Game Theory for Simulator-Based Testing of Autonomous Vehicles

Kevin McAreavey

16 December 2020

Abstract

This is a technical report for TSL on the use of game theory for simulator-based testing of autonomous vehicles, including some possible research directions.

1 Problem Analysis

Applicability of game theory: Game theory captures the notion that an ego vehicle is an autonomous entity (i.e. one outside of our control) having partial control over the environment state (and thus the playout of a test run). An ego vehicle may cause the system to avoid interesting test situations simply because it is rational to do so (e.g. because they correspond to situations where the vehicle may be subject to a high risk of collisions). Conversely, an ego vehicle may instead cause the system to avoid these situations by strategically attempting to *pass* a test (e.g. through design or by learning) and thus exhibit strongly adversial behaviour with respect to testing. Either outcome has the potential to frustrate our achievement of test objectives.

Number of players: An autonomous vehicle simulator is comprised of a collection of environment properties and objects exhibiting both static and dynamic characteristics. For example, the environment may have static roads but dynamic weather conditions, while vehicle objects may have static dimensions but dynamic positions. Dynamic objects can be regarded as simulation *bodies*, each with their own sensors and effectors mapped to characteristics of themselves and the environment. Dynamic environment properties can then can be regarded characteristics of a special *environment body*. Static characteristics are simply constants associated with a specific body. Game theory does not require or assume that each of these bodies are considered to be players in a game. Instead the problem can be more naturally formulated as a two-player game between the ego vehicle (or ego agent) and a single tester agent. In this formulation the ego agent has control over one or more effectors—typically associated with a single body but not strictly required—while the remaining effectors are under the control of the tester agent. The simulator can thus be regarded as a set of effectors such that a body is simply a convenient encapsulation of a subset of those effectors. It is conceivable to divide the effectors under the control of the tester agent among multiple tester agents, but the

possibility of competitive behaviour emerging among tester agents may be difficult to justify and introduce unnecessary complexity. In the absence of an ego agent, we could fully control the simulator using multi-effector and/or multi-body planning [9].

Time: We can reasonably assume that simulators are composed of discrete timesteps, with real-time being a product of the timestep and a time resolution constant. This is consistent with sequential AI and game-theoretic frameworks, which typically assume that time is implicit and discrete. However, high-fidelity simulators (especially those based on game engines) may use variable time resolutions which is likely to complicate these existing frameworks. For example, if time resolutions are not known in advance, then projecting environment dynamics into the future (i.e. estimating the state at a future timestep) is more difficult because we cannot determine the real-time interval that will occur in the intervening period.

Determinism: In theory, simulators are deterministic unless they have been designed to simulate non-deterministic environments. In practice, however, simulators may exhibit non-deterministic behaviour even when they are intended to simulate deterministic environments. We know that CARLA exhibits non-deterministic behaviour under certain conditions (e.g. high CPU load), so this is likely to be true of most current high-fidelity simulators. Conversely, we have not yet found any situations where CAV-GYM exhibits non-deterministic behaviour, which suggests that unintentional non-determinism could be solved by designing an appropriate simulator and test bench (e.g. by avoiding dynamic time resolutions or similar visualisation-centric devices). At the level of testing and/or test generation, unintentional non-determinism will manifest to the agent as environmental uncertainty and thus may influence its own decision-making, which in turn may affect the quality of generated tests. For example, a test generated when the simulator is behaving completely deterministically may produce different results if the test is rerun when the simulator is behaving non-deterministically, and vice versa.

Continuous states/actions: A key characteristic of autonomous vehicle simulators is that they exhibit continuous state spaces (coordinates, orientations, etc.) as well as continuous action spaces (throttles, steering, etc.). This is true of high-fidelity simulators such as CARLA as well as low-fidelity simulators such as CAV-GYM: the simplifications introduced by low-fidelity simulators replace high-dimensional spaces with low-dimensional spaces but do not solve the core problem of continuous spaces. From a theoretical perspective, many of the key theorems in game theory assume a finite number of actions (e.g. existence of a Nash equilibrium was originally proved under this assumption). Conversely, from a practical perspective, many existing algorithms for implementing agents assume a finite number of states and/or actions (e.g. Q-learning assumes finite states and actions, while approximate Q-learning assumes finite actions). These problems pose a significant research challenge but not an insurmountable one. For example, a Nash equilbrium is known to exist if continuous action spaces satisfy certain properties (e.g. that they are compact sets in metric space), while more sophisticated RL algorithms are able to handle both continuous state and action spaces. Finally, while continuous state spaces are unavoidable, continuous action spaces can be simplified through discretisation, but it is unclear what impact this would have on testing.

Composite/durative actions: The obvious method for discretisation of continuous actions is to simply select a finite subset of actions (e.g. a finite number of valid steering angles). However, this discretisation method may not be sufficient to enable desirable agent behaviour and instead more complex methods may be required. For example, if we discretise steering actions into a finite number of target orientations, then any given orientation may only be reachable by the execution of a sequence of steering actions. Conversely, if we discretise actions into a finite number of relative waypoints, then any given waypoint may only be reachable by the execution of a sequence of steering and throttle actions. This suggests that meaningful discretisation of continuous actions may depend on composite and/or durative actions. Unfortunately, this also complicates the underlying frameworks; possibly leading to a disconnect between the environment and the agent's model of the environment. For example, if RL is applied at the level of durative actions, ignoring events that occur during those time-intervals, then the agent will have a severly limited understanding of the environment dynamics.

Sensors: Many existing high-fidelity simulators support the ability to simulate vehicle sensors, such as computer vision and classification components. From a theoretical perspective, this avoids the strong assumption that an agent has full observability of the environment state, and instead allows the simulator to account for situations where an agent has partial (and potentially imperfect) observability. This is undoubtedly useful from the perspective of simulator-based testing of autonoomous vehicles, but may be superfluous to the sub-problem of testing the vehicle's controller. Essentially, if the agent is assumed to have full observability, then testing of sensors can be decoupled from testing of the controller. If fully integrated testing remains the ultimate goal, the most logical research direction may be to first develop methods for testing the controller under full observability and subsequently extend those methods to the setting of partial observability. Thus, full observability is a reasonable assumption, but game-theoretic frameworks that can be suitably adapted to partial observability may still be desirable.

Objectives: During any simulation run, the actions of both ego and tester agent are guided by their objectives. Consequently, from the perspective of simulator-based test generation, there are three main questions: (i) what are these objectives; (ii) how are these objectives best represented; and (iii) who specifies these objectives. For the tester agent, we have been working under the assumption that their overriding objective is to transition the system into a state satisfying an assertion precondition: presumably a test run would then be regarded as a failed test if the assertion was immediately invalidated, or a pass otherwise. This suggests that the tester agent is equipped with a set of terminating goal states, corresponding to those states satisfying a given assertion precondition. It is reasonable to assume that these assertions could be acquired from the domain and thus specified as part of the test setup. In addition, the agent may be equipped with feedback on the quality/cost of different states or state-transitions, and these would serve to stratify the set of valid tests. Within AI, such feedback would be commonly represented as rewards and regarded as part of the simulation. The use of rewards seems reasonable but it is less clear who is responsible for specifying those rewards and how those rewards would impact the quality of generated tests. For the ego agent, the situation is on the whole much less clear. For example, we could assume that the ego is equipped with equivalent environmental rewards dictating the ego's be-

haviour, and yet these rewards may not accurately reflect the design of the controller. Perhaps a more likely scenario is that the ego agent is instructed to follow a specific path, but this may be overly rigid, and path following does not easily map into either goal states or rewards (as we have seen with the Frenet controller). If this path has an end-point, then reaching that end-point may correspond to a terminating goal state for the ego, which could permit the ego to terminate the test itself. Again, it is unclear who specifies this path and what impact the chosen path would have on test generation.

Knowledge: The use of a simulator suggests that the environment model (e.g. transition probabilities) is unknown and must instead be sampled through interaction with the simulator. This in turn suggests that RL, including techniques at the intersection between RL and planning (such as Monte Carlo tree search), may be the most promising directions. These techniques would benefit greatly from a well-optimised simulator that is able to run simulations very quickly. An alternative option is to learn a model directly from the simulator and then apply standard online or offline planning techniques to that model. However, the complexity of the simulator (and thus the complexity of the model) is likely to make this approach infeasible in practice. There is also the option of using domain knowledge to program agents directly (e.g. using agent-oriented programming or rule-based systems) but this approach would be rigid and labour-intensive. Crowdsourced test generation may be a more effective way to capture this kind of domain knowledge. In addition to knowledge of the model, game theory also raises the question of a player's knowledge of the game that is being played, which in turn may influence their optimal actions. Given that we have previously identified a discrete time framework, we can reasonably conclude that the game is a simulaneous-move rather than sequential-move game. The means players must choose an action at each timestep without prior knowledge of the actions of other players, although they can observe actions when they are actually executed. More important to test generation is the question of whether the ego has knowledge of the tester agent's goal states: if the ego has knowledge of these goal states, then intuitively the ego would benefit from diverting the system away from these states. This possibility suggests that a player's knowledge of the game may depend on the test setting: a vehicle manufacturer is incentivised to ensure their controller passes the tests of a regulator, but the same incentive does not exist during internal development (where failed tests are useful).

Initial state: Every simulation run starts with an initial state. In principle, a test generated from a single run would only be applicable with respect to that initial state. The choice of an initial state thus has a significant impact on test generation: the same goal state may be reachable from two initial states in very different ways, and likewise may be unreachable from some other initial state. Offline solutions for sequential AI and game-theoretic frameworks would account for this issue in one of two ways: either (i) the initial state is given as part of the problem and the solution only needs to specify an action for states that might be reached from that initial state; or (ii) the initial state is not given and the solution needs to specify an action for any state. The former option is arguably more appealing than the latter, due to the latter's obvious computational complexity and the fact that solutions would need to specify actions for states that may never be encountered in practice (not to mention the issue of continuous state spaces). However, the former raises its own challenges. Two problems with different

initial states but being otherwise identical are still different problems, and a solution to one does not necessarily constitute a solution to the another. We could then conclude that the initial state should be part of the test specification, but it is likely that some initial states will prove better than others (e.g. based on the likelihood of achieving the test objective) and so it is unclear how this initial state can be chosen. A third option is to integrate testing and test generation by computing solutions in an online fashion for states that have not been encountered before while reusing solutions for previously encountered states. This would not solve the issue of choosing the initial state for each test run, but would at least account for the possibility of different initial states without needing to compute a complete solution offline.

2 Existing Frameworks

2.1 Single-agent Frameworks

Definition 1. *A Markov decision process (MDP) is a tuple* (S, A, T, R) *where:*

- S *is a (possibily infinite) set of states*
- A *is a (possibly infinite) set of actions*
- $T : S \times A \to \Delta(S)$ *is a (stochastic) transition function*[1]
- $R : S \times A \times S \to \mathbb{R}$ *is a reward function*

In an MDP, the transition function describes the environment, while the reward function describes the agent's objective. Solutions to MDPs are defined in terms of optimal (reward-maximising) policies, and are only well-defined for special classes of MDP, notably finite-horizon MDPs and infinite-horizon (discounted-reward) MDPs. Note that the reward function can be replaced by a cost function $C : S \times A \times S \to \mathbb{R}$ where optimal policies are then defined as cost-minimising policies. Reward and cost functions may also be constrained, e.g. $C(s, a, s') \geq 0$ for all $s, s' \in S$ and each $a \in A$. A finite-horizon MDP is a tuple $(S, A, T, R, t_{\max})$ with $t_{\max} \in \mathbb{N}$ a finite-horizon and $D = \{1, \ldots, t_{\max}\}$ the set of timesteps. An infinite-horizon MDP is a tuple (S, A, T, R, γ) with $\gamma \in [0, 1)$ a discount factor. Both classes of MDP guarantee finite value for any policy. Optimal policies for infinite-horizon MDPs are determinsitic and stationary, taking the form $\pi : S \to A$, while optimal policies for finite-horizon MDPs are determinsitic but non-stationary, taking the form $\pi : S \times D \to A$. An MDP may be known or unknown: typically the states and actions are known, but the transition and reward functions may be known or unknown. Solving a known MDP can be viewed as a form of planning, sometimes called decision-theoretic planning [4]. As with many areas of AI, decision-theoretic planning has seen much recent success in the appplication of Monte Carlo tree search [2]. Solving an unknown MDP is the domain of RL [12]. Learning the transition and reward functions and then solving the known MDP can be regarded as a simple RL method, but RL methods more commonly learn the optimal policy directly

[1] $\Delta(S)$ is the set of probability distributions over S.

and do not attempt to learn the model. RL typically focuses on infinite-horizon MDPs but uses finite-length episodes to aid learning (by improving exploration). However, episodes are distinct from finite-horizons, and care must be taken to ensure that the correct MDP class (and thus correct policy formalism) is used [6].

Definition 2. *A stochastic shortest path problem (SSP) is a tuple* (S, A, T, C, G) *where:*

- (S, A, T, C) *is an MDP with* C *a cost function*
- $G \subseteq S$ *is a (possibily infinite) set of terminating goal states*

Unlike standard classes of MDP, a well-defined SSP has an indefinite horizon: that is, a finite but a priori unknown horizon. This complicates much of the theory, since SSPs are only well-defined if they exhibit at least one policy that is guaranteed to reach the goal with probability 1, known as a proper policy. This precludes many natural SSPs, such as those exhihiting unavoidable dead-ends, but there has also been much effort in identifying special classes of SSP where solutions are guaranteed to exist. It is known that SSPs subsume both finite- and infinite-horizon MDPs, where the finite-horizon and discount factor induce special terminating states: with the former, states are time-indexed and thus terminating states are those at the horizon; with the latter, the discount factor specifies a probability that the system will transition at any point into a special terminating state. SSPs also subsume classical planning when the transition function is deterministic. There is some work in RL based on SSPs [5], but this work is less well-established than RL work based on infinite- or even finite-horizon MDPs.

2.2 Multi-agent Frameworks

Definition 3. *A Markov game (or stochastic game) [10] is a tuple* $(N, S, A, T, \boldsymbol{R})$ *where:*

- $N = \{1, \ldots, n\}$ *is a (finite) set of players*
- S *is a (possibily infinite) set of states*
- $A = A_1 \times \cdots \times A_n$ *is a set of action profiles with* A_i *the (possibly infinite) set of actions available to player* $i \in N$
- $T : S \times A \to \Delta(S)$ *is a (stochastic) transition function*
- $\boldsymbol{R} = (R_1, \ldots, R_n)$ *is a profile of reward functions with* $R_i : S \times A \times S \to \mathbb{R}$ *the reward function for player* $i \in N$

A Markov game extends an MDP with a set of players (or agents), each with their own reward function, such that the system transitions between states following the execution of joint actions (derived from the action-choice of each agent). A Markov game is thus able to express the idea that agents in a multi-agent system are autonomous (can choose their own actions) and self-interested (have their own reward function) but may need to account for the actions of others (since the state depends on the joint action, rather than individual actions). The framework assumes that agents choose their

actions simultaneously and thus without knowledge of the actions choosen by other agents. However, agents can typically observe the joint action that is executed, and can make predictions about what actions other agents are likely to execute. Key insights from MDPs extend to Markov games, including that finite- and infinite-horizon Markov games guarantee finite value for any policy, and that optimal policies are stationary in the infinite-horizon case and non-stationary in the finite-horizon case. However, unlike MDPs, optimal policies for Markov games are typically stochastic rather than deterministic policies, taking the form $\pi_i : S \to \Delta(A_i)$ or $\pi_i : S \times D \to \Delta(A_i)$. Solving an unknown Markov game can then be regarded as the domain of multi-agent RL (MARL) [1], with perhaps the best-known example being minimax-Q-learning [3]. As with RL, much of the existing work on MARL focuses on the infinite-horizon case.

Definition 4. *A (two-player, zero-sum) SSP game [7, 8] is a tuple* (N, S, A, T, C, G) *where:*

- $N = \{\min, \max\}$ *is the set of players*
- (S, A, T) *is defined as in a Markov game*
- $C : S \times A \times S \to \mathbb{R}$ *is a cost function for the* min *player*
- $G \subseteq S$ *is a (possibily infinite) set of goal states for the* min *player*

Similar to SSPs and MDPs, SSP games extend Markov games with the introduction of terminating goal states. Many of the insights from SSPs extend to SSP games, including the notion of proper policies and the fact that finite-horizons and discount factors induce special goal states. However, unlike SSPs and MDPs, SSP games do not fully subsume Markov games because existing definitions are restricted to the two-player zero-sum case. In this setting there are two players—a min player and a max player—but only one cost function. In addition, there is a set of terminating goal states for the min player. The objective for the min player is to reach a goal state while minimising cost, while the objective for the max player is to maximise cost for the min player (which might be achieved by delaying termination). It is unclear from the literature if or how SSP games can be generalised to an arbitrary number of players, to arbitrary cost functions for each player, and/or to arbitrary goal states for each player.

3 Research Directions

Generalised SSP games: It is easy to argue that simulator-based testing of autonomous vehicles is a two-player game, so the restriction of SSP games to the two-player setting does not pose any meaningful obstacles. However, the restriction of SSPs to zero-sum games with goal states for only one player is potentially more problematic. There does not appear to be any existing work on general-sum SSP games and/or SSP games with goals states for both players. Unfortunately, it is unlikely that this lack of existing work means that the problems are yet to be studied, but more likely that the problems are simply very challenging from a mathematical and computational perspective. Therefore, solving these problem should not be regarded as a realistic research objective for TSL,

and instead we should take a pragmatic view: formulating the problem as an SSP game can help to identify incremental research objectives (e.g. see below) while avoiding the pitfalls of research directions that are unlikely to be well-founded (e.g. immediately attempting to solve general-sum problems with multiple goals).

Time-limited tests: We know from the literature that finite-horizon MDPs are a special class of SSP where the horizon guarantees the existence of a proper policy, and that SSP games are well-defined when there exists a proper policy for the min player. As with finite-horizon Markov games, it is likely that we can guarantee the existence of a proper policy in an SSP game by imposing a time limit: this would be a kind of finite-horizon SSP with early terminations where the system terminates early if a goal state is reached but otherwise terminates when the horizon is reached. The main challenge is that policies arriving in a goal state may be more costly than policies that run to completion, even though policies that arrive in a goal state should (intuitively) always be preferred. If general cost/reward functions (i.e. those taking positive and negative values) are permitted, for example, then early terminations would offer the agent less opportunity to accumulate reward. An obvious solution would then be to impose local constraints on the cost/reward functions, but the more constraints that are imposed, the less interesting the solution may be to the broader research community (since it would permit fewer applications). A final observation is that, while this class of SSP game would guarantee the existence of a proper policy, it would also imply that our solutions should be non-stationary rather than stationary policies.

Ego objective: Specifying the ego's objectives during during test generation—or more generally ensuring that the ego behaves *normally*—may be a fundamental conceptual challenge. We have seen this to some extent when trying to design our own ego agents, where we have needed to specify paths for the Frenet agent and design new environment rewards for the approximate Q-learning ego agent. The choice of ego objectives during test generation will strongly influence the quality (and possibly the validity) of generated tests since the ego agent maintains partial control over the environment state. However, to address this issue, it may be a simple matter of identifying some realistic use cases that determine who would specify these objectives and how.

Multi-agent RL: From a practical perspective, a promising research direction is to explore the use of appropriate (game-theoretic) MARL techniques [11] in designing a single multi-body tester agent and comparing those to the use of single-agent RL techniques for the same agent. Likewise, these multi-body agents could be compared to a set of single-body tester agents each designed using single-agent RL techniques. Standard minimax-Q-learning is not applicable because it assumes finite state and action spaces, but there may be more recent variants that permit continuous state and/or action spaces. Another interesting avenue is the use of Monto Carlo tree search techniques to design the tester agent: these techniques are generally applied to sequential-move games but the issue might be solved by simple methods to translate a simulataneous-move game into a sequential-move game.

References

[1] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[2] Thomas Keller and Patrick Eyerich. PROST: Probabilistic planning based on UCT. In *Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 119–127, 2012.

[3] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine learning (ICML'94)*, pages 157–163. 1994.

[4] Mausam and Andrey Kolobov. *Planning with Markov decision processes: An AI perspective*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[5] Gergely Neu, Andras Gyorgy, and Csaba Szepesvári. The adversarial stochastic shortest path problem with unknown transition probabilities. In *Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS'12)*, pages 805–813, 2012.

[6] Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML'18)*, pages 4045–4054, 2018.

[7] Stephen D. Patek. *Stochastic and shortest path games: theory and algorithms*. PhD thesis, Massachusetts Institute of Technology, 1997.

[8] Stephen D. Patek and Dimitri P. Bertsekas. Stochastic shortest path games. *SIAM Journal on Control and Optimization*, 37(3):804–824, 1999.

[9] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.

[10] Lloyd S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences*, 39(10):1095–1100, 1953.

[11] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial intelligence*, 171(7):365–377, 2007.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2nd edition, 2018.