A Model-Based Framework for Using Non-Cooperative Multi-Agent Games to Explore Human-Based Uncertainty for Autonomous Vehicles

Anonymous Author(s)

ABSTRACT

Autonomous vehicles must operate safely in the face of uncertainty in addition to satisfying operational constraints, including those induced by human behaviors. Specifically, deployed autonomous vehicles (AVs) must exhibit safe responses when encountering previously unseen behavior(s) from human drivers with different driving styles. For example, aggressive drivers may cut off other vehicles to merge into a lane, or distracted drivers may fail to respond to changing road conditions. A key challenge is how to assess and/or improve the onboard AV decision-making capabilities to detect and mitigate those potentially unsafe scenarios due to one or more human-operated vehicles. We observe that the AV and the other vehicles on the roadway may share common functional objectives (e.g., to navigate to a given target destination), but otherwise may be motivated by different non-functional objectives, such as safety, minimizing transport time, minimizing fuel consumption, etc. This paper introduces a goal model-based, non-cooperative, multi-agent gaming framework to enable an AV developer to operationally assess and even improve the robustness of an AV. Specifically, the developer can systematically use the goal models to guide the exploration of the non-cooperative gaming behavior of the AV and the human-operated vehicle(s) against a spectrum of operating contexts. We use reinforcement learning and their reward policies to implement the goal-based specifications of the functional and non-functional goals of the AV and the human-operated vehicles, respectively. We demonstrate the model-based capabilities of our approach on a number of scenarios based on real-world traffic accident data involving human drivers.

ACM Reference Format:

Anonymous Author(s). 2024. A Model-Based Framework for Using Non-Cooperative Multi-Agent Games to Explore Human-Based Uncertainty for Autonomous Vehicles. In *Proceedings of ACM 27th International Conference on Model Driven Engineering Languages and Systems (MODELS).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Increasingly, autonomous vehicles (AVs) are being deployed in complex and dynamic environments, where they must safely interact with any number of road users with different behaviors, objectives, and driving styles. To ensure operational correctness and prevent failures, developers must understand how AVs respond to different sources of uncertainty, including the potentially unpredictable behaviors of human drivers. However, significant safety concerns and high resource costs limit "in the field" testing of safety-critical systems (i.e., AVs) in real-world settings. Thus, there is growing interest in the use of simulations to test AVs in representative operating contexts, including elements that pose uncertainty, such as other human-operated vehicles and road conditions [11]. This paper proposes a model-based framework to design goal-based vehicle agents and corresponding multi-agent simulation environments to assess and improve the robustness of AVs in response to human-based uncertainty.

Previous research in simulation-based testing has largely focused on sources of uncertainty that can be explicitly defined (e.g., weather conditions, static obstacles) [23, 35]. However, modeling human-based uncertainty is a challenging problem as human behavior cannot be explicitly identified and defined [8]. Recently, Chan *et al.* [9] proposed to combine non-cooperative game theory [33] and reinforcement learning (RL) [37] in order to operationally discover various types of human driver behavior(s) for a given driving style (e.g., aggressive driver). Commonly, model-based simulation for AV testing research largely use mathematical models to describe physics-based phenomena, and do not explicitly account for human-based behavior. RL-based AV simulation often relies on trial-and-error empirical techniques to determine reward functions. Furthermore, these complementary programming-based approaches to AV-related simulation typically require significant development efforts to accommodate any variations on the number and types of drivers, road scene changes, and other operational context changes.

This work introduces JanuSRL,[1] a goal model-based framework to support the systematic development and developer-in-the-loop exploration of how an AV under study may interact with one or more human-operated vehicles. Specifically, these vehicles operate according to human-based non-functional objectives (e.g., drivers in a hurry or overly cautious drivers) in intelligent transportation system (ITS) simulations. JanuSRL synergistically integrates a number of disparate modeling languages and machine learning components to model and operationally realize non-cooperative games between an AV and human-operated vehicles in an ITS simulation. Specifically, our work uses goal models to capture human driving styles and traffic accident data for an RL agent and explores its interactions with an AV in simulation. JanuSRL's developer-in-the-loop modeling approach enables developers to explore uncertainty in different traffic configurations (e.g., changes to functional/non-functional goals, number of agents, etc.) and simulations by revising goal models rather than low-level code.

JanuSRL innovatively integrates and uses two types of goal models to support the RL-driven simulation for a non-cooperative game approach to assess and improve the robustness of AVs in response to uncertainty posed by one or more human-based agents. First, we use the i^* modeling language [46] to capture the elements (i.e.,

[1] Janus is the name of the ancient Roman god associated with conflicts and choices.

MODELS, 2024, Austria
2024. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

agents) of the non-cooperative ITS game based on domain knowledge (e.g., AV, other vehicles, ITS roadways). In addition to specifying agent interactions, the i^* model specifies vehicle intentions, that is, high-level functional objectives (e.g., "arrive at destination") and non-functional objectives that characterize how it will be satisfied (e.g., "drive safely", "drive aggressively"). Second, for each agent in the i^* model, JanuSRL uses the KAOS goal modeling language [42] to refine its high-level functional objective via subgoal decomposition to obtain low-level functional objectives (leaf-level goals considered as requirements) that can be operationalized and assessed for satisfaction. We associate utility functions with KAOS requirements to assess their satisfaction [22], which is also used to inform the RL rewards for the corresponding agent. As such, JanuSRL integrates i^* and KAOS goal models to specify the behavior of ITS elements (AV and human-based), where utility functions for KAOS requirements are used to specify RL rewards, all of which is used to manage uncertainty discovery through non-cooperative games in RL simulation.

To demonstrate how JanuSRL can be used to (1) discover unexpected interactions between AVs and external agents in the environment and (2) improve the robustness of the AV with respect to human-based uncertainty, we have applied it to two different traffic use cases that capture real-world accidents. We show how JanuSRL models the descriptive, prescriptive, and predictive elements to combine non-cooperative games and RL using goal modeling languages. The remainder of this paper is organized as follows. Section 2 provides background information and overviews enabling technologies. Section 3 describes the JanuSRL framework. Section 4 shows the results of our case studies. Section 5 discusses our results, and Section 6 provides the threats to validity. Finally, Section 7 concludes this paper and discusses future work.

2 BACKGROUND

This section describes background topics and enabling technologies used in JanuSRL. First, we describe existing challenges for AVs. Next, we discuss existing state-of-the-art using non-cooperative game theory and RL to discover uncertainty for AVs. Finally, we overview the different types of models used in JanuSRL.

2.1 Challenges for Autonomous Vehicles

Advances in machine learning have improved the capabilities of self-driving vehicles. Recently, a number of different companies have deployed them in real-world settings, including Tesla's Autopilot [4], Waymo and Cruise's Taxi services [15], and ADASTEC's autonomous bus services [41]. AVs show promising results when deployed, leading to fewer and less severe accidents per mile driven than human driven-vehicles [10, 17]. To avoid accidents and protect its occupants, deployed AVs must demonstrate safe behavior in addition to satisfying operational constraints. However, various uncertainty factors, including those introduced by machine learning and other humans, have been shown to cause unexpected behaviors in AVs. For example, human drivers exhibit a spectrum of driving styles, including aggressive, distracted, or even malicious behaviors (e.g., brake-checking, tailgating, etc.). These driving styles lead to various uncertain driving maneuvers on the road, such as aggressively cutting off another vehicle or drifting out of a lane due to distractions. In order to ensure that AVs are capable of reacting safely when encountering these previously unseen maneuvers, AVs must be exposed to a wide range of human-based behaviors during training and testing.

2.2 Reinforcement Learning for uncertainty exploration

In order to ensure safe behaviors, a developer may employ a number of techniques to test the AV before deployment, including human-based testing in a simulation environment [16, 39, 49]. However, human-based testing faces the challenges of testing bias and incur expensive development time and effort [25]. As such, there has been a growing interest in complementing human-based testing with automated simulation testing techniques [47, 48]. Recently, Chan *et al.* [9] proposed the combination of non-cooperative game theory and RL to discover previously unseen or undesirable behaviors for AVs [9]. Non-cooperative game theory better captures the relationship between road users than cooperative game theory, as drivers are mainly concerned with their individual objectives and do not (typically) maliciously interact with other road users. RL is then used to operationalize the individual defined goals or driving styles in a simulation setting. This information enables developers to discover previously unseen interactions that may lead to undesirable behaviors, potentially revealing areas of improvement for the AV. However, existing simulation-based testing techniques may use ad hoc and/or hard-coded techniques to generate traffic scenarios, driving styles, and RL agents, which do not support a systematic approach to discovering human-induced uncertainty.

2.3 Goal-Oriented Requirements Engineering

This section describes the different modeling languages used in JanuSRL. JanuSRL uses two types of Goal-Oriented Requirements Engineering (GORE) models synergistically to abstract and systematically define the ITS and the goals of the agents under study. Compared to traditional modeling languages used in requirements engineering that focus on activities and information flows, models in GORE also reason about "why" the requirements are needed [43]. GORE models typically define a high-level goal or objective, further refining and clarifying them by decomposing them into subgoals through an incremental process.

i^* modeling notation. The i* modeling language [46] proposes an agent-oriented framework to model the relationships between agents of a system. Compared to other goal modeling languages, the i* modeling language places an emphasis on the "intentionality" of individual agents. By focusing on their individual intentions (i.e., motivations for functional objectives), developers can gain informative insights on the interactions and relationship between multiple parties, which may share and/or carry conflicting objectives. In an ITS, an i* model may specify the interactions between different agents (vehicles) in the environment. An *actor* defines a generic entity that contains one or more goals and/or provides resources for other agents, such as the ITS. High-level *agents* of a system are connected with others through dependency relationships. Each agent contains a number of *hard goals* and *soft goals*. Hard goals, or functional objectives, for agents may include '`complete the task within the time limit`', '`prevent accidents`',

and '`follow traffic laws`'. Soft goals, or non-functional objectives, may include '`drive safely`' and '`drive comfortably`'. *Resources* denote entities (physical or informational) that an agent may use, such as the '`availability of a lane`'. A *task* represents a specific activity that an agent may perform.

KAOS Goal Model. KAOS [42] is a goal modeling language that hierarchically decomposes high-level functional objectives into low-level system requirements. In KAOS, a high-level *goal* (i.e., '`drive safely`') may be refined using `AND` or `OR` decompositions into several *subgoals*, such as '`driving at the speed limit`' and '`maintaining a minimal trailing distance`'. At the leaf level, KAOS goals are considered requirements that are discharged to *system components* for satisfaction or satisficement (i.e., degree of satisfaction) [42].[2] Langford *et al.* [22] showed that *utility functions* can be used to measure the satisficement of KAOS goal models. Specifically, leaf-level subgoals are annotated with utility functions that specify how the corresponding system component can be used to measure a quantifiable attribute of the requirement. The utility function maps scalar values obtained from real-world sensors to the subgoals of the KAOS goal model in order to determine whether the measured value is within an acceptable range for the satisficement.

3 METHODOLOGY

This section overviews the JanuSRL framework, including its conceptual organization, aggregate models, and analysis processes.

3.1 Preliminaries

This section overviews the conceptual foundation for JanuSRL and introduces our running example. Table 1 provides an overview and definitions for the different technical terms used in this work.

Table 1: Overview of terminologies used in this paper.

Term	Definition
ITS	A transportation ecosystem comprising one or more intelligent vehicle(s) (e.g., AVs) as well as one or more human-operated vehicle(s).
Actor	A generic entity that has goals and/or provides resources (e.g., ITS).
Agents	Entity with a concrete role (e.g., vehicle) that has goals and is of interest in the ITS under study.†
AV	The ego autonomous vehicle under study.
Non-ego vehicle	Human-operated vehicle that is part of the operational context for the AV in the ITS.
Functional goal	Goals that describe functions that the system should perform (e.g., arrive at destination).
Non-functional goal	Goals that describe desired properties (e.g., defensive driving) of how a system satisfies its functional goals (e.g., arrive at destination).
Intention	An agent's observable behavior based on an aggregation of high-level goals, beliefs, tasks, motivations, and aim/desire to achieve their goals [45]. Intentions can comprise both functional and non-functional goals.

† The term *agent* is common to the disparate techniques used in this work (e.g., goal modeling, reinforcement learning, game theory). We use the definition presented in this table whenever the term *agent* is used.

3.1.1 MODA Reference Architecture. The modeling of socio-technical systems (e.g., ITSs) necessitates the use of different models and data to capture relevant information, contexts, and requirements. To this end, the *Models and Data* (MODA) framework provides a conceptual reference architecture for organizing different types of models and understanding the roles they play in scientific/engineering contexts [13]. Specifically, MODA describes three key roles that models can play: descriptive, prescriptive, and predictive. *Descriptive models* are used to document existing systems and/or phenomena to support analysis tasks and gain insight into the model's subject. For example, a descriptive model may describe specific types of driving styles that lead to result in traffic accidents. Next, *prescriptive models* provide a constructive means to design a system and/or address a problem. For example, a prescriptive model can specify an adaptive mechanism to mitigate environmental uncertainty identified by a descriptive model. As such, descriptive models often capture real-world phenomena and are then transformed into prescriptive models during the development process. Finally, *predictive models* are used to produce new information and/or knowledge about a target system or phenomena. For example, a simulation environment can play a predictive role if the results of a simulation yield new insights that better describe a real-world phenomena or guide decision-making during design.

In the context of the MODA conceptual framework, JanuSRL provides a concrete model-based realization of a non-cooperative gaming multi-agent ITS framework for discovering and mitigating human-based uncertainties. Figure 1 shows an overview of JanuSRL, where blue boxes depict model roles that are described in white rectangles, yellow rectangles represent modeling languages, orange rectangles represent RL components, and labeled arrows indicate relationships between modeling elements.

3.1.2 Gaming Setup. JanuSRL uses goal models to describe a *non-cooperative game* between vehicles in an ITS. While MODA is used to organize different types of models, non-cooperative game theory provides the foundation for the contents and structure of the models themselves. In game theory, a game involves a number of agents that interact with each other in an environment and seek to achieve individual and/or shared objective(s). In a cooperative game, agents can collaborate to maximize collective rewards. However, as AVs are deployed in real-world traffic and interact with traditional human-operated vehicles, it is not feasible to assume collaboration. To this end, we frame the ITS as a non-cooperative game where agents operate independently as vehicles may not be aware of other vehicle's intentions and do not explicitly coordinate actions [9]. Additionally, vehicles sharing the road may have common functional objectives (e.g., arrive at destination), where their ability to achieve these objectives may be impacted by the behavior of independent external vehicles. We use agent-based goal modeling to capture the agents that participate as players of the game (i.e., AV and non-ego vehicles), their intentions (e.g., aggressive, safety-focused, navigation), and their interactions (e.g., lane merging, intersecting paths). Then we use KAOS functional goal modeling to describe the functional objectives of each agent that participates as a player in the non-cooperative game.

3.1.3 Reinforcement Learning. RL is a class of machine learning that trains agents to take actions in an environment to maximize some notion of *reward* [37]. A reward function quantifies the resulting state of the environment after an action has been taken, based on the agent's objectives within a particular environment. RL models in JanuSRL are implemented as deep neural networks (DNNs) that represent control policies, approximating an optimal

[2] This work uses "system components" instead of "agents" in the KAOS goal model to avoid confusion with use of the term "agent" for the AV and non-ego vehicles.

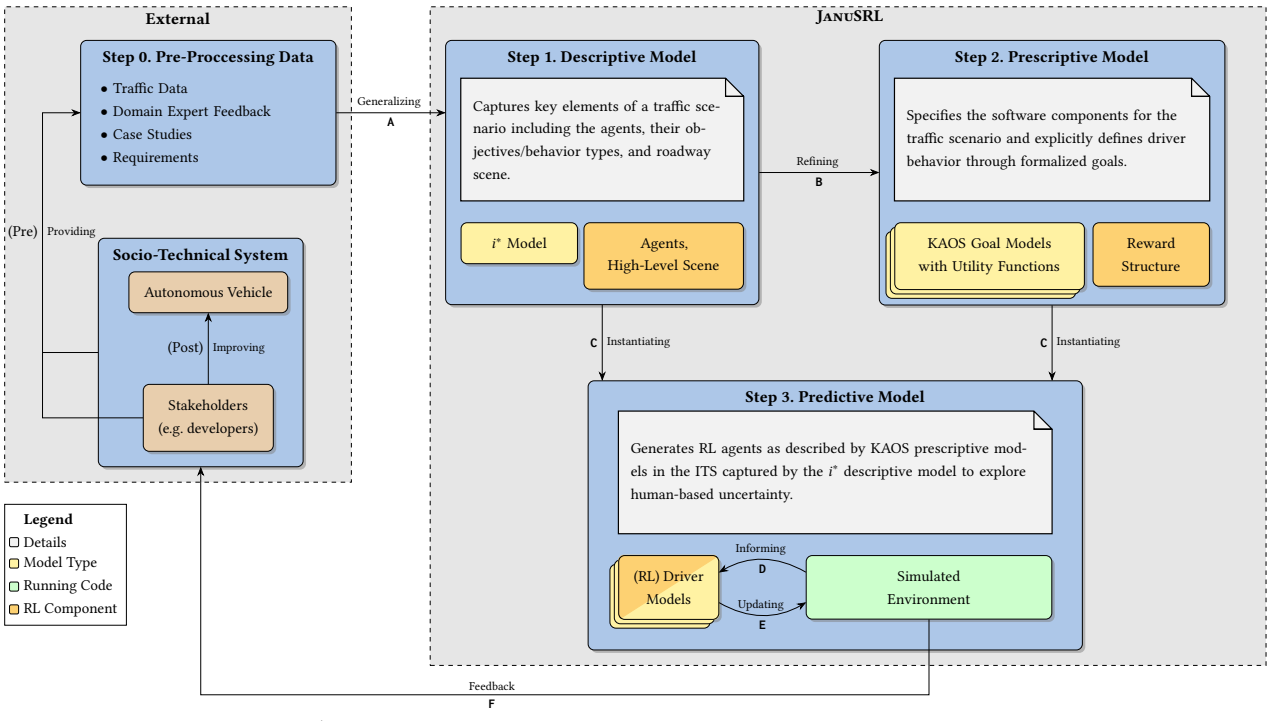


Figure 1: Overview of JanuSRL. White boxes show the description detail for a given step, yellow boxes indicate the model types for each step, orange boxes denote the corresponding RL component, and green boxes depict the simulation environment.

decision-making mechanism with respect to a reward function. For example, an RL algorithm training an AV to navigate to a destination might reward the AV based on the distance to the destination and penalize the agent for collisions or deviations from the optimal driving path [9]. In contrast with hard-coded rule-based agents where their behaviors are rigidly defined, we use RL models as a means to automatically discover different behaviors that satisfy high-level human-based non-functional goals (e.g., aggressive driving).

3.1.4 Running Example. We use a running example of a merge use case where a two-lane road converges to a single lane to help illustrate the JanuSRL framework. Figure 2 provides a 2D illustration of the ITS under study, where the AV (blue) is driving in the right lane, and an aggressive non-ego vehicle (orange) must merge into the AV's lane. We next describe details of each JanuSRL component in terms of MODA model categories.

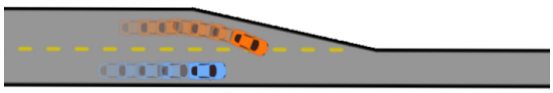


Figure 2: Merge scenario sample showing the AV (blue) and non-ego vehicle (orange) in the TinyRoads simulator [9].

Step 0. Pre-Processing

JanuSRL starts by *pre-processing* / aggregating historical data, documentation, and expert input regarding driving data and traffic accidents to identify different types of driver intentions. As such, JanuSRL can be used to explore human-based uncertainty found in real-world traffic scenarios. We assume that such information (e.g., proprietary data from insurance agencies, government reporting, in-house data monitors, etc.) is accessible to the stakeholders of an AV and/or ITS under study. In our running example, we opted to study a merge scenario, motivated by National Highway Traffic Safety Administration (NHSTA)'s 2022 accident reports [31, 32] that list lane merging as a frequent cause of accidents.

Step 1. Descriptive Model

The *descriptive modeling step* in JanuSRL captures high-level elements of a traffic scenario including agents, their intentions, and their interactions using the i^* modeling language [46] based on the aggregated information (arrow A *generalizing* in Figure 1). Figure 3 shows the i^* model for the merge traffic scenario, where circles represent agents, dashed circles encapsulate the behavior of a given agent, where clouds represent non-functional goals, ellipses represent functional goals, arrows with plus signs (+) represent contributions between goals, and arrows with half circles represent dependencies. This step captures the high-level traffic scenario and the agents in the non-cooperative game for RL (orange box of Step 1. in Figure 1). The i^* model for each agent is described by internal goals (enclosed in gray ellipse) comprising functional and non-functional goals, as well as their relationships. The i^* model does not include the operational details for achieving these goals, only the high-level operational context and driving styles (e.g.,

"safe", "rushed", etc.) that should be exhibited during satisfaction of the goals. In addition to modeling the AV and non-ego vehicles as agents, we also model the ITS as an actor that includes shared resources (e.g., main road, merge lane) with the vehicle agents, including global goals for vehicles making use of the ITS (e.g., avoid collisions, maximize throughput on the roadways).

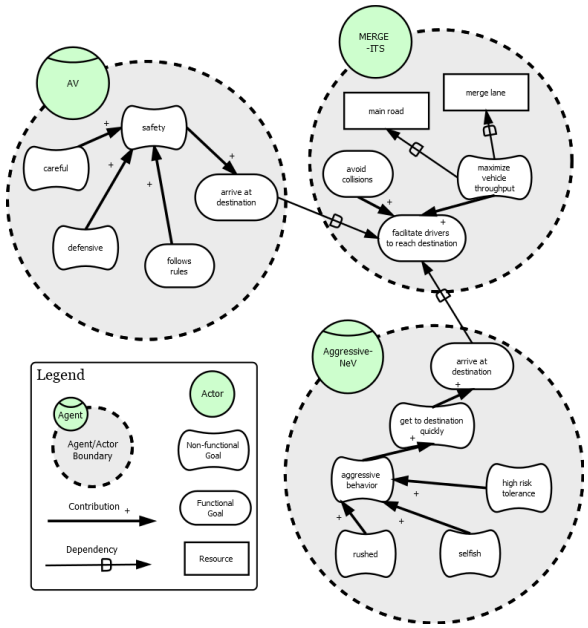


Figure 3: i^* model for the merge use case, showing AV and AGGRESSIVE-NEV agents, the MERGE-ITS actor, and their interactions.

Modeling Agent-Specific Goals. First, JANUSRL models each agent's internal functional and non-functional goals that represent a specific type of driving style (e.g., safety-aware driver, aggressive). For the **AV**, we consider the safety non-functional goal and three additional subgoals that contribute to safety: the functional goal of rule-following; and two non-functional goals of carefulness, and defensive driving. For the **AGGRESSIVE-NEV**, the non-functional goal '`aggressive behavior`' captures the aggressive driving style (we use `typeset text` with single quote delimiters to denote *goals* and SMALL-CAPS to denote agents in the ITS). We use real-world data [30, 31] to identify additional non-functional goals as contributors to the aggressive behavior non-functional goal, such as selfishness, high-risk tolerance, and rushed behavior. These described goals are internal to the agent and do not rely on ITS-specific context, and therefore can be reused to explore different ITS scenarios. As this modeling step is not dependent on any specific implementation-level details, input(s) from a broad range of domain experts (e.g., social sciences, humanities, civil engineers, etc.) may be aggregated to inform the analysis of a traffic scenario from the perspective of different stakeholders [46]. When compared with existing ad hoc development approaches that relied on low-level code changes to explore different traffic scenarios, the i^* high-level models enable the reusability of agent models, capturing agent intentions and internal goal structures that are applicable to different traffic scenarios.

Modeling ITS-Specific Goals. While the agent's internal goals (green circle and dashed gray circles) are not dependent on a specific ITS under study, the relationship between the agents and the ITS is captured through their interactions (connections) to the *ITS actor*. Notably, the ITS actor specifies the necessary information to design and instantiate the ITS simulation. In Figure 3, the **MERGE-ITS** actor is shown in the upper right green bubble and corresponding gray dashed circle (we use bold capital font to denote actors). In our running example, both the **AV**'s and **AGGRESSIVE-NEV**'s top-level functional goal '`arrive at destination`' depends on an *external* goal (i.e., goals outside of an agent's ellipse), namely the **MERGE-ITS** functional goal '`facilitate drivers to reach destination`'. The **MERGE-ITS** top-level functional goal has two contributing subgoals: functional subgoal '`avoid collisions`' and non-functional subgoal '`maximize vehicle throughput`', where the latter depends on two resources, the '`merge lane`' and the '`main road`'. The external goals may not be known by the agents (i.e., agents are not aware of the intentions of other agents), thus capturing the non-cooperative gaming aspect of the **MERGE-ITS**. By separating internal and external agent goals, agents can be reused in different traffic scenarios and developers need only to instantiate a new ITS actor (and corresponding connections to agents) to configure the ITS for a different traffic scenario.

Step 2. Prescriptive Model

We use the KAOS goal modeling language as a prescriptive modeling language to elaborate the functional objectives of the i^* agent goals. Figure 4 illustrates how we bridge the i^* high-level agent model with KAOS functional goal models (arrow B *refining* in Figure 1). Each KAOS model prescribes a specific functional subgoal decomposition that achieves a given agent's high-level functional objective (i.e., top-level goal) guided by its non-functional objectives (specified in the agent's i^* non-functional subgoals). Each KAOS model functional subgoal decomposition strategy is independent of that used for the KAOS models for other agents, which facilitates its reuse for different use cases requiring similar agent behavior.

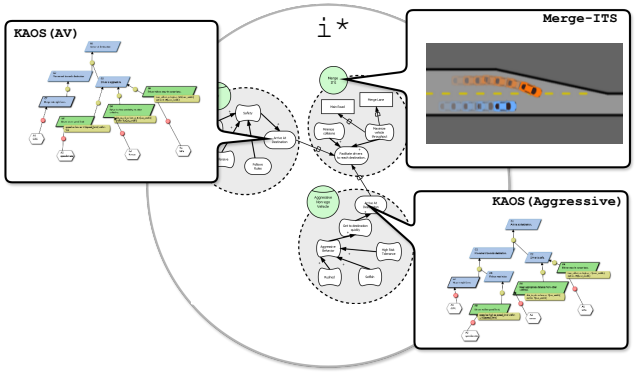


Figure 4: The relationship between i^* and KAOS goal models in JANUSRL.

KAOS Functional Goal Models. The behavior of each agent in the i^* model is elaborated in terms of a KAOS functional goal model [42]

that is hierarchically refined in terms of subgoals, eventually resulting in requirements that can be operationalized and assessed for satisfaction. Utility functions are used to annotate the leaf-level subgoals (i.e., requirements) and then are also used to specify reward functions for the RL model for each agent (orange box of Step 2. in Figure 1). Figure 5 and Figure 6 show sample KAOS goal models for the **AV** and **Aggressive-NeV**, respectively. *KAOS Goals* are depicted by parallelograms, leaf-level goals associated with utility functions are represented by green parallelograms, goal decomposition is represented by refinement arrows, and utility functions are represented by yellow ellipses, and elements in charge of discharging a particular leaf-level goal are represented by white hexagons.

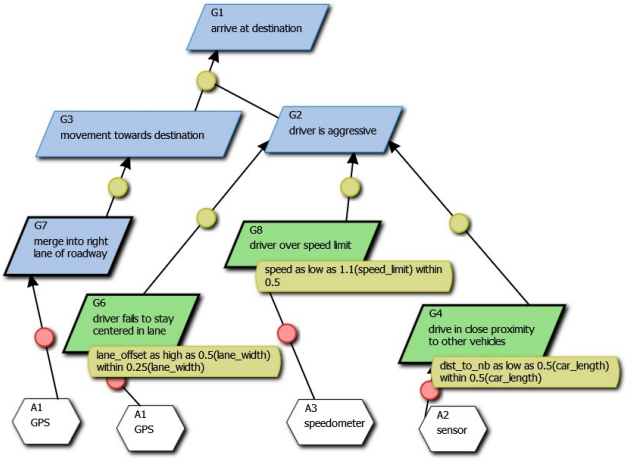


Figure 5: KAOS goal model for the AV.

KAOS Goal Model for the AV. For the **AV**, the top level goal G1 '`arrive at destination`' is OR- decomposed into the functional goal G2 '`move towards destination`' and agent-specific goal G3 '`safe driving`'. The safety goal G3 is further decomposed into subgoals. Associated with each leaf-level goal (i.e., requirement) is a utility function that quantifies its satisficement. For example, the functional goal '`keep appropriate distance from other vehicles`' (see G4, Figure 5) can be formally captured by Equation (1), where x corresponds to the distance from the nearest vehicle and y corresponds to the standard length of a car.

$$f(x) = \begin{cases} 0 & \text{if } x \leq 3.5y \\ 1 & \text{if } x >= 3y \\ \frac{3.5y - x}{0.5} & \text{if } 3y \leq x \leq 3.5y \end{cases} \quad (1)$$

KAOS Goal Model for Aggressive NeV. For the **Aggressive-NeV**, the top level goal G1 '`arrive at destination`' is OR-decomposed into the ITS-specific functional goal G2 '`move towards destination`' and agent-specific goal G3 '`aggressive driving`'. The aggressive driving goal G3 is further decomposed into quantified subgoals including speeding, swerving, and distance to vehicles. The prescriptive model provides the "middleware" between the descriptive model and the ITS simulation, enabling developers to formally realize high-level stakeholder input in terms of low-level functional goals. The utility functions corresponding to KAOS requirements can be used to inform the predictive modeling step, which we describe next.

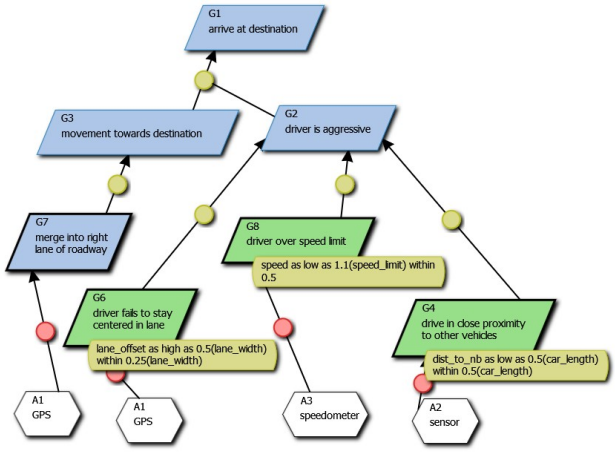


Figure 6: KAOS goal model for Aggressive-NeV.

Step 3. Predictive Models

The *predictive modeling step* integrates and operationalizes the i^* and KAOS goal models into a non-cooperative game to explore human-based uncertainty in an ITS simulation (arrows C *instantiating* in Figure 1). Figure 7 illustrates how the information from the goal models are used by JanuSRL to configure the non-cooperative game. First, the i^* `Codifier` processes i^* models to produce simulation platform-specific scenario configurations (see `YAML` [44] that are used to automatically configure and instantiate the simulation operating environment (e.g., road type, number of lanes, and types of lanes). Next, for each agent processed by the i^* `Codifier`, the `Utility Function Extractor` automatically extracts utility functions from the associated KAOS goal model for the `RL Configurator` to create the corresponding reward functions (i.e., the function that maps the utility functions to specific observable properties of the simulation platform) and instantiate RL models. The simulation environment asynchronously trains the respective RL agents to exhibit specific driving styles, where RL agents update the environment's state and the environment informs the RL learning process (arrow D *informing* and arrow E *updating* in Figure 1). As such, Step 3. of JanuSRL bridges the gap between previous modeling steps and the non-cooperative ITS game using RL and the simulation environment.

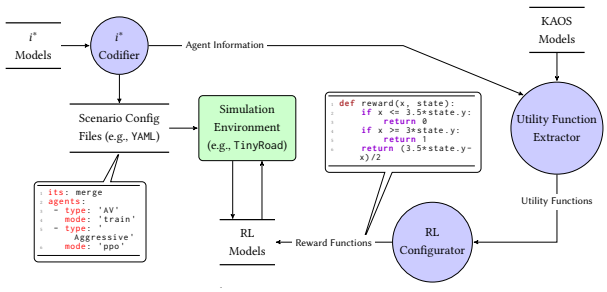


Figure 7: Overview of data flow from the i^* and KAOS models to the simulation environment.

Discovering Behavior. This section describes how JANUSRL uses an asynchronous non-cooperative game setup to discover uncertainty. The non-cooperative game contains an *initialize* step and three *rounds* of games. We enumerate each round below and describe how models are integrated and enable non-cooperative gaming. *Training mode* describes RL agents that are currently learning from the simulation. *Executing mode* describes RL agents that use (execute) their previously learned strategies in the simulation.

Round 0: Initialize Game and AV: Round 0 trains a baseline AV to learn to perform the given driving tasks. The baseline AV is in training mode, where the non-ego vehicles are executing as standard waypoint-following external traffic vehicles (i.e., there is no RL involved for the non-ego vehicles). The roadway for the simulation environment is created based on the descriptions of the ITS actor in the i^* model. The AV agent learns to perform driving tasks by learning behavior that satisfy its function goals specified in the KAOS goal model that satisfy the non-functional and functional goals specified in its i^* model (as depicted in Figure 3).

Round 1: Train Non-Ego Vehicles: Round 1 trains the non-ego vehicle(s) to complete driving tasks while exhibiting a specific driving style in the presence of the AV (trained in *Round 0*). The baseline AV is in execution mode, and the non-ego vehicle(s) are in training mode. The non-ego vehicle(s) learn to complete their respective functional goal specified in the corresponding KAOS goal model while exhibiting a particular driving style as specified by their respective i^* model (e.g., aggressive, distracted, timid, etc.).

Round 2: Assess AV's Robustness: Round 2 assesses the ability of the AV (trained in *Round 0*) to operate correctly in the presence of the non-ego vehicle(s) (trained in *Round 1*). The baseline AV is in execution mode, and the non-ego vehicle(s) are in execution mode. Agents exhibit styles of driving behavior that achieve their respective top-level functional goal according to the prescribed KAOS functional goal model that satisfies its non-functional goals (specified in the respective agent's i^* model). The previously unseen interactions or uncertainty exhibited by the AV during the games inform the developer-in-the-loop of its robustness to the human-based uncertainty posed by the trained non-ego vehicle(s).

Round 3: Improve AV's Robustness: Round 3 retrains the AV (trained in the *Round 0*) to improve its robustness in the presence of the non-ego vehicles (trained in *Round 1*). The AV is in training mode and the non-ego vehicle(s) are in execution mode. The AV agent learns to complete the function goals specified in the KAOS goal model (same as *Round 0*) that satisfy the non-functional and functional goals specified in its i^* model in the presence of uncertainty posed by the non-ego vehicle(s) that have learned to exhibit specific driving styles specified by their corresponding goal models (same as *Round 1*). The result of this round is a retrained AV that can move robustly and safely complete the given task in the presence of human-based uncertainty exhibited by the non-ego vehicle.

JANUSRL enables the flow of information from the different driver intentions found in real-world traffic data into an operational predictive model that yields new insights regarding the robustness of AVs (arrow F *feedback* in Figure 1).

4 DEMONSTRATION

To demonstrate use of the JANUSRL framework to discover unexpected human-based behavior and assess AV responses, we have applied it in two different proof-of-concept use cases. Our use cases include a weave lane scenario and an uncontrolled left turn intersection, both of which identify frequent traffic accident scenarios [1, 2, 28, 34]. For each study, we have followed the JANUSRL modeling and simulation steps to discover unexpected human-based behavior. We present results for the discovered human-based driving styles that cause the AV to fail, and then illustrate how developers can use these results to improve an AV's robustness in response to the human-based uncertainty.

Experiment Setup. Our demonstration is intended as a proof-of-concept study to illustrate the modeling capabilities of our framework. To this end, we use existing tooling for our experiments. First, we use the TINYROAD simulation platform [9] to provide a light-weight 2D traffic environment. TINYROAD uses a bicycle-kinematics physics model [21] and simple 2D graphics which require less computational overhead compared to more sophisticated and computationally expensive simulation platforms (e.g., CARLA [14], Gazebo [20]). Thus, the selected simulation platform supports quick training and evaluation of RL agents while preserving the key social/behavioral characteristics of AV and non-ego vehicle interactions. Next, we use Proximal Policy Optimization [37] and the OpenAI Gym [5] libraries to implement RL training and agents within the simulation. We refer readers to the work of Chan *et al.* [9] for further details on the use of RL within TINYROAD.

4.1 Use Case: Weave Lane

Our first case study explores a weave lane highway merge use case. Figure 8 provides a graphical depiction of the weave lane highway merge use case, where the AV (blue) is trailed by the non-ego vehicle (orange). Both vehicles must exit the on-ramp and merge onto the highway. A weave lane is a challenging highway merge scenario, as vehicles entering the highway often start at a low speed due to the entrance curve speed limit and can cause automated vehicles to fault [2]. For example, a 2020 Waymo AV exhibited unexpected behavior during a highway merge scenario where the AV failed to complete the merge maneuver [28]. We next detail how JANUSRL is used to model and explore specific human-based behaviors that may lead to vehicle failures in the weave lane scenario.

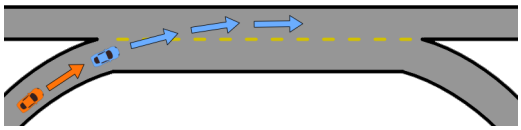


Figure 8: An overview of the weave lane highway merge use case.

Developing Goal Models. This section describes the development of i^* and KAOS goal models for the weave lane use case. Using historical data and/or domain expertise, we model the agents, interactions, and ITS setup in a *descriptive model* (Step 1.). For demonstration, we have opted to study an aggressive driver that is attempting to enter the highway in close proximity to the AV. As the internal objectives of both the AV and the aggressive non-ego vehicle are equivalent to those defined in our running example (see Figure 3), this experiment demonstrates how JANUSRL facilitates the reusability of models by using previously defined *internal* agent goal structure(s) in a new ITS setting. To connect these agents to the weave lane use case, we develop a **`WEAVE-ITS`** i^* actor. Figure 9 illustrates

how the two existing agent models (i.e., **AV** and **Aggressive-NeV**) are connected to the newly instantiated **WEAVE-ITS** (denoted by the red dashed circle). Both vehicles depend on the **WEAVE-ITS** goal of '`faciliate drivers to reach destination`', which in turn has two contributing subgoals: functional subgoal '`avoid collisions`' and non-functional subgoal '`maximize vehicle throughput`', where the latter subgoal depends on the resources of the '`weave lane`' and the '`highway`'. Thus, we have reused a majority of existing models to explore a different ITS operational context and need only to update a single actor (i.e., **WEAVE-ITS**) and its corresponding connections to existing agents (i.e., **AV** and **Aggressive-NeV**).

Next, we update the KAOS goal models for each agent for the weave lane use case (Step 2.). Similar to the process of updating the i^* model, we can reuse portions of the KAOS goal models (e.g., internal functional KAOS goals such as lane keeping, speeding, etc.) and update the ITS-specific goals (e.g., '`enter highway`') to reflect the objectives of the new **WEAVE-ITS**, rather than the **MERGE-ITS**. Next, to explore human-based uncertainty in a non-cooperative game, JanuSRL uses the defined i^* and KAOS goal models to instantiate the simulation environment (Step 3.). The i^* model informs the roadway setup, as specified by the **WEAVE-ITS** actor. After instantiating the operating environment, we define each agent's reward function. We use KAOS goal models and corresponding utility functions to directly structure the reward of the weave lane. Specifically, JanuSRL traverses each agent's KAOS goal model and extracts utility functions that are then transformed into RL reward functions by mapping variables from the utility function (e.g., '`speed_limit`' from goal G8, Figure 5) to observable properties of the specific simulation platform. The aggregate reward functions (i.e., representing utility functions from functional goals G8, G4, G6, Figure 5) inform RL learning agents (e.g., **AV**) to select for behaviors that increase the satisficement of the corresponding functional KAOS goal.

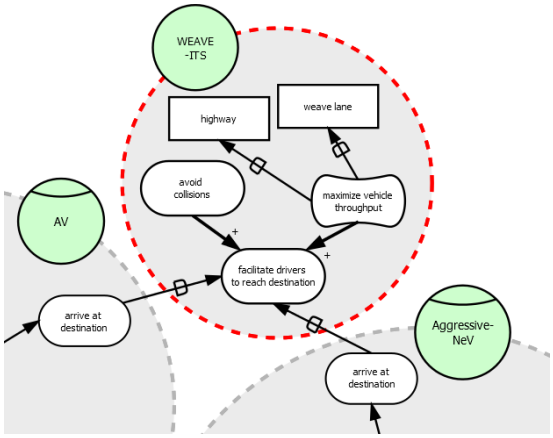
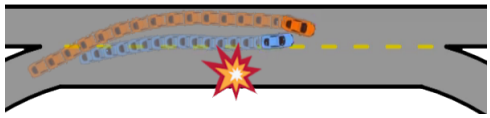
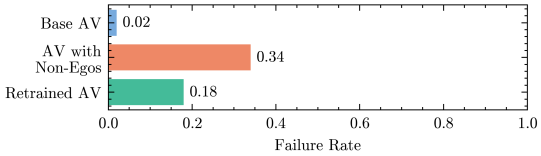


Figure 9: Overview of the weave lane use case illustrating the reuse of two existing agent models and their respective connections to the newly instantiated WEAVE-ITS actor, depicted by a red dashed circle.

Discovering Human-Based Behavior. Figure 10a illustrates an AV failure observed during the analysis phase (Step 4). In the presence of the non-ego vehicle, the AV attempted an early merge maneuver onto the highway to maintain a safe distance away from the aggressive non-ego vehicle. However, the AV failed to account for the trajectory of the non-ego vehicle, resulting in an increased failure rate when both vehicles merged in close proximity to each other. Figure 10b shows a comparison between the average AV failure rate over 100 random episodes. In Round 0, the baseline AV is able to complete the highway merge in the presence of external rule-following traffic with a failure rate of only 2.0%. However, when exposed to the aggressive non-ego vehicle (Round 2), the AV's failure rate increases to 34.0%. After retraining (Round 3), the AV learned to reduce its speed during the highway merge maneuver, thereby reducing its failure rate to 18.0%. Specifically, after retraining, the AV learned to delay the merging maneuver to allow the aggressive driver to pass at a safe distance. This demonstration illustrated how JanuSRL was able to reuse existing behavior models and apply them to a new use case to discover dangerous human behavior and unexpected AV responses. Then JanuSRL used this information to retrain the AV, resulting in a reduced failure rate.



(a) Weave lane failure example



(b) Weave lane failure rate

Figure 10: Example failure and failure rate for weave lane use case.

4.2 Use Case: Left Turn

To further demonstrate how JanuSRL supports reusability and extendability for exploring human driving styles in ITS simulation, we have applied it to a different use case (i.e., a left-turn intersection) that considers a different number of agents including a new type of agent. Figure 11 provides an overview of the left-turn use case considered, where the AV (blue) seeks to make a left turn and non-ego vehicles (orange and green) are traveling in the left lane (towards the AV). According to the NHTSA, a left-turn maneuver is one of the most frequent pre-crash events [12], especially at interactions without controlled signals [1, 34]. Therefore, it is of utmost importance to ensure that AVs are able to safely handle this maneuver, especially in the presence of different types of drivers. We next describe how developers can use JanuSRL to model and explore unexpected human-based driving styles in a left-turn scenario.

Goal Modeling. This section describes the extension and development of i^* and KAOS goal models for the left-turn use case. First, we specify the agents, the ITS actor, and their corresponding interactions in the i^* model as shown in Figure 12. Similar to the previous use case, we instantiate a new ITS actor for the left-turn use case as the **LEFT-TURN-ITS** actor, shown in the red

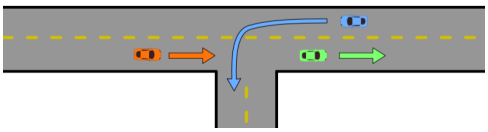


Figure 11: An overview of the left turn use case. The AV (blue) attempts to make a left turn, while the non-ego vehicles (orange and green) is driving in the right lane.

dashed circle. In addition to the **AV** and the **Aggressive-NeV** from the previous study, this experiment demonstrates JanuSRL's capabilities to extend the ITS with a new agent, the **Distracted-NeV**, denoted in teal. We define a **Distracted-NeV** with non-functional goals '`inattentive`' and '`inconsistent`' (i.e., they may drive slowly during distracted periods followed by an overcorrection to drive fast). Additionally, we include a means-end relationship where the task of '`checking a phone`' is a means to achieve the '`inattentive`' non-functional subgoal. Figure 13 shows the corresponding KAOS model for the **Distracted-NeV**, where the top level goal of '`arrive at destination`' is decomposed into subgoals such as '`driver enters distracted state`', '`lane swerving`', and '`operating in close proximity to other vehicles`'. Rather than hard-coding reward functions, JanuSRL automatically aggregates the utility functions from the **Distracted-NeV**'s KAOS goal model in order to instantiate corresponding reward functions. For example, the reward function corresponding to the low-level subgoal '`lane swerving`' (see G6, Figure 13) quantifies the **Distracted-NeV**'s satisficement of the functional KAOS goal for a specific simulation platform and encourages the distracted non-ego vehicle to exhibit a driving style specified in its corresponding i^* agent model. We describe the results for the left-turn use case next.

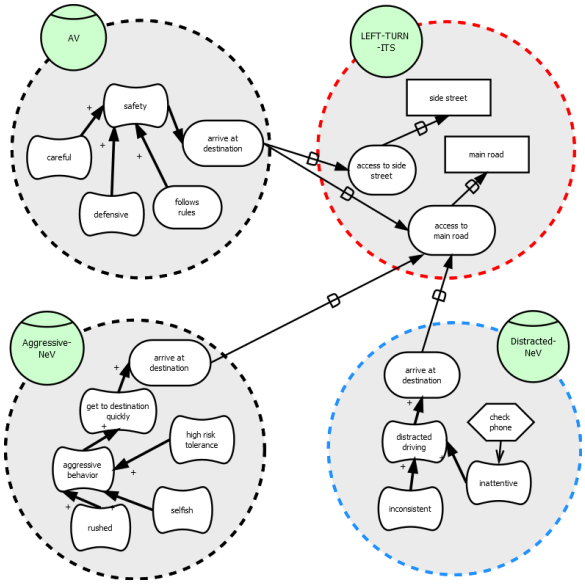


Figure 12: Diagram for the left-turn use case, where the red and teal dashed circles denote new elements to be instantiated.

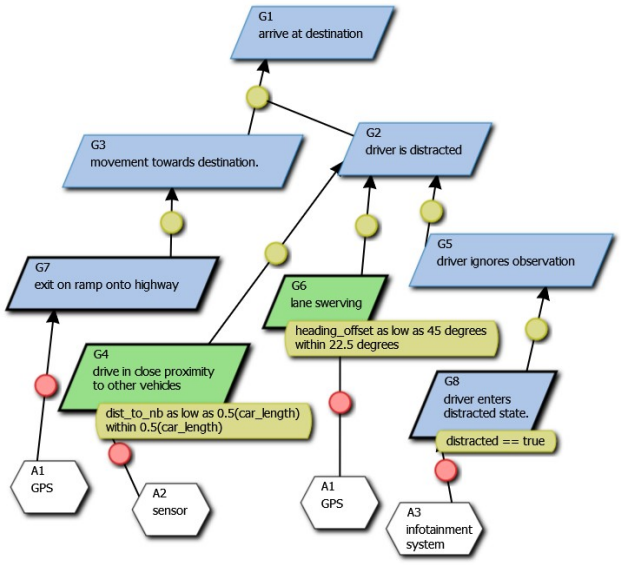


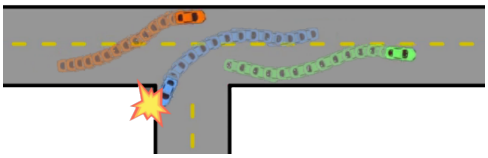
Figure 13: KAOS goal model for the Distracted-NeV.

Results. Figure 14a illustrates a failure scenario observed during the analysis phase (Step 4.) for the left-turn use case. The AV's initial attempt to make a left-turn maneuver is aborted due to the swerving behavior of the distracted non-ego vehicle. Then the AV attempts another left turn maneuver in close proximity to the aggressive non-ego vehicle, causing the non-ego vehicle to enter the AV's lane. The aggressive non-ego vehicle's behavior causes the AV to accelerate and collide with the road barrier, therefore resulting in an increased failure rate when the non-ego vehicles are in close proximity to the AV's lane. Figure 14b provides quantitative results for the left-turn case study. In Round 0, the baseline AV is able to complete the left turn in the presence of external rule-following traffic with a failure rate of only 1.0%. However, when exposed to the non-ego vehicles (Round 2), the AV's failure rate increases to 53.0%. After retraining (Round 3), the AV learned to reduce its speed during the left turn, thereby reducing its failure rate to 12.0%. This case study demonstrated how JanuSRL was able to reuse existing goal models, extend them with different number and types of drivers, apply them to a new ITS use case to discover unexpected human behavior with the corresponding unexpected AV responses, and then use this information to retrain the AV, resulting in a reduced failure rate.

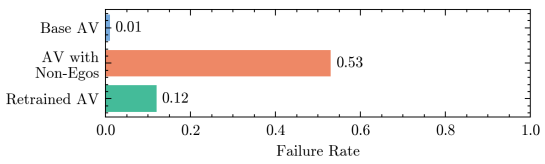
5 RELATED WORK

This section overviews related work relevant to JanuSRL. First, we discuss related work with non-cooperative game theory and RL. Next, we discuss relevant research for simulation-based testing. Finally, we overview other model-based approaches to RL, uncertainty discovery, or AVs.

A number of researchers have proposed game theory or RL-based approaches to train AV logic or improve the robustness of AVs against uncertainty. Liniger *et al.* [29] proposed a non-cooperative game theory approach to train agents for an autonomous racing



(a) Left-turn failure example



(b) Left-turn failure rate

Figure 14: Example failure and failure rate for left-turn use case.

games, but uses the same reward objective for agents (i.e., agents compete towards the same goal). Cao *et al.* [7] proposed an imitation learning approach to learn different types of driving models for an AV, and uses RL to learn to swap between the different learned policies. Li *et al.* [26] proposed a cooperative game-theoretic approach to model driver and vehicle interactions, but do not consider human-based uncertainty using a non-cooperative setting and have strictly defined reward functions for RL. Gupta *et al.* [18] introduced a framework to train two dueling agents in an RL setting, training a resilient ego vehicle against a (possibly adversarial) non-ego agent. Zhou *et al.* [50] proposed the UBRL framework, identifying potentially unreliable decisions of a DRL agent, but do not account for human-induced uncertainty. Chan *et al.* [9] demonstrated that RL and non-cooperative game theory can be synergistically combined to discover undesirable AV behaviors. However, existing work leverages an ad hoc development approach for RL, where their implementation and parameters are not systematically defined. Our work proposes a goal model-based framework to provide a systematic, goal-based, and reusable approach for uncertainty discovery using non-cooperative game theory and RL.

Other related work has explored testing AV behaviors in simulation. Dosovitskiy *et al.* [14], Son *et al.* [39], and Zhou *et al.* [49] proposed a number of simulation environments that can be used for AV testing. Birchler *et al.* [3], Zheng *et al.* [47], Zhong *et al.* [48] proposed several methods to generate test cases for an AV in simulation, but do not consider uncertainty from human-induced behaviors. Gambi *et al.* [16] combined procedural content generation and search-based testing to explore AV failure in various automatically generated traffic scenario settings. Stocco *et al.* [40] introduced ThirdEye, a white-box AV failure predictor using machine learning that periodically monitors the AV's reliability by identifying instances where the AV may be unconfident. While these approaches also seek to improve AV robustness, they do not address human-induced uncertainty and do not support a model-based approach.

Finally, other researchers have proposed model-based approaches for AVs or RL. Langford *et al.* [22] introduced MoDALAS, demonstrating a model-based approach to manage and verify the run-time assurance of machine learning components using KAOS goal models and utility functions. Liaskos *et al.* [27] proposed extending the i* framework to model a formal specification for Markov decision processes. In contrast, our approach leverages i* to model only the ITS and its (autonomous and human-based) agents at an organization level. Rudolph *et al.* [36] proposed a method to identify and consider strategies of RL agents in the presence of other players for self-adaptation. Jiang *et al.* [19] proposed the THGC model, which uses prior domain knowledge to group agents in a multi-agent reinforcement learning setting. Bruggner *et al.* [6] proposed a model-based approach to AV simulation, but models the different components of individual autonomous agents (i.e., perception, planning, and control). Leung *et al.* [24] introduced goal modeling for RL agents, where policies are represented as goals. However, their work does not use models to capture the high-level objectives of the agents. Schwan *et al.* [38] proposed a goal specification language to formalize a reward function for a given RL task (e.g., drive to destination). In contrast, our approach uses KAOS functional objectives to specify RL rewards as a means to train agents that exhibit specific driving styles, including those that are human-based.

6 THREATS TO VALIDITY

This paper shows an agent-based goal modeling approach to uncertainty discovery for AVs using non-cooperative theory and RL. There may be possible deviations between agent behaviors observed in simulation and reality (i.e., "reality gap"). This work uses RL to train the behavior of the base AV agent for evaluation as a proof-of-concept. As such, the base ego AV may not exhibit perfect behavior and reflect the exact behaviors of AVs deployed in a real-world setting. Future work may explore other simulation environments with proprietary AV software, which may better reflect the behavior of the AV system under study. Finally, repeated experiments may lead to or discover different types of agent behaviors or uncertainty, as RL relies on trial-and-error and non-determinism to train agents.

7 CONCLUSION

We introduced an agent-based goal modeling framework to provide a systematic process to discover unexpected behaviors or undesirable interactions and enable the robustification of the AV under study. Different use cases were provided to demonstrate that our approach can systematically refine high-level human driving styles into functional and non-functional goals. RL can then be used to operationalize the information, discovering failures in AVs and improving the robustness of the AV.

Future work will explore the training of multiple RL agents with additional driving styles (e.g., timid, intoxicated, etc.) simultaneously to discover unique strategies of agents and potentially reduce training time. Additional studies may consider different gradients of human driving styles (i.e., different levels of aggressiveness or speeding). Other research may investigate the use of procedural content generation to automatically create traffic scenarios for study. Finally, future work may include different types of environmental conditions (e.g., slippery road surfaces due to rain or snow, road obstacles, potholes, etc.) and explore their effect on the interaction of the ego and non-ego vehicles.

REFERENCES

[1] Types of Unsignalized Intersections - Unsignalized Intersection Improvement Guide. https://toolkits.ite.org/uiig/types.aspx.

[2] Thwarted on the On-ramp: Waymo Driverless Car Doesn't Feel the Urge to Merge. https://www.thetruthaboutcars.com/2018/05/thwarted-ramp-waymo-driverless-car-doesnt-feel-urge-merge/, May 2018.

[3] Birchler, C., Khatiri, S., Bosshard, B., Gambi, A., and Panichella, S. Machine learning-based test selection for simulation-based testing of self-driving cars software. *Empirical Software Engineering 28*, 3 (2023), 71.

[4] Bradley, R. Tesla Autopilot. https://www.technologyreview.com/technology/tesla-autopilot/.

[5] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).

[6] Bruggner, D., Hegde, A., Acerbo, F. S., Gulati, D., and Son, T. D. Model in the loop testing and validation of embedded autonomous driving algorithms. In *2021 IEEE Intelligent Vehicles Symposium (IV)* (2021), IEEE, pp. 136–141.

[7] Cao, Z., Biyik, E., Wang, W. Z., Raventos, A., Gaidon, A., Rosman, G., and Sadigh, D. Reinforcement learning based control of imitative policies for near-accident driving. *arXiv preprint arXiv:2007.00178* (2020).

[8] Carley, K. M. Social-behavioral simulation: Key challenges. *Social-Behavioral Modeling for Complex Systems* (2019), 741–752.

[9] Chan, K. H., Zilberman, S., Polanco, N., Siegel, J. E., and Cheng, B. H. C. SafeDriveRL: Combining Non-cooperative Game Theory with Reinforcement Learning to Explore and Mitigate Human-based Uncertainty for Autonomous Vehicles. In *Proceedings of the 19th International Conference on Adaptive and Self-Managing Systems (SEAMS 2024)* (2024).

[10] Channamallu, S., Kermanshachi, S., and Pamidimukkala, A. Impact of autonomous vehicles on traffic crashes in comparison with conventional vehicles. In *Transportation Safety and Emerging Technologies* (United States, 2023), H. Wei, Ed., International Conference on Transportation and Development 2023: Transportation Safety and Emerging Technologies - Selected Papers from the International Conference on Transportation and Development 2023, American Society of Civil Engineers (ASCE), pp. 39–50. Publisher Copyright: © ASCE.; International Conference on Transportation and Development 2023, ICTD 2023 ; Conference date: 14-06-2023 Through 17-06-2023.

[11] Chao, Q., Bi, H., Li, W., Mao, T., Wang, Z., Lin, M. C., and Deng, Z. A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 287–308.

[12] Choi, E.-H. Crash Factors in Intersection-Related Crashes: An On-Scene Perspective: (621942011-001), 2010.

[13] Combemale, B., Kienzle, J., Mussbacher, G., Ali, H., Amyot, D., Bagherzadeh, M., Batot, E., Bencomo, N., Benni, B., Bruel, J.-M., et al. A hitchhiker's guide to model-driven engineering for data-centric systems. *IEEE Software 38*, 4 (2020), 71–84.

[14] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. Carla: An open urban driving simulator. In *Conference on robot learning* (2017), PMLR, pp. 1–16.

[15] Freedman, I. G., Kim, E., and Muennig, P. A. Autonomous vehicles are cost-effective when used as taxis. *Injury epidemiology 5* (2018), 1–8.

[16] Gambi, A., Mueller, M., and Fraser, G. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis* (2019), pp. 318–328.

[17] Goodall, N. Comparability of automated vehicle crash databases. *arXiv preprint arXiv:2308.00645* (2023).

[18] Gupta, P., Coleman, D., and Siegel, J. E. Towards safer self-driving through great pain (physically adversarial intelligent networks). *arXiv preprint arXiv:2003.10662* (2020).

[19] Jiang, H., Shi, D., Xue, C., Wang, Y., Wang, G., and Zhang, Y. Multi-agent deep reinforcement learning with type-based hierarchical group communication. *Applied Intelligence 51* (2021), 5793–5808.

[20] Koenig, N., and Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)* (2004), vol. 3, IEEE, pp. 2149–2154.

[21] Kong, J., Pfeiffer, M., Schildbach, G., and Borrelli, F. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE intelligent vehicles symposium (IV)* (2015), IEEE, pp. 1094–1099.

[22] Langford, M. A., Chan, K. H., Fleck, J. E., McKinley, P. K., and Cheng, B. H. Modalas: Model-driven assurance for learning-enabled autonomous systems. In *2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS)* (2021), IEEE, pp. 182–193.

[23] Langford, M. A., and Cheng, B. H. C. Enki: A Diversity-driven Approach to Test and Train Robust Learning-enabled Systems. *ACM Transactions on Autonomous and Adaptive Systems 15*, 2 (June 2020), 1–32.

[24] Leung, J., Shen, Z., Zeng, Z., and Miao, C. Goal modelling for deep reinforcement learning agents. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part I 21* (2021), Springer, pp. 271–286.

[25] Leventhal, L. M., Teasley, B. M., Rohlman, D. S., and Instone, K. Positive test bias in software testing among professionals: A review. In *Human-Computer Interaction: Third International Conference, EWHCI'93 Moscow, Russia, August 3–7, 1993 Selected Papers 3* (1993), Springer, pp. 210–218.

[26] Li, N., Oyler, D. W., Zhang, M., Yildiz, Y., Kolmanovsky, I., and Girard, A. R. Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. *IEEE Transactions on control systems technology 26*, 5 (2017), 1782–1797.

[27] Liaskos, S., Khan, S. M., Golipour, R., and Mylopoulos, J. Towards goal-based generation of reinforcement learning domain simulations. In *iStar* (2022), pp. 22–28.

[28] Liberatore, S. Self-driving race car crashes into a wall from the starting line. https://www.dailymail.co.uk/sciencetech/article-8899021/Self-driving-race-car-crashes-straight-wall-starting-line-Roborace.html, Oct. 2020.

[29] Liniger, A., and Lygeros, J. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology 28*, 3 (2019), 884–897.

[30] Media, NHTSA. Distracted driving, 2021.

[31] Media, NHTSA. Driving behaviors reported for drivers and motorcycle operators involved in fatal crashes, 2021.

[32] Mergia, W. Y., Eustace, D., Chimba, D., and Qumsiyeh, M. Exploring factors contributing to injury severity at freeway merging and diverging locations in ohio. *Accident Analysis & Prevention 55* (2013), 202–210.

[33] Nash, J. F., et al. Non-cooperative games.

[34] National Highway Traffic Safety Administration. Query of fatality analysis reporting system (fars) web-based encyclopedia. https://www.nhtsa.gov/research-data/fatality-analysis-reporting-system-fars, 2014. Accessed: 2014-12-04.

[35] Pei, K., Cao, Y., Yang, J., and Jana, S. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles* (2017), pp. 1–18.

[36] Rudolph, S., Tomforde, S., and Hähner, J. On the detection of mutual influences and their consideration in reinforcement learning processes. *arXiv preprint arXiv:1905.04205* (2019).

[37] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms, Aug. 2017.

[38] Schwan, S., Klös, V., and Glesner, S. A goal-oriented specification language for reinforcement learning. In *International Conference on Modeling Decisions for Artificial Intelligence* (2023), Springer, pp. 169–180.

[39] Son, T. D., Bhave, A., and Van der Auweraer, H. Simulation-based testing framework for autonomous driving development. In *2019 IEEE International Conference on Mechatronics (ICM)* (2019), vol. 1, IEEE, pp. 576–583.

[40] Stocco, A., Nunes, P. J., d'Amorim, M., and Tonella, P. Thirdeye: Attention maps for safe autonomous driving systems. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (2022), pp. 1–12.

[41] Us, Y. Overcoming deployment hurdles: Adastec's approach to automated bus integration, oct 2023.

[42] Van Lamsweerde, A. Goal-oriented requirements engineering: A guided tour. In *Proceedings fifth ieee international symposium on requirements engineering* (2001), IEEE, pp. 249–262.

[43] Werneck, V. M. B., Oliveira, A. d. P. A., and do Prado Leite, J. C. S. Comparing gore frameworks: i-star and kaos. In *WER* (2009), Citeseer.

[44] YAML. Yaml: Yaml ain't markup language. https://yaml.org/.

[45] Yu, E. Agent-oriented modelling: software versus the world. In *International Workshop on Agent-Oriented Software Engineering* (2001), Springer, pp. 206–225.

[46] Yu, E. Modeling strategic relationships for process reengineering, 2010.

[47] Zheng, X., Liang, H., Yu, B., Li, B., Wang, S., and Chen, Z. Rapid generation of challenging simulation scenarios for autonomous vehicles based on adversarial test. In *2020 IEEE International Conference on Mechatronics and Automation (ICMA)* (2020), IEEE, pp. 1166–1172.

[48] Zhong, Z., Kaiser, G., and Ray, B. Neural network guided evolutionary fuzzing for finding traffic violations of autonomous vehicles. *IEEE Transactions on Software Engineering* (2022).

[49] Zhou, L., Song, Y., Gao, Y., Yu, Z., Sodamin, M., Liu, H., Ma, L., Liu, L., Liu, H., Liu, Y., et al. Garchingsim: An autonomous driving simulator with photorealistic scenes and minimalist workflow. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)* (2023), IEEE, pp. 4227–4232.

[50] Zhou, W., Cao, Z., Deng, N., Jiang, K., and Yang, D. Identify, estimate and bound the uncertainty of reinforcement learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems 24*, 8 (2023), 7932–7942.