachable state set of agent Ag denoted $\mathscr{R}_{\text{Ag}}(s_0)$ represents the set of all states reachable by Ag from the state s_0. The forward reachable set is defined as $\mathscr{R}_{\text{Ag}}(s_0) \triangleq \{s \in S_{\text{Ag}} \mid \exists a \in \rho_{\text{Ag}}(s_0).s = \tau(s_0, a)\}$. Similarly, we define the (1-step) backward reachable state set $\mathscr{R}^{-1}_{\text{Ag}}(s_0)$ as the set of all states from which the state s_0 can be reached by Ag. Formally, $\mathscr{R}^{-1}_{\text{Ag}}(s_0) \triangleq \{s \in S_{\text{Ag}} \mid \exists s \in S_{\text{Ag}}.\exists a \in \rho_{\text{Ag}}(s).s_0 = \tau(s, a)\}$.

Definition 0.3 (Forward Reachable Nodes): We denote by $\mathscr{G}^{\mathscr{R}}_{\mathrm{Ag}}(s_0)$ 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 $\mathscr{R}_{\mathrm{Ag}}(s_0)$. Specifically,

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

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

Definition 0.4 (Occupancy Preimage): For $n \in G$, where G are the nodes in the road network graph $\mathfrak{R}$, the *occupancy preimage* $\mathscr{G}^{\mathscr{R}^{-1}}_{\mathrm{Ag}}(n)$ 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,

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

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. 10. The bubble is defined to be the union of these sets of grid points.

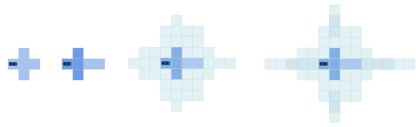

Fig. 10: Bubble if all $\mathrm{Ag} \in \mathfrak{A}$ have the Agent Dynamics specified in Section IV-A. Construction of this set defined in the Appendix.

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 $\mathscr{G}^{\mathscr{R}}_{\mathrm{Ag}}(s_0)$ is shown in the left-most figure in Fig. 10. The grid points an agent occupies when executing its backup plan from a state in the agent's forward reachable set $\mathscr{R}_{\mathrm{Ag}}(s_0)$ is given by:

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

These grid points are shown in the second from the left subfigure in Fig. 10. The set-valued map

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

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 $\mathrm{Ag}' \in \mathfrak{A}$ and $\mathrm{Ag}' \neq \mathrm{Ag}$. The set:

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

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

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

These grid points are defined in the third from the left subfigure in Fig. 10.

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:

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

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 $\mathscr{Z}_{\mathrm{Ag}}(s_0)$. Let

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

This is the set of all states backward reachable to the states in $\mathscr{S}^{BP}_{\mathrm{Ag}'}(\mathrm{Ag}, s_0)$. 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 $\mathscr{Z}_{\mathrm{Ag}}$. We project this set of states to a set of grid points as

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

Note, this set of grid points is shown in the right-most subfigure in Fig. 10. The bubble is then defined as the union of all the sets of grid points specified above.

Definition 0.5 (Bubble): Let us consider an agent Ag with state $s_0 \in S_{\mathrm{Ag}}$ and agent Ag′ be another agent. Then the bubble of Ag with respect to agents of the same type as Ag′ is given by

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

Note that under almost all circumstances, we should have

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

so $\mathscr{B}_{\mathrm{Ag}}(s_0)$ is simply equal to $\mathscr{G}^{\mathscr{R},BP}_{\mathrm{Ag}'}(\mathrm{Ag}, s_0)$. 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.

Lemma 0.1: 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 $\mathfrak{A}/\sim$, where $S/\sim$ defines the quotient set of a set S.

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 $\mathrm{Ag} \in R_C$ be one of these representatives. Applying the second 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. ■

C. Oracle Definitions

1) $O_{\textbf{static safety}}(s,a,u)$ 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.
2) $O_{\textbf{traffic light}}(s,a,u)$ 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 $s' = \tau_{\mathrm{Ag}}(s,a)$ and not violate the traffic-light law.
3) $O_{\textbf{legal orientation}}(s,a,u)$ returns `T` if the action a from the state s follows the legal road orientation.
4) $O_{\textbf{traffic intersection clearance}}(s,a,u)$ returns `T` if the action causes the agent to enter the intersection and leave it when the traffic light turns red and if 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.
5) $O_{\textbf{traffic intersection lane change}}(s,a,u)$ returns `T` if the action is such that $\gamma_{Ag} = \{\texttt{left-lane change}, \texttt{right-lane change}\}$ and the agent either begins in an intersection or ends up in the intersection after taking the action.
6) $O_{\textbf{maintains progress}}(s,a,u)$ returns `T` if the action a from the state s stays the same distance to its goal.
7) $O_{\textbf{forward progress}}(s,a,u)$ returns `T` if the action a from the state s will improve the agent's progress towards the goal.

D. Safety Lemmas

In the following lemma, we show that an agent cannot send (or receive) a conflict request to (from) an agent outside its bubble.

Lemma 0.2: Let us consider agent Ag with state s and agent Ag$'$ at state s'. Ag `send` Ag$' \Rightarrow$ Ag $\in \mathscr{B}_{\mathrm{Ag}'}(s')$.

Proof: If A `send` B this means that all of the conditions specified in Section IV-F.1 must hold. IV-F.1 specifies that $(A,a_i) \dagger (B,a_i')$. This condition is only valid if either 4.4 holds, which implies $\mathrm{proj}_G s \in \mathscr{G}_{F,B}(B,A)$ or that 4.4 holds, which would imply
$\mathrm{proj}_G s \in \mathscr{G}_{F,BP}(B,A)$. Membership of Agent A's state in either of these sets implies $A \in \mathscr{B}(B)$. ■

The following lemma follows from the lemma above.

Lemma 0.3: 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 $\mathscr{C}(\mathrm{Ag})$. It follows from Lemma 0.2 that $\forall$Ag$'$, s.t. Ag `send` Ag$'$Ag$' \in \mathscr{B}_{Ag}(s)$ and Ag $\in \mathscr{B}_{Ag'}(s')$. It also follows that $\forall$Ag$'$ s.t. ,Ag `send` Ag$'$, Ag $\in \mathscr{B}_{\mathrm{Ag}'}(s')$ and Ag$' \in \mathscr{B}_{\mathrm{Ag}}(s)$. This means 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 following lemma states that if all 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 $\mathscr{B}_{\mathrm{Ag}}(s)$.

Lemma 0.4: If Ag is following the Agent Protocol, and $S_{\mathrm{Ag},bp}(u) = \mathtt{T}$, Ag will only choose an action $a \in Act_{\mathrm{Ag}}$ for which the following two conditions hold: 1) $\mathscr{G}_{\mathrm{Ag}}(s,a) \cap (\cup_{\mathrm{Ag}' \in S} \mathscr{G}_{\mathrm{Ag}'}(s',a')) = \emptyset$ and 2) $\forall \mathrm{Ag}' \in S$, $\neg((\mathrm{Ag},a) \perp \mathrm{Ag}')$, where the set $S \triangleq \{\mathrm{Ag}' | \mathrm{Ag}' \notin \mathscr{B}_{\mathrm{Ag}}(s) \wedge ((\mathrm{Ag}' \sim \mathrm{Ag}) \vee (\mathrm{Ag}' \prec \mathrm{Ag}) \vee (\mathrm{Ag} \prec \mathrm{Ag'}))\}$.

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

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

Proof: We prove this by using definitions of elements in the Agent Protocol.

1) Let us first show 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. IV-E.
   a) According to the Action Selection Strategy defined in Section IV-G, 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} is selected via the Agent Profile and consistent-function evaluator defined in Section IV-E.
      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 Agent 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 Agent 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 IV-E, this implies 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 than it.

Lemma 0.6: If Ag is following the Agent Protocol, and $S_{\mathrm{Ag},bp}(u) = \mathtt{T}$, Ag will only choose an action $a \in Act_{\mathrm{Ag}}$ for which the following two conditions hold: 1) $\mathscr{G}_{Ag}(s,a) \cap (\cup_{Ag' \in S} \mathscr{G}_{\mathrm{Ag}'}(s',a')) = \emptyset$ and 2) $\forall \mathrm{Ag}' \in S,\ \neg((Ag,a)\perp Ag')$, where the set $S \triangleq \{\mathrm{Ag}' | \mathrm{Ag} \prec \mathrm{Ag}'\}$, 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 $Ag' \notin \mathscr{B}_{\mathrm{Ag}}(s)$.
   a) Proof by Lemma 0.4.
2) Now, let us consider all Ag' where $\mathrm{Ag} \prec \mathrm{Ag}'$ and $\mathrm{Ag}' \in \mathscr{B}_{\mathrm{Ag}}(s)$.
3) According to Lemma 0.5, Ag will only take an action that satisfies all oracles in the top two tiers, including $O_{\text{dynamic safety}}(s,a,u)$.
4) Since a is such that $O_{\text{dynamic safety}}(s,a,u) = \mathtt{T}$, by definition of the oracle, Ag will not cause collision with any $Ag' \in \mathscr{B}_{\mathrm{Ag}}(s)$.
5) For any $\mathrm{Ag} \prec \mathrm{Ag}'$, where Ag' has higher precedence than Ag, then $\mathrm{proj}_{\text{long}}(\mathrm{Ag}) < \mathrm{proj}_{\text{long}}(\mathrm{Ag}')$, i.e. Ag' is longitudinally ahead of Ag.
6) In order for $(\mathrm{Ag},a)\perp\mathrm{Ag}'$, the action a would have to have to be such that $s_f = \tau_{Ag}(s,a)$, and $La(s_f) = La(s')$ and $\mathrm{proj}_{\text{long}}(\mathrm{Ag}) > \mathrm{proj}_{\text{long}}(\mathrm{Ag}')$, where Ag is directly in front of Ag'.
7) Because of the agent dynamics defined in Section IV-A, any a such that $(\mathrm{Ag},a)\perp\mathrm{Ag}'$ will require $\mathscr{G}(\mathrm{Ag},a) \cap \mathscr{G}(\mathrm{Ag}') \neq \emptyset$.
8) Thus, any such action a will not satisfy the oracle $O_{\text{dynamic safety}}(s,a,u)$.
9) Since $S_{\mathrm{Ag},bp}(u) = \mathtt{T}$, by Assumption 6 in Section -E, the agent will have at least one action a_{bp} for which $O_{\text{dynamic safety}}(s,a,u) = \mathtt{T}$.
10) Since the agent will only choose an action for which $O_{\text{dynamic safety}}(s,a,u) = \mathtt{T}$ and it always has at least one action a_{bp} that satisfies the oracle, the agent will always choose an action for which $O_{\text{dynamic safety}}(s,a,u) = \mathtt{T}$ and thus will take an action such that $\forall Ag' \in S \neg((\mathrm{Ag},a)\perp\mathrm{Ag}')$.

■

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 lower precedence than it.

Lemma 0.7: If Ag is following the Agent Protocol, and $S_{\mathrm{Ag},bp}(u) = \mathtt{T}$, Ag will only choose an action $a \in Act_{\mathrm{Ag}}$ for which the following two conditions hold: 1) $\mathscr{G}_{Ag}(s,a) \cap (\cup_{Ag' \in S} \mathscr{G}_{\mathrm{Ag}'}(s',a')) = \emptyset$ and 2) $\forall Ag' \in S,\ \neg((Ag,a)\perp Ag')$, where the set $S \triangleq \{\mathrm{Ag}' | \mathrm{Ag}' \prec \mathrm{Ag}\}$, 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 $\mathrm{Ag} \prec \mathrm{Ag}'$ and $\mathrm{Ag}' \notin \mathscr{B}_{\mathrm{Ag}}(s)$.
   a) Proof by Lemma 0.4.
2) Now, let us consider all Ag' where $\mathrm{Ag} \prec \mathrm{Ag}'$ and $\mathrm{Ag}' \in \mathscr{B}_{\mathrm{Ag}}(s)$.
3) According to 3, Ag will only take an action that satisfies all oracles in the top two tiers, including $O_{\text{dynamic safety}}(s,a,u)$.
4) Since a is such that $O_{\text{dynamic safety}}(s,a,u) = \mathtt{T}$, by definition of the oracle, Ag will not cause collision with any $Ag' \in \mathscr{B}_{\mathrm{Ag}}(s)$.
5) According to the Action Selection Strategy defined in Section IV-G, 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, $((\mathrm{Ag},a_{bp})\perp\mathrm{Ag}')$ only if $La(\mathrm{Ag}) = La(\mathrm{Ag}')$.
   b) W.l.o.g., let us consider Ag' that is directly behind Ag.
   c) Since $S_{\mathrm{Ag}',bp}(s,u) = \mathtt{T}$, by Assumption 6 in Section -E, $O_{\text{dynamic safety}}(s,a_{bp},u) = \mathtt{T}$, 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 5.3, $\neg((\mathrm{Ag},a_{bp})\perp\mathrm{Ag}')$.
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 that a_{bp}.
8) Let us consider the intended action a_i.
   a) Let us consider when $\gamma_{\mathrm{Ag}} = \{\texttt{straight}\}$.
      i) This follows from 6.
   b) Let us consider when $\gamma_{\mathrm{Ag}} \in \{\texttt{right-turn}, \texttt{left-turn}\}$.
      i) If Ag takes such an action, Ag will end up in a state where $Bu(\mathrm{Ag}') \neq Bu(\mathrm{Ag})$ and from Definition 5.3, agent in different bundles cannot violate each others' backup plans.
   c) Let us consider when $\gamma_{\mathrm{Ag}} \in \{\texttt{right-lane change}, \texttt{left-lane change}\}$.
      i) $(\mathrm{Ag},a_i)\perp\mathrm{Ag}'$ when a_i is a lane change and the

agent 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 $\mathscr{F}_{\text{Ag}}(u,a_i)$, defined in Section 4.5 will be set.

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

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

■

The following lemma states that if all $\text{Ag} \in \mathfrak{A}$ 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 equal precedence.

Lemma 0.8: If Ag is following the Agent Protocol, and $S_{\text{Ag},bp}(u) = \texttt{T}$, Ag will only choose an action $a \in Act_{\text{Ag}}$ for which the following two conditions hold: 1) $\mathscr{G}_{\text{Ag}}(s,a) \cap (\cup_{\text{Ag}' \in S}\mathscr{G}_{\text{Ag}'}(s',a')) = \emptyset$ and 2) $\forall \text{Ag}' \in S$, $\neg((\text{Ag},a)\perp\text{Ag}')$, where the set $S \triangleq \{\text{Ag}'|\text{Ag}' \sim \text{Ag}\}$, 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 $\text{Ag} \prec \text{Ag}'$ and $\text{Ag}' \notin \mathscr{B}_{\text{Ag}}(s)$.
   a) Proof by Lemma 0.4.
2) Now, let us consider all Ag′ where $\text{Ag} \prec \text{Ag}'$ and $\text{Ag}' \in \mathscr{B}_{\text{Ag}}(s)$.
3) Let us first consider the agent itself, since an agent has equivalent precedence to itself.
   a) This is true by Lemma 0.5.
4) For any other agents of equivalent precedence that is not the agent itself, can be proven as follows.
5) Agents with equal precedence take actions simultaneously so $O_{\text{dynamic safety}}(s,a,u)$ does not guarantee no collision.
6) According to the Action Selection Strategy defined in Section IV-G, 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 $\text{Ag}' \sim \text{Ag}$ 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 $\text{Ag}' \in S$ and 2) $\neg((\text{Ag},a_{bp})\perp\text{Ag}')$.
   a) W.l.o.g. let us consider Ag′ where $\text{Ag}' \sim Ag$.
   b) The flag $\mathscr{F}_{\text{Ag}'}(u,a_i) = \texttt{T}$ 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 is action a_i when $\mathscr{F}_{\text{Ag}'}(u,a_i) = \texttt{T}$ 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 $\mathscr{F}_{\text{Ag}}(u,a_i)$ would be set.
9) Let us show Ag will only choose an a_{st} if it will 1) not collide with $\text{Ag}' \in S$ 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 4.6, the agent's action a_{st} will not cause Ag to collide with any $\text{Ag}' \in S$, 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 0.2, 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 $\text{Ag}' \in C_{\text{Ag}}$ where $\text{Ag} \sim \text{Ag}'$ will take a straight action.
   f) Since agents of equivalent precedence are initially in separate lanes by 7 and any $\text{Ag}' \in S$ 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 5.3, the action will not cause Ag to collide with any $\text{Ag}' \in S$, and $\forall \text{Ag}' \in S, \neg((\text{Ag},a_{st})\perp\text{Ag}')$.
10) Let us show Ag will only choose an a_i if 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 $\gamma_{\text{Ag}} = \texttt{straight}$ 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 $\mathscr{F}_{\text{Ag}}(u,a_i) = \texttt{F}$ and in the case that it is part of a conflict cluster, it is the winner of the conflict cluster resolution, i.e. $\mathscr{W}_{\text{Ag}} = \texttt{T}$,
      ii) By definition of $\mathscr{F}_{\text{Ag}}(u,a_i)$, 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 $\mathscr{W}_{\text{Ag}}$, by Lemma 0.2, if $\mathscr{W}_{\text{Ag}} = \texttt{T}$, 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 0.9: If Ag is following the Agent Protocol, and $S_{\text{Ag},bp}(u) = \texttt{T}$, Ag will only choose an action $a \in Act_{\text{Ag}}$ for which the following two conditions hold: 1) $\mathscr{G}_{\text{Ag}}(s,a) \cap (\cup_{\text{Ag}' \in S} \mathscr{G}_{\text{Ag}'}(s', a')) = \emptyset$ and 2) $\forall \text{Ag}' \in S$, $\neg((\text{Ag}, a) \perp \text{Ag}')$, where the set $S \triangleq \{\text{Ag}' | \text{Ag}' \not\sim \text{Ag}\}$, 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 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 top two tiers of their profile.
   e) Any Ag' in a perpendicular bundle will not enter 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 $O_{\text{unprotected left-turn}}(s,a,u)$.
   h) Thus Ag will not collide or violate the backup plan of agents in bundles of oncoming traffic.
2) Let us show 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 Agent 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 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 $\gamma_{Ag} =$ `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 $O_{\text{intersection lane-change}}(s,a,u)$ 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, 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 $O_{\text{dynamic safety}}(s,a,u) =$ `T` and $O_{\text{traffic light}}(s,a,u) =$ `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 $O_{\text{traffic-light}}(s,a,u) =$ `T` and $O_{\text{unprotected left-turn}}(s,a,u) =$ `T`.
      iv) Since all agents are following the traffic laws based on Proof -E, $O_{\text{traffic light}}(s,a,u) =$ `T` means action will not cause 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.

■

E. Safety Proof

Theorem 0.10: Given all agents $\text{Ag} \in \mathfrak{A}$ in the quasi-simultaneous game select actions in accordance to the Agent Protocol specified in Section IV, then 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 $\forall Ag, S_{\text{Ag},bp}(s,u) = \texttt{T}$, i.e. each agent has a backup plan action that is safe, as defined in 5.2. 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 (i.e. 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.

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) $\forall \text{Ag}, S_{\text{Ag},bp}(u') = \texttt{T}$ 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 $\text{Ag} \in \mathfrak{A}$ 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 IV-A.
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.
   a) In the case that the assertion P_t holds, let us show Ag will only choose an action $a \in Act_{Ag}$ for which the following two conditions hold: 1) $\mathscr{G}_{Ag}(s,a) \cap (\cup_{Ag' \in S} \mathscr{G}_{\text{Ag}'}(s',a')) = \emptyset$ 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 0.6.
      ii) $S \triangleq \{\text{Ag}' | \text{Ag}' \prec \text{Ag}\}$, i.e. agents with lower precedence than Ag. Proof by Lemma 0.7.
      iii) $S \triangleq \{\text{Ag}' | \text{Ag}' \sim \text{Ag}\}$, i.e. agents with equal precedence than the agent. Proof by Lemma 0.8.
      iv) $S \triangleq \{\text{Ag}' | \text{Ag}' \not\sim \text{Ag}\}$, i.e. agents with precedence incomparable to the agent. Proof by Lemma 0.9.
   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 such that 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 in the immediate time-step from Condition 1 in, and the fact that all Ag have a safe backup plan action a_{bp} to choose from when Condition 2 holds, and will always be able to (and will) take an action from which it can avoid collision in future time steps.

■

F. Liveness Lemmas

Lemma 0.11: If the only $a \in Act_{Ag}$ for an agent Ag for which $O_{\text{destination reachability}}(s,a,u) = \texttt{T}$ and $O_{\text{forward progress}}(s,a,u) = \texttt{T}$ is an action such that: $\gamma_{\text{Ag}} \in \{\texttt{right-turn}, \texttt{left-turn}\}$ and the grid-point $s_f = \tau_{\text{Ag}}(s,a)$ 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 $\mathscr{F}_{\text{Ag}}(u,a_i)$ will never be set to $\texttt{T}$.
2) In accordance to the Action Selection Strategy, for Ag to take action a, all the oracles in the Agent 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:
   $O_{\text{traffic intersection lane-change}}(s,a,u) = \texttt{T}$
   $O_{\text{legal orientation}}(s,a,u) = \texttt{T}$, $O_{\text{static safety}}(s,a,u) = \texttt{T}$ and $O_{\text{traffic intersection clearance}}(s,a,u) = \texttt{T}$.
      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 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, $O_{\text{traffic light}}(s,a,u) = \texttt{T}$.
      ii) $O_{\text{unprotected left-turn}}(s,a,u)$ is vacuously true for a right-turn action.
      iii) Since $O_{\text{traffic intersection clearance}}(s,a,u) = \texttt{T}$ and by the safety proof -E, 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 $O_{\text{dynamic safety}}(s,a,u) = \texttt{T}$.
      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 -E, 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, $\square\lozenge O_{\text{unprotected left-turn}}(s,a,u)$, specifically when the traffic light switches from green to red and Ag has been waiting at traffic light.
iv) Thus, $\square\lozenge O_{\text{unprotected left-turn}}(s,a,u)$ after light turns from green to red.
v) Further, $O_{\text{unprotected left-turn}}(s,a,u) = \texttt{T}$ combined with $O_{\text{traffic intersection clearance}}(s,a,u) = \texttt{T}$ implies $O_{\text{dynamic safety}}(s,a,u) = \texttt{T}$.
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 Agent Profile will always eventually be satisfied, and Ag will take a such that $O_{\text{destination reachability}}(s,a,u) = \texttt{T}$ and $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.

■

Lemma 0.12: If the only $a \in Act_{\text{Ag}}$ for which $O_{\text{destination reachability}}(s,a,u) = \texttt{T}$ and $O_{\text{forward progress}}(s,a,u) = \texttt{T}$ is when a has $\gamma_{\text{Ag}} \in \{\texttt{right-lane change, left-lane change}\}$ and the grid-point(s) $\mathscr{G}(s,a)$ 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 $\text{Ag} \in \mathfrak{A}$ 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 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 Ag will always eventually be able to take a.
2) In accordance to the Action Selection Strategy, for Ag to take a is that 1) all the oracles in the agent profile must be simultaneously satisfied (so the action a is chosen over any other $a' \in Act_{\text{Ag}}$, 2) $\mathscr{F}_{\text{Ag}}(u,a_i) = 0$ 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: $O_{\text{traffic light}}(s,a,u) = \texttt{T}$, $O_{\text{traffic intersection lane-change}}(s,a,u) = \texttt{T}$, $O_{\text{unprotected left turn}}(s,a,u) = \texttt{T}$, $O_{\text{traffic intersection clearance}}(s,a,u)$, $O_{\text{static safety}}(s,a,u) = \texttt{T}$, $O_{\text{traffic orientation}}(s,a,u) = \texttt{T}$.
      i) The first four hold vacuously and the others hold by Assumption 6, and the last holds by Agent dynamics and the Road Network.
   b) $O_{\text{dynamic safety}}(s,a,u) = \texttt{T}$.
      i) By the definition road network $\mathfrak{R}$, agent dynamics in Section IV-A 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, $O_{\text{dynamic safety}}(s,a,u) = \texttt{T}$ whenever an agent is in this state.
4) In accordance to the action selection strategy, for Ag to take a it must be that $\mathscr{F}_{\text{Ag}}(u,a_i) = 0$, i.e. the max-yielding-flag-not-enough must not be set. Let us show this is always true.
   a) The only Ag' that can cause the $\mathscr{F}_{\text{Ag}}(u,a_i) = 1$ 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 $\mathscr{F}_{\text{Ag}}(u,a_i) = 0$.
5) In accordance to the action selection strategy, for Ag to take a is 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, $\mathscr{W}_{Ag} = 1$ and Ag will be able to take its forward action since all criteria are satisfied.
   b) Any $\text{Ag}' \in \mathscr{C}_{Ag}$ will be of equal or lower precedence than Ag.
   c) Any Ag' with the maximum number of tokens will move to its goal since $\mathscr{W}_{\text{Ag}} = 1$ and all the other criteria required for that agent to takes its action will be True.
   d) By Definition of the Action Selection Strategy in Section IV-G, 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 subgoal (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 $\gamma_{\text{Ag}} \in \{\texttt{left-lane change,right-lane change}\}$.
7) In accordance to the Action Selection Strategy, for Ag to take a is that 1) $W_{\text{Ag}} = 1$, 2) $\mathscr{F}_{\text{Ag}}(u,a_i) = 0$, i.e. the max-yielding-flag-not-enough must not be set and 3) all the oracles in the Agent Profile must be simultaneously satisfied.
8) Let us first consider when $W_{\text{Ag}} = 1$, then $\square W_{\text{Ag}}$ 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 $O_{\text{dynamic safety}}(s,a,u)$.
10) By the lemma assumption that all Ag' occupying grid points will always eventually take their respective forward progress actions, $\Box\Diamond O_{\text{dynamic safety}}(s,a,u)$.
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 $\mathscr{F}_{\text{Ag}}(u,a_i) = 0$.
13) Thus, all criteria for which Ag can take its forward progress action a will be simultaneously satisfied.
14) When W_{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 $\Box\Diamond O_{\text{dynamic safety}}(s,a,u)$.
    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 0.13: Let us consider a road segment $rs \in RS$ where there exist grid points $g \in \mathscr{S}_{\text{sinks}}$. Every $\text{Ag} \in rs$ will always eventually be able to take $a \in Act_{\text{Ag}}$ for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.

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 $\gamma_{Ag} \in \{\texttt{left-lane change, right-lane change}\}$, then Ag will take always eventually this action by Lemma 0.12.
    b) If a is such that $\gamma_{\text{Ag}} = \texttt{straight}$:
    c) In accordance to the Action Selection Strategy, for Ag to take a is that 1) all the oracles in the agent profile must be simultaneously satisfied (so the action a is chosen over any other $a' \in Act_{\text{Ag}}$, 2) W_{Ag} = 1.
    d) First, we show that all oracles in the agent 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 0.12
    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 $\forall \text{Ag}$ where $m_{\text{Ag}} = N$ always eventually take $a \in Act_{\text{Ag}}$ for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
3) Case $m = N+1$: Let us show $\forall \text{Ag}$ where $m_{\text{Ag}} = N+1$ always eventually take a for which $O_{\text{forward progress}} = \texttt{T}$.
    a) Any Ag for which $m_{\text{Ag}} > 1$ will always have an a where $\gamma_{\text{Ag}} = \texttt{straight}$ such that $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
    b) Thus, we show Ag always eventually will take $\gamma_{\text{Ag}} = \texttt{straight}$ such that $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
    c) W.l.o.g. let us consider Ag for which $m_{\text{Ag}} = N+1$.
    d) In accordance to the Action Selection Strategy, for Ag to take a is 1) W_{Ag} = 1 and 2) all the oracles in the agent 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 $\text{Ag}' \in \mathscr{C}_{\text{Ag}}$ 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 conflict cluster will have token count of 0.
    f) After the assignment $W_{\text{Ag}} = 1$, $\Box W_{\text{Ag}}$ 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 the oracles in the Agent Profile will always evaluate to T.
        i) The same arguments in Lemma 0.12.1 for all oracles except for $O_{\text{dynamic safety}}(s,a,u)$, where $\Box\Diamond O_{\text{dynamic safety}}(s,a,u) = \texttt{T}$ by the inductive Assumption 2.

■

Lemma 0.14: Let Ag be on a road segment $rs \in RS$, where RS is the set of nodes in the dependency road network dependency graph $\mathscr{G}_{\text{dep}}$. Let rs be a road segment for which $\forall rs' \in RS s.t. \exists e : (rs', rs)$, each road segment rs' has vacancies in the grid points where $\text{Ag} \in rs$ would occupy if it crossed the intersection (i.e. $s_f = \tau_{\text{Ag}}(s,a)$), we show Ag will always eventually take an action $a \in Act_{\text{Ag}}$ where

$O_{\text{progress oracle}}(s,a,u) = \texttt{T}$.

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 Ag will always eventually take a for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
   a) If the only a where $O_{\text{forward progress}} = \texttt{T}$ is such that $\gamma_{\text{Ag}} \in \{\texttt{left-turn, right-turn}\}$, proof by Lemma 0.11.
   b) If the only a where $O_{\text{forward progress}}(s,a,u) = \texttt{T}$ is such that $\gamma_{\text{Ag}} = \texttt{straight}$.
      i) In accordance to the Action Selection Strategy, for Ag to take a is that 1) all the oracles in the Agent 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) $O_{\text{unprotected left-turn}}(s,a,u) = \texttt{T}$, $O_{\text{traffic intersection lane-change}}(s,a,u) = \texttt{T}$, $O_{\text{static safety}}(s,a,u) = \texttt{T}$, $O_{\text{traffic intersection clearance}}(s,a,u) = \texttt{T}$ $O_{\text{legal orientation}}(s,a,u) = \texttt{T}$.
         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) $\square\lozenge O_{\text{traffic light}}(s,a,u)$ by Assumption 5.
         D) $O_{\text{dynamic obstacle}}(s,a,u) = \texttt{T}$ because by the safety proof, all Ag take $a \in Act_{\text{Ag}}$ that satisfy first top tiers of Agent 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 $O_{\text{traffic light}}(s,a,u) = \texttt{T}$, then it $O_{\text{dynamic obstacle}}(s,a,u) = \texttt{T}$ 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 $O_{\text{traffic intersection lane-change}}(s,a,u)$ according to Safety Proof in Section A-E.
   c) By the safety proof in -E, Ag will only take $a \in Act_{\text{Ag}}$ that satisfy the top two tiers of the Agent Profile, so Ag will not take an a where $\gamma_{\text{Ag}} \in \{\texttt{left-lane change,right-lane change}\}$ 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 $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
3) Case $m_{Ag} = N+1$: Let us show any Ag that is a longitudinal distance of $N+1$ from the destination, will always eventually take a for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
   a) Let us consider when Ag's only a such that $O_{\text{forward progress}}(s,a,u) = \texttt{T}$ is $\gamma_{\text{Ag}} \in \{\texttt{right-lane change,left-lane change}\}$.
   b) Although Ag may not have priority (since it does not have max tokens in its conflict cluster), any Ag that occupies grid points $\mathscr{G}(s,a,u)$ 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 0.12.
   e) For all other $a \in Act_{\text{Ag}}$ are actions for which $\gamma_{\text{Ag}} = \texttt{straight}$, and the and same arguments as in the proof of straight actions for rs with $g \in \mathscr{S}_{\text{sinks}}$ in 3 hold.

■

G. Liveness Proof

Theorem 0.15 (Liveness Under Sparse Traffic Conditions): Under the Sparse Traffic Assumption given by 6.2 and given all agents $\text{Ag} \in \mathfrak{A}$ in the quasi-simultaneous game select actions in accordance to the agent protocol specified in Section IV, liveness is guaranteed, i.e. all $\text{Ag} \in \mathfrak{A}$ 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 the Appendix.
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 for some time window Δt_{tl} such that $t_{\min} < \Delta t_{\text{tl}} < \infty$, where $t_{\min}$ is defined in the Appendix in Section -H.1.
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 3.2) is a directed-acylcic-graph (DAG), we prove all $\text{Ag} \in \mathfrak{A}$ will always eventually take $a \in Act_{Ag}$ for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$ 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 $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
      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 0.13.
      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 in road segment where the linear order $l = N+1$ 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 -E, the clearance spots will be given precedence to $\text{Ag} \in rs$ for a positive, finite time and thus the assumptions required in Lemma 0.11 and 0.12 used to prove Lemma 0.14 will hold.
      vi) Thus, the Lemma 0.14 can be used to show that all Ag for which $l = N+1$ always eventually take an action for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
3) When the graph G_{dep} is cyclic, the Sparsity Assumption 6.2 can be used to prove all agents always eventually take an action for which $O_{\text{forward progress}}(s,a,u) = \texttt{T}$.
   a) The sparsity assumption 6.2 ensures 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 0.11, 0.12 and arguments in Lemma 0.14 1b. If the only forward progress action a makes Ag leave the loop, Ag will always eventually take its action by the sparsity assumption 6.2 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 0.12 hold. If the only action Ag can take is such that Ag crosses into an intersection, the traffic light rules ensure Ag has precedence over any Ag in the loop. Thus, Ag will always eventually take a forward progress action by Lemma 0.11 and Lemma 0.14 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 6.2 and the inductive arguments in the Liveness proof argument 2c.
4) By the induction arguments and by definition of the forward progress oracle $O_{\text{forward progress}}(s,a,u)$, all Ag will always eventually take actions that allow them to make progress to respective destinations, and liveness is guaranteed.

■

H. 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`.

1) 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 $\mathscr{F}_{\text{Ag}}(u,a_i) = \texttt{T}$ 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.

I. Simulation Maps

Fig. 11: Straight road map environment.

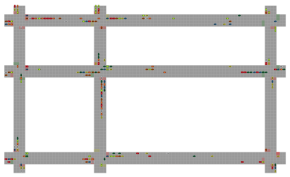

Fig. 12: City blocks map environment.

J. Simulation Environment Features

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 IV-D.

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 following attributes in our simulation: parameters like min and max velocity and accelerations, dynamics specified by agent actions and their corresponding occupancy grids, goal location, agent color, ID, token count. Note, these attributes can be modified depending on what the user wants to include. For each agent, a graph-planning algorithm is used to compute a high-level motion plan on the map graph to get the agent to its goal.

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.

RoSE also offers a collection of debugging tools to help reconstruct scenarios that occurred during a simulated game. If the user would like to regenerate the same initialization, the simulation has a feature where users can specify a specific randomization seed. There is a configuration tool that allows users to prescribe the states of a set of agents and their respective goals. A final debugging tool outputs the variables of the agent that were relevant to the decision-making process.