Quantifying Faulty Assumptions in Heterogeneous Multi-Agent Systems*

Steven Carr [1] Tichakorn Wongpiromsarn[2] Ufuk Topcu [1]

***Abstract*— We study the problem of analyzing the effects of inconsistencies in perception, intent prediction, and decision making among interacting agents. When accounting for these effects, planning is akin to synthesizing policies in uncertain and potentially partially-observable environments. We consider the case where each agent, in an effort to avoid a difficult planning problem, does not consider the inconsistencies with other agents when computing its policy. In particular, each agent assumes that other agents compute their policies in the same way as it does, i.e., with the same objective and based on the same system model. While finding policies on the composed system model, which accounts for the agent interactions, scales exponentially, we efficiently provide quantifiable performance metrics in the form of deltas in the probability of satisfying a given specification. We showcase our approach using two realistic autonomous vehicle case-studies and implement it in an autonomous vehicle simulator.**

I. INTRODUCTION

Autonomous vehicles offer great promises for improving transportation safety and efficiency [1], [2], [3]. However, with the current technology, they tend to be more conservative than the average human driver, leading to instances of confusion and frustration of human drivers when encountering an autonomous vehicle. For example, they may lose patience when an autonomous vehicle precautiously slows down for potential jaywalkers and try to overtake it dangerously. Additionally, slowing down could be misinterpreted by the potential jaywalkers as giving them the right of way; thus, encouraging them to cross the road. As a result, even though autonomous vehicles may directly cause fewer accidents than human drivers, their conservative behaviors and inconsistencies with local driving practices potentially lead to more accidents in the overall transportation networks since they induce more risky behaviors by other road users, see the scenarios presented in [4] for additional information.

In their early adoption, autonomous vehicles with different levels of autonomy and capabilities are likely to share the road with human drivers. This sharing problem results in a transportation network where agents with different levels of autonomy exhibit different behaviors under the same situation due to differences in perception, intent prediction and decision making. These inconsistencies may cause confusion among agents, as evidenced by many reported accidents where autonomous vehicles were hit from behind [5], [6].

While recent efforts have been devoted to formalizing specifications of autonomous vehicles [7], [8], [9], the result of handling the inconsistency with human driving behaviors is not well understood. Further, existing approaches for handling both decision-making [10], [11], and incomplete perception [12] model the uncertainty in the environment as partially observable Markov decision processes (POMDPs) where the agents make decisions based upon their observations of these effects. In practice, under these approaches it is not only difficult to find tractable solutions but they also focus on single agents interacting with uncontrollable environments. Even the task of verifying that the behavior induced by a given policy on a POMDP model satisfies a temporal logic specification is non-trivial [13].

We are interested in the local interaction of agents, which involves coordinated planning across a system. When the agents coordinate, the planning problem can be expressed as the composition of subsystems. While each vehicle is capable of making decisions independent of the others' choices, they are often required to act in a manner that takes into account the joint action space [14]. If we were to consider a composed system under the aforementioned approaches, the natural formal model is a decentralized POMDP. For even two agents, planning in such a framework is NEXP-complete [15] and thus completely intractable for any environment.

Therefore for agents to effectively plan when accounting for local interactions, they must make some planning assumptions. With two assumptions, agents can reduce the planning problem from a decentralized POMDP (DEC-POMDP) model to a joint, multiple-agent uncertain MDP, which can be formalized as a single uncertain MDP (uMDP) whose complexity is NP-hard [16].

First, each agent assumes that the other agents compute their policies based on the similar objectives (e.g., satisfying the *rule of the road*) and based on the same model of the system. In doing so, they can eliminate the need to account for decentralized planning. This assumption is not unrealistic since the objectives of the vehicles typically correspond to obeying the traffic rules and reaching the destination [8]. While not all the road users follow the rules, it is expected that the majority do, especially for the rules that are safety critical such as avoiding collisions, staying in lane, stopping at a stop line, and giving way to a pedestrian at a crosswalk. Additionally, even though most vehicles have different final destinations, those that are traveling in the same direction typically share the same local destination (e.g., reaching the other side of the crosswalk, following the road until the upcoming intersection, etc). When agents make behavioral decisions, they are typically based on these local destinations, rather than the final destinations, which are too far for the

*NSF 2211141, NSF 2211432 & NSF 1652113 supported this work.

[1] The University of Texas at Austin
[2] Iowa State University

agents to anticipate how the world will look like.

Second, we assume that each agent makes a decision believing that the other agents are subject to the same uncertainties. This assumption arises naturally from the fact that each agent cannot access the perception of the other agents or the true state of the surroundings. In such uncertain situations, the best each agent can do is to assume that the other agents see the world in the same way that it does. For example, numerous accidents involving autonomous vehicles being hit from behind are caused by this assumption: human drivers did not anticipate abrupt braking in autonomous vehicles as they see the road as empty [5], [6]. When the agent assumes that environmental uncertainties at different states depend on each other, the vehicle can compute a conservative policy for a uMDP that maximizes the probability of reaching a safe state [17].

Regardless of the synthesis method, a policy, when applied to the true instantiation for the uncertain MDP model, induces a Markov chain which is efficiently verifiable using probabilistic model checking tools [18]. However, these assumptions come at a significant cost: namely if the agents do not agree on the fixed value of the parameter and are unable to communicate it to each other, the result contradicts policies that may lead to the agents to unsafe states.

This paper serves as an initial step to provide a quantitative analysis of the effect of inconsistencies in perception, intent prediction, and decision making among different agents on the overall system. In particular, we propose a formal approach to analyzing the application of policies on the composed system and separately verifying the induced Markov chain. We demonstrate that the associated *cost*, or difference in the probability of satisfying a given specifications, of a faulty assumption largely depends on the interactions. Further, we provide detail on the impact of faulty assumptions in a realistic simulation environment.

A. Related Work

Existing literature on interacting heterogeneous agents largely focuses on the synthesis of policies. For example, [19] formulate the problem as a product MDP and propose an incremental approach that successively adds agents to the planning problems up to a computational threshold; [20] synthesize a policy for an agent in a two-player stochastic game; and [21] decouple the sub-systems based on objectives and perform compositional reactive synthesis using maximally permissive policies. However, none of these approaches address potential assumptions made about the behavior or intent of interacting agents.

Approaches to incorporate intent from interacting agents, such as that of human pedestrians [22] or human drivers [23], rely on partially observable models, which rapidly become intractable as the number of agents increase. Robust synthesis methods for uncertain environments, such as [24] and [18], can tractably handle many agents but do not investigate the circumstances when the agents disagree on the uncertainty in their environments.

B. Running Example

Example 1. For the purpose of concretizing the problem setup, consider a simple car following scenario with a potential pedestrian transitioning a crosswalk shown in Fig. 1. The system consists of two heterogeneous agents ($\mathcal{A}_{back}$ and $\mathcal{A}_{front}$), one of which may be human-driven, and a stochastically moving pedestrian that will move into and out of the pedestrian crossing with probability $p \in [0, 1]$. For every location, each agent has only two action choices (Go or $Stop$). In this environment, the trailing vehicle ($\mathcal{A}_{back}$) may not pass the leading vehicle ($\mathcal{A}_{front}$), and the two agents may not occupy the same location without crashing. The collective objective is for both agents to successfully transit the crosswalk at location x_3 without occupying the same location as the pedestrian.

C. Contributions

The key contribution of the paper is a computational approach to quantitatively answer the two questions: (1) What price do we pay in the simplifying assumptions to make the planning problem computationally feasible? (2) Given the performance of the perception system of the autonomous vehicle, how safe is the overall system, when the autonomous vehicle interacts with other (possibly autonomous or human-driven) vehicles?

II. PRELIMINARIES

A *probability distribution* over a finite or countably infinite set X is a function $\mu\colon X \to [0, 1]$ with $\sum_{x\in X} \mu(x) = \mu(X) = 1$. The set of all distributions on X is $Distr(X)$.

Definition 1 (Markov decision process)**.** A (labeled) Markov decision process (MDP) M is a tuple $M = (S, A, s_0, T, \mathcal{AP}, L)$ with finite state S and action A sets, an initial state s_0, a transition function $T\colon S\times A \to Distr(S)$, a finite set of atomic propositions $\mathcal{AP}$ and a labeling function $L\colon S \to 2^{\mathcal{AP}}$ which assigns each state $s \in S$ a set of atomic propositions $L(s) \subseteq \mathcal{AP}$. We assume that the available actions are the same for every state $s \in S$. In doing so, we use the shorthand notation to describe the transition probabilities when taking action $a \in A$ as transition matrix $T^a \in \mathbb{R}^{n\times n}$. $\mathbb{T}$ is the set of all possible transition matrices. For more detail on labeled MDPs see [25].

A finite path h of an MDP M is a sequence of states; last(h) is the last state of h and the set of finite paths of

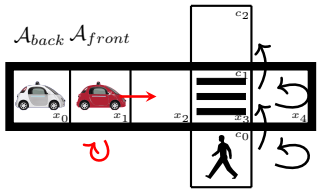


Fig. 1: Two cars on a straight street with a crosswalk. Agent $\mathcal{A}_{front}$ is in front of Agent $\mathcal{A}_{back}$. the composed objective is for $\mathcal{A}_{front}$ in state x_4 and agent $\mathcal{A}_{back}$ in state x_3 without colliding with both the pedestrian and each other.

M is Paths^M_{fin}. A Markov chain (MC) $\mathcal{D}$ is an MDP with $|A(s)| = 1$ for all $s \in S$.

Definition 2 (Uncertain MDP)**.** Let the transition matrix uncertainty set be defined as $\mathcal{T} \subseteq \mathbb{T}$, where every $T \in \mathcal{T}$ is a transition matrix. An uncertain MDP $\mathcal{M} = (S, A, s_0, \mathcal{T}, \mathcal{AP}, L)$ is a family of MDPs such that for every transition matrix T inside of the uncertainty set $\mathcal{T}$ is an MDP $M' = (S, A, s_0, T, \mathcal{AP}, L)$. Similarly, a given uncertain MC $\mathcal{D}$ may form a (finite) family of MCs $\mathbb{D}$, where $\mathcal{D} \in \mathbb{D}$.

Definition 3 (Policy)**.** A policy π for an MDP is a function $\pi\colon \mathsf{Paths}^M_{fin} \to A$ with $\pi(h) \in A(\text{last}(h))$ for all $\pi \in \mathsf{Paths}^M_{fin}$. Let Π^M be the finite set of all policies of M.

Definition 4 (Product MDP)**.** The product automaton of MDP M_1 and MDP M_2 is a labeled MDP $M_1||M_2 = (S, A, (s^1_0, s^2_0), T_{1,2}, \mathcal{AP}, L)$ with a finite composed state $S = S_1 \times S_2$ and action $A = A_1 \times A_2$ sets, a transition function $T_{1,2}\colon S_1 \times S_2 \times A_1 \times A_2 \to S_1 \times S_2$, a set of atomic propositions $\mathcal{AP} = \mathcal{AP}_1 \cup \mathcal{AP}_2$ and a labeling function $L\colon S_1 \times S_2 \to 2^{\mathcal{AP}_1 \cup \mathcal{AP}_2}$ which assigns each product state $s = (s^1, s^2) \in S$ a set of atomic propositions $L(s) \subseteq \mathcal{AP}$. A policy π_i for the composed system $M_1||M_2$ can be factored into two separate policies $\pi_{i|1}$ and $\pi_{i|2}$ that map $\pi_{i|1}\colon \mathsf{Paths}^M_{fin} \to A_1$ and $\pi_{i|2}\colon \mathsf{Paths}^M_{fin} \to A_2$.

Example 1 (continued)**.** In the simple environment of Fig. 1 each vehicle ($\mathcal{A}_{back}$ and $\mathcal{A}_{front}$) can be modeled as a two-action MDP (M_b and M_f) with initial states of x^b_0 and x^f_0 respectively. For each two-action MDP, a policy π maps the car's location in the street to an action choice $\{Go, Stop\}$. When we ignore the pedestrian, we can model the system as the product MDP $M_f||M_b$, where the composed state space $S = S_b \times S_f$ is a set of car position pairs for the street.

Definition 5 (Specifications)**.** We define a formal specification ψ using linear temporal logic (LTL), which concisely defines desired system behavior [26]. In this work, we make use of the LTL operator *until* U. A path h satisfies $(\psi_1 \mathsf{U}\, \psi_2)$ if there is a suffix of h that satisfies ψ_2 and all longer suffixes satisfy ψ_1. For full details on the LTL syntax and semantics, we refer the reader to [25].

Definition 6 (Induced Markov chain)**.** For an MDP M and a policy $\pi \in \Pi^M$, the MC induced by M and π is given by $M^\pi = (\mathsf{Paths}^M_{fin}, A, s_0, \mathcal{P}^\pi, \mathcal{AP}, L^\pi)$ where:

$$\mathcal{P}^\pi(h, h') = \begin{cases} \mathcal{P}(\text{last}(h), \pi(h), s') & if\ h' = h\pi(h)s', \\ 0 & otherwise, \end{cases}$$

and $L^\pi(h) = L(\text{last}(h))$.

Probabilistic model checking can be employed to formally verify quantitative properties of systems that exhibit probabilistic behavior [25]. In particular, given a system modeled by a Markov chain and an LTL specification, it can compute, in linear time, the probability that the system satisfies the specification. Probabilistic model checkers such as PRISM [27] and STORM [28] have been demonstrated to successfully analyze systems modeled by Markov chains with billions of states.

III. PROBLEM STATEMENT

Consider a set $\mathbb{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$ of decision-making agents. Each agent $\mathcal{A}_i$ is modeled by a Markov decision process M_i. The agents operate in an environment whose true model $\mathcal{D}_T$ belongs to the finite family of Markov chains $\mathbb{D}$. For each agent $\mathcal{A}_i$, let $\mu_i : \mathbb{D} \to [0, 1]$ be a probability distribution over $\mathbb{D}$ such that for any $\mathcal{D} \in \mathbb{D}$, $\mu_i(\mathcal{D})$ is the probability that $\mathcal{A}_i$ observes the environment as $\mathcal{D}$.

The objective of the agents is to maximize the probability that the complete system satisfies the specification ψ. However, they cannot communicate with each other and do not have access to the true model $\mathcal{D}_T$ of the environment. Additionally, they do not take into account the probability distributions $\mu_i, i \in \{1, \ldots, N\}$ when computing its policy. Instead, each agent $\mathcal{A}_i$ takes its observation as the true model of the environment and assumes that the other agent has the same observation. Formally, suppose agent $\mathcal{A}_i$ observes the environment as $\mathcal{D}_i \in \mathbb{D}$. The complete system, constructed based on $\mathcal{A}_i$'s observation, is given by $\mathcal{S}_i = M_1||M_2||\cdots||M_N||\mathcal{D}_i$. Let π_i be a policy of the complete system that maximizes the probability that $\mathcal{S}_i$ satisfies the specification ψ. The policy of $\mathcal{A}_i$ is given by the projection $\pi_{i|i}$ of π_i onto its action space.

Problem 1 (Analysis of the observation inconsistencies)**.** *Given the models $M_1, \ldots, M_N$ of all the decision-making agents, the probability distributions $\mu_1, \ldots, \mu_N$, the true model $\mathcal{D}_T$ of the environment, and the specification ψ, compute the probability that $\mathcal{S} \models \psi$, where $\mathcal{S} = M_1||M_2||\cdots||M_N||\mathcal{D}_T$ is the complete system, assuming that the policy of each agent is constructed as described in the previous paragraph.*

Example 1 (continued)**.** Recalling that the two vehicles ($\mathcal{A}_{back}$ and $\mathcal{A}_{front}$) can be modeled two-action MDPs (M_b and M_f), see Fig. 2 (left). When we include the pedestrian model, which moves stochastically at each time-step with a fixed probability $p \in [0, 1]$, with the family of Markov chains $\mathbb{D}$, see Fig. 2 (right). The true model $\mathcal{D}_T$ corresponds to the actual value of the pedestrian's movement probability p_T and thus the complete system $\mathcal{S} = M_{back}||M_{front}||\mathcal{D}_T$ is formed

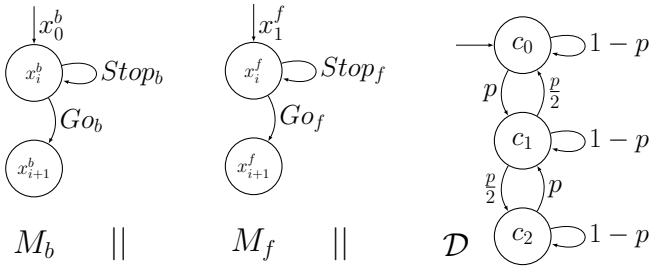


Fig. 2: The sub-models for the two car, one pedestrian crosswalk environment introduced in Fig. 1. The entire composed model is taken from the product $M_1||M_2||\mathcal{D}$.

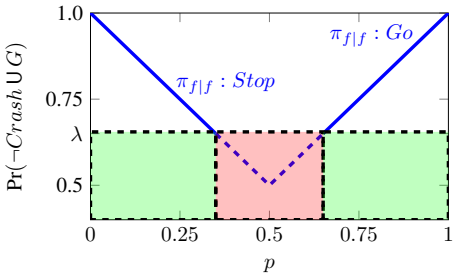


Fig. 3: For Example 2, the probability of satisfying $\neg Crash \,\mathsf{U}\, G$ for different policies at initial state (x_0^f, c_0) of agent $\mathcal{A}_{front}$ and obstacle $\mathcal{D}$. Policy π refers to the action taken at state (x_2^f, c_1), just before the crosswalk.

by the product of the three sub-models in Fig. 2. Meanwhile, the agents estimate the pedestrian probabilities p_b and p_f, according to distributions μ_b and μ_f respectively, which correspond to models $\mathcal{D}_b \in \mathbb{D}$ and $\mathcal{D}_f \in \mathbb{D}$. Additionally, they synthesize policies according to the systems $\mathcal{S}_b = M_b||M_f||\mathcal{D}_b$ and $\mathcal{S}_f = M_{back}||M_{front}||\mathcal{D}_f$

IV. POLICY ANALYSIS

In this section, we describe the process for performing analysis on the composed system whereby policies are implemented based upon the assumptions made by the agents.

A. Analysis for a Composed System

Consider a composed system $\mathcal{S}$ for set of decision-making agents $\mathbb{A}$ and a corresponding set of policies $\{\pi_1, \ldots, \pi_N\}$.

1) Policy Splitting: Agent $\mathcal{A}_i$'s policy π_i represents a composition of individual policies of all N agents: $\pi_i : ||_{j=1:N}\pi_{i|j}$ where $\pi_{i|j}$ is the projection of policy π_i under Agent $\mathcal{A}_i$'s assumption on the decisions of Agent $\mathcal{A}_j$. Agent $\mathcal{A}_i$ operates under the assumption that the other agents are also implementing π_i and therefore implements $\pi_{i|i}$. Each agent builds policies the same way and therefore, the implemented policy π can be described by $\pi : ||_{j=1:N}\pi_{j|j}$.

2) System Analysis: Finally, we analyze the probability $\Pr(\mathcal{S}_T^\pi \models \psi)$ that true composed system $\mathcal{S}_T$ satisfies specification ψ, which is dependent on the finite nature of the policies. By taking sample instantiations for each agent $\mathcal{A}_i$, according to μ_i and using probabilistic model checking [28], we generate a set of benchmarks or quantitative regions for parameter values similar to those shown in Fig. 3. Using these benchmarks, we determine the relevant assumptions to satisfy the specification and those with limited impact. An example is the inflexion point at $p = 0.5$ in Fig. 3, where an instantiation $p_i < 0.5$ has a different policy to instantiation $p_j > 0.5$. If we require the probability of the composed system is above $\lambda = 0.5$, then the policy satisfies this require if the parameter $p < 0.33$ or $p > 0.66$ Here, these benchmarks refer to both the inflexion points and also the parameter values that lead to the highest probability of satisfying policies.

B. Finite Policies for Large Model Sets

Example 2. First, consider a slight modified version of Example 1 with only a single car $\mathcal{A}_{front}$, which also attempts to transit a crosswalk without hitting the pedestrian.

For each simulation there exists a true model $\mathcal{D}_T$, which accurately describes the value of p, and one agent $\mathcal{A}_{front}$ makes an assumption about the value of p and plans according to their belief of the model $\mathcal{D}_i$. In this example, the set of pedestrian models $\mathbb{D}$ is uncountably infinite while for the agents there exists a finite number of deterministic policies. The agent's goal is to reach location x_4^f without crashing, which occurs if the car shares the same location as the pedestrian, which is state (x_3^f, c_1) in the composed system $M_f||\mathcal{D}_T$.

Now we consider when the vehicle attempts to successfully navigate the crosswalk with 0.65 probability, i.e. we seek to satisfy the specification $\psi_p = \Pr_{\geq\lambda}(\neg Crash \,\mathsf{U}\, G)$ where $\lambda = 0.65$. The policy that optimizes reachability is dependent on the value of parameter p. Here, the optimal policy takes two forms depending on one of two different action choices at critical state state (x_2^f, c_1), prior to crosswalk and obstacle at the edge. At state (x_2^f, c_0), the optimal policy requires selecting action Go_1 for the case where $p < 0.5$ and $Stop_1$ otherwise. Fig. 3, shows for which values of $p \in [0, 1]$ this policy satisfies the specification ψ. For any distribution μ, there exists a piecewise linear and convex (PWLC) function π that finds a locally optimal policy for maximizing probability of satisfying the specification [29]. Consequently, even though the number of obstacles models is large, the set of possible policies for this state is finite regardless of the value of p.

V. SYNTHESIS OF COMPOSED SYSTEM

The focus of this work is the verification framework, which is independent of any policy synthesis method. For comparative purposes, we present an example approach to multi-agent synthesis that operates under the described assumptions on agent behavior. While we can verify a given policy applied to a system in linear time [17], the policy synthesis problem is NP-Hard [30]. Example 2 describes a single agent composed with the obstacle model; we now describe the policy synthesis process for multiple agents.

Example 3. Consider agent $\mathcal{A}_{front}$ and its observation of the environment $\mathcal{D}_f \in \mathbb{D}$ according to μ_i. The state space for system $\mathcal{S}_f$ can be represented by a tuple of the vehicles'

Problem type (N,m)	States in Composed System	Time for Synthesis (s)	Time for Verification (s)
Crosswalk	63	2.11×10^{-3}	4.40×10^{-4}
Gridworld (2,3)	365	0.55	8.82×10^{-2}
Gridworld (2,4)	3352	4.52	0.59
Gridworld (2,5)	8672	23.95	1.11
Gridworld (2,6)	17541	603.95	37.39
Gridworld (3,4)	5188	453.34	3.62
Gridworld (3,5)	187522	5104.01	100.32

TABLE I: Average computation time for composition and verification of different sized environments.

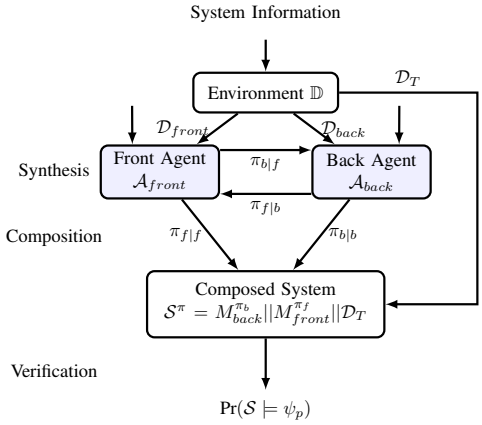


Fig. 4: Analysis process for agents performing synthesis.

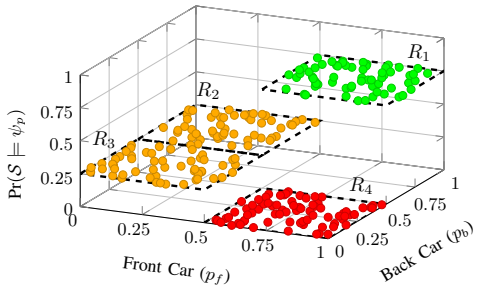


Fig. 5: Probability that system $\mathcal{S}$ with $p_T = 0.75$ for pedestrian model $\mathcal{D}_T$ satisfies the specification. 100 samples were taken, where μ was a uniform distribution for all agents. Verification of the composed systems reveals three level sets of probabilities that correspond to four regions under the policy assumptions $\{R_1, R_2, R_3, R_4\}$.

locations in their sub-models $(x_i^f, x_i^b, c_i) \in S_f \times S_b \times S_c$. We synthesize a policy π_f for the system $\mathcal{S}_f$ such that $\Pr_{max}(\mathcal{S}_f \models \psi_p)$. In doing so, agent $\mathcal{A}_{front}$ assumes that agent $\mathcal{A}_{back}$ chooses its action according to $\pi_{f|b}$. i.e. that the other agent observes the environment as $\mathcal{D}_f$. This process is repeated for the back agent $\mathcal{A}_{back}$ and environment observation $\mathcal{D}_b$ for policy $\pi_{b|f}$.

a) Extension to general number of agents: Similarly we can be extend Example 3 to the general set of agents $\mathbb{A}$, where for example agent $\mathcal{A}_1$ observes the environment as $\mathcal{D}_1$. Subsequently agent $\mathcal{A}_1$ assumes that every agent $\mathcal{A}_j \in \mathbb{A}$ makes choices according to $\pi_{1|j}$, i.e. that every other agent also observes the environment as $\mathcal{D}_1$. This process can than be repeated for every agent $\mathcal{A}_i \in \mathbb{A}$ and environment observation $\mathcal{D}_i \in \mathbb{D}$ until policies $\pi_{i|j}$ for all combinations of i and j are obtained.

b) Synthesis using model checking: For each agent $\mathcal{A}_i$, probablisitc model checkers such as STORM [28] or PRISM [27] can compute the policy π_i that maximizes the probability of satisfying the specification $\Pr_{max}(\mathcal{S}_i \models \psi)$. These tools compose the system $\mathcal{S}_i$ and specification ψ into an analyzable model, which can then reduce the synthesis problem to either a linear program or a binary decision diagram [31].

As described in Example 2, if one takes enough sample instantiations from the distributions μ_i and μ_j then we can deduce the locations for which the differences between synthesized policies π_i and π_j are suboptimal (red region in Fig. 3), for additional information see [17]. In this work, we show that this assumption-based approach to synthesis leads to sub-optimal outcomes. In Fig. 4, we describe the analysis algorithm. Each agent observes the environment, collecting a Markov chain $\mathcal{D}$, they then sythesizes a policy that feeds into the composed system $\mathcal{S}$. Finally, we verify the composed system $\mathcal{S}$ against the specification ψ_p.

VI. CASE STUDIES

In this section, we explore a set of case studies that quantify the effect of making faulty policy assumptions. The settings chosen range from a simple pedestrian crossing to a scalable gridworld environment with a general number of agents. To synthesize and evaluate policies, we implement a Python toolchain that employs the probabilistic model checker STORM [28]. We performed all analysis on a 1.9GHz machine with a 12GB memory limit. All code used to generate and verify policies can be found at `https://u-t-autonomous.github.io/heterogenous_assumptions/` .

a) Toolchain description: For each sample point in Fig. 5 and Fig. 7 (corresponding to an instantiation of the true system S_T), we instantiate N environments based on each agent's observations, which gives a set of N values for each parameter and a set of corresponding set of Markov chains $\{\mathcal{D}_1, \cdots, \mathcal{D}_N\} \subset \mathbb{D}$. In parallel, for each agent $\mathcal{A}_i \in \mathbb{A}$ and system S_i we synthesize a policy π_i using STORM's *sparse* engine [31], which formulates the MDP as a set of linear equations and finds the optimal action choices for maximizing the probability of satisfying the specification ψ. Once we have obtained a policy for each agent, we extract and apply each policy $\pi_{i|i}$ on the true system $\mathcal{S}_T$ by constructing MC $\mathcal{S}_T^{\pi}$. We then verify, again using STORM's model checker to provide a quantative performance metric on the policy assumptions made for system $\mathcal{S}_T$ in the form of the probability of satisfying the specification $\Pr(\mathcal{S}_T^{\pi} \models \psi)$.

b) Model checking and verification: We present the model sizes, average times for synthesis, and verification for both sets of case studies in Table I. Note that the synthesis method scales exponentially with the number of agents and is used to demonstrate the efficiency gained by the proposed approach to verification on the sampled distributions.

A. Pedestrian Crossing

Returning to the scenario presented in Fig. 1, the composed system has a goal state of $G : x_4^f \wedge x_3^b$ and additionally, a crash occurs if the agents occupy the same space or the same space as a pedestrian, given by $Crash : (x_3^b \wedge c_1) \vee (x_3^f \wedge c_1) \bigvee_{j=1}^{4} (x_j^b \wedge x_j^f)$. The vehicles attempt to find policies that satisfy $\psi_p = \neg Crash \cup G$. Each agent, makes an assumption of the pedestrian model p_b and p_f forming $\mathcal{D}_b$ and $\mathcal{D}_f$

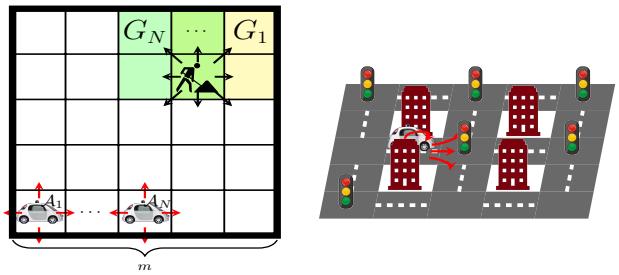


Fig. 6: City grid environment with m number of cars. Included are two examples of possible operating regions for the construction crew with the true region (yellow) and region shifted in x (green). Each grid location represents an intersection for the vehicle to navigate.

respectively. After making this assumption, the agents then synthesize a strategy for the composed system $M_b||M_f||D_i$.

a) Result: Fig. 5 shows the differing outcomes dependent on the agents' assumptions. As described in the one car case, there exists a policy inflexion point at $p = 0.5$ (see Fig. 3). Consequently, we can separate the policy space into quadrants based on the values of p_b and p_f. When both probabilities agree with the true model, the implemented policy maximizes the probability of satisfaction (see R_1 of Fig. 5. When the probabilities are inconsistent with the true model, a crash is not necessarily guaranteed – especially if the front agent A_{front} is less conservative, i.e. it assumes that $p_f < 0.5$, which means it will choose Go_f more frequently than $Stop_f$ (R_2 for example).

b) Simulated Environment: We simulate these policy constructions for different example instantiations on the open-source simulator CARLA [32]. We run CARLA 0.9.10 on a 3.1 GHz machine with a GeForce RTX2060 graphics and 32GB of memory. In CARLA, the car identifies the pedestrian waiting at the edge of the crosswalk but we assume that the sensors are not capable of perfectly observing the pedestrian's intent to cross the road. We simulate the car's observation of pedestrian intent from the distributions μ_b and μ_f, which reflect the simulated sensor uncertainty. The observations give two instantitions of $\mathcal{D}_f$ and $\mathcal{D}_b$, which we can efficiently and in real-time (see Crosswalk in Table I) synthesize policies using the STORM toolchain described earlier. We include an example run in the video attachment. In this example, the front car acts according to the belief that the pedestrian will not cross with high probability and therefore goes through the intersection. Meanwhile, the back car believes that the pedestrian will cross with high probability and therefore assumes the front car will stop. This run would constitute a sample taken in the region R_2.

c) Comparison to parametric synthesis: We can reframe the model as the composition of two uncertain MDPs $(M_b||\mathcal{D}_b)||(M_f||\mathcal{D}_f)$, each with a transition matrix uncertainty set dictated by the parameters p_f and p_b. In this framework, the composed model has 2 parameters with 361 states. Storm's integrated parametric model checking solver can efficiently solves such a model in approximately 5×10^{-3} seconds. The output policy for every instantiation of $p_f, p_b \in [0, 1]$ induces a Markov chain that would put the reachability in the level set with the same value of R_1. However, as described in the introduction, the back car does not have access to the instantiation of the front car and vice versa, which does not make this approach a fair comparison.

B. Scalable Fleet in City Streets with Blockages

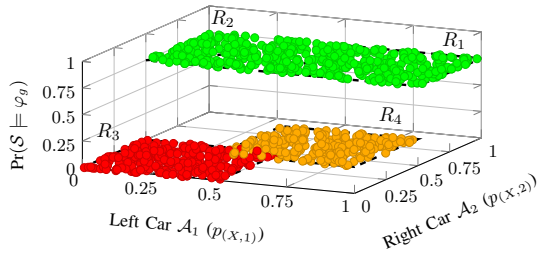
We describe a case involving a fleet of vehicles navigating a city with a grid structure. This environment can be modeled as a gridworld where an individual square represents an intersection. For this example, we assume that the vehicles can choose to move in any direction at equal cost at each intersection, i.e. U-turns are legal.

Additionally, in this city environment, there exists a moving construction crew that occupies one intersection at a given time. The construction crew has a restriction on its region of work, which is not directly known by the vehicles themselves, see shaded region in Fig. 6. Each vehicle assumes what this operating region could be from a distribution over two parameters $p_{(X,i)} \in [0, 1]$ and $p_{(Y,i)} \in [0, 1]$. For example, in Fig. 6 a value of $p_{(X,i)} = 1$ means that Agent 1 believes that the construction crew is operating in the yellow region with probability 1. Further, a value of $p_{(X,i)} = 0$ means that with probability 1 the construction region will be shifted to the left in the x direction (green area in Fig. 6). Similiarly $p_{(Y,i)}$ represents the shift probability of the construction region in the y-axis. Further, we assign the initial positions of the vehicles from west to east along the south end of the city and their goal locations are mirrored from the initial positions along the north end of the city. If a vehicle $A_i \in \mathbb{A}$ occupies the same physical location as the construction, we consider it $Stuck_i$. Additional each vehicle has a target location at G_i The objective ψ_g for the composed system $\mathcal{S}$ can be expressed as $\psi_g = \neg \bigvee_{A_i \in \mathbb{A}} Stuck_i \cup \bigwedge_{A_i \in \mathbb{A}} G_i$.

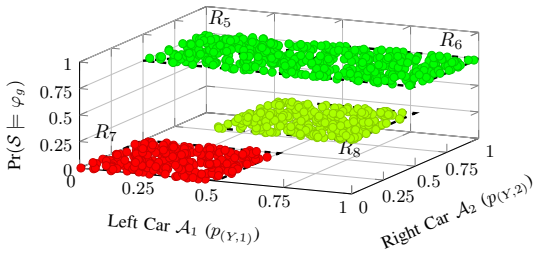
a) Result: Fig. 7 shows the impact of the agents' assumptions on the construction zone. In the two-car example, when the leading agent A_2 makes the correct assumption on the construction zone, the synthesized policy has the maximum reachability. This behavior is a result of the fact that agent A_2's behavior has more impact on A_1 than vice versa. Similarly, even if agent A_2 assumes incorrectly (Regions R_2 and R_5), the two cars will not crash into each other if the trailing agent A_1 makes the correct assumption and thus behaves conservatively with respect to the construction crew.

VII. CONCLUSIONS

This paper presented a comprehensive approach using probabilistic model checking to quantify the impact of inconsistencies in perception, intent prediction, and decision making among different agents in heterogeneous multi-agent systems. By focusing on the case where agents do not consider these inconsistencies in order to maintain computational tractability, we demonstrate that such incorrect assumptions can result in the overall system's failure to



(a) Shift in x-axis



(b) Shift in y-axis

Fig. 7: Two agent (A_1 and A_2) instance of fleet, we show the probability of satisfying the specification based upon shifts in construction region in x and y. The true region is the 2x2 yellow shaded region in the northeast corner of Fig. 6.

meet its specifications. In providing simple, but illustrative case studies we show these faulty assumptions can manifest in real-world scenarios. Moving forward, the research aims to extend its investigation by exploring the modeling of assumptions under partial observability and quantifying the effects of divergent beliefs among interacting agents. These future directions will contribute to a deeper understanding of the challenges associated with multi-agent autonomous systems and pave the way for more robust and reliable approaches in the design and deployment of autonomous technologies.

REFERENCES

[1] J. M. Anderson, K. Nidhi, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola, *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.

[2] D. Howard and D. Dai, "Public perceptions of self-driving cars: The case of berkeley, california," in *Transportation research board 93rd annual meeting*, vol. 14, no. 4502, 2014, pp. 1–16.

[3] Z. Wadud, D. MacKenzie, and P. Leiby, "Help or hindrance? the travel, energy and carbon impacts of highly automated vehicles," *Transportation Research Part A: Policy and Practice*, vol. 86, pp. 1–18, 2016.

[4] W. Zhan, C. Liu, C. Chan, and M. Tomizuka, "A non-conservatively defensive strategy for urban autonomous driving," in *ITSC*. IEEE, 2016, pp. 459–464.

[5] California Department of Motor Vehicles, "Report of traffic collision involving an autonomous vehicle (OL 316)." [Online]. Available: https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/autonomousveh_ol316

[6] "Why people keep rear-ending self-driving cars," Oct 2018. [Online]. Available: https://www.wired.com/story/self-driving-car-crashes-rear-endings-why-charts-statistics/

[7] *Technical Reference for Autonomous Vehicles: Part 1 : Basic behaviour*, ser. Technical reference. Enterprise Singapore, 2019.

[8] A. Censi, K. Slutsky, T. Wongpiromsarn, D. S. Yershov, S. Pendleton, J. G. M. Fu, and E. Frazzoli, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *ICRA*. IEEE, 2019, pp. 8536–8542.

[9] T. Phan-Minh, K. X. Cai, and R. M. Murray, "Towards assume-guarantee profiles for autonomous vehicles," in *CDC*. IEEE, 2019, pp. 2788–2795.

[10] W. Liu, S. Kim, S. Pendleton, and M. H. Ang, "Situation-aware decision making for autonomous driving on urban road using online POMDP," in *Intelligent Vehicles Symposium*. IEEE, 2015, pp. 1126–1133.

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

[12] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *ITSC*. IEEE, 2014, pp. 392–399.

[13] S. Carr, N. Jansen, and U. Topcu, "Verifiable RNN-based policies for POMDPs under temporal logic constraints," in *IJCAI*. ijcai.org, 2020, pp. 4121–4127.

[14] A. Beynier, F. Charpillet, D. Szer, and A.-I. Mouaddib, "DEC-MDP/POMDP," *Markov Decision Processes in Artificial Intelligence*, pp. 277–318, 2013.

[15] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Math. Oper. Res.*, vol. 27, no. 4, pp. 819–840, 2002.

[16] T. Winkler, S. Junges, G. A. Pérez, and J. Katoen, "On the complexity of reachability in parametric markov decision processes," in *CONCUR*, ser. LIPIcs, vol. 140, 2019.

[17] M. Cubuktepe, N. Jansen, S. Junges, J. Katoen, and U. Topcu, "Scenario-based verification of uncertain MDPs," in *TACAS*, ser. LNCS, vol. 12078. Springer, 2020, pp. 287–305.

[18] S. Junges, E. Ábrahám, C. Hensel, N. Jansen, J. Katoen, T. Quatmann, and M. Volk, "Parameter synthesis for markov models," *CoRR*, vol. abs/1903.07993, 2019.

[19] 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*. IEEE, 2013, pp. 5011–5018.

[20] T. Chen, M. Z. Kwiatkowska, A. Simaitis, and C. Wiltsche, "Synthesis for multi-objective stochastic games: An application to autonomous urban driving," in *QEST*, ser. LNCS, vol. 8054. Springer, 2013, pp. 322–337.

[21] R. Alur, S. Moarref, and U. Topcu, "Compositional and symbolic synthesis of reactive controllers for multi-agent systems," *Inf. Comput.*, vol. 261, no. Part, pp. 616–633, 2018.

[22] T. Bandyopadhyay, Z. J. Chong, D. Hsu, M. H. A. Jr., D. Rus, and E. Frazzoli, "Intention-aware pedestrian avoidance," in *ISER*. Springer, 2012.

[23] A. Jain, H. S. Koppula, B. Raghavan, S. Soh, and A. Saxena, "Car that knows before you do: Anticipating maneuvers via learning temporal driving models," in *ICCV*. IEEE Computer Society, 2015.

[24] M. Cubuktepe, Z. Xu, and U. Topcu, "Policy synthesis for factored MDPs with graph temporal logic specifications," in *AAMAS*, 2020.

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

[26] A. Pnueli, "The temporal logic of programs," in *FOCS*. IEEE Computer Society, 1977, pp. 46–57.

[27] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *CAV*, ser. LNCS. Springer, 2011.

[28] C. Dehnert, S. Junges, J. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *CAV*, ser. LNCS. Springer, 2017.

[29] M. Cubuktepe, N. Jansen, S. Junges, J. Katoen, I. Papusha, H. A. Poonawala, and U. Topcu, "Sequential convex programming for the efficient verification of parametric mdps," in *TACAS*, ser. LNCS, 2017.

[30] W. Wiesemann, D. Kuhn, and B. Rustem, "Robust markov decision processes," *Math. Oper. Res.*, vol. 38, no. 1, pp. 153–183, 2013.

[31] C. Hensel, S. Junges, J. Katoen, T. Quatmann, and M. Volk, "The probabilistic model checker storm," *CoRR*, vol. abs/2002.07080, 2020.

[32] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun, "CARLA: an open urban driving simulator," in *CoRL*. PMLR, 2017.