Simulator-Based Verification of Autonomous Vehicles: Agent-Based Test Generation

11:21am, 17 September 2020

1 Preliminaries

We rely on some standard mathematical notation: v_i is an element of vector $\boldsymbol{v} = (v_1, \ldots, v_n)$ with $\boldsymbol{v}_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ the subvector of $\boldsymbol{v}$ excluding v_i, $|S|$ is the cardinality of set S, 2^S is the powerset of S, $\Delta(S)$ is the set of probability distributions over S, $\mathbb{R}$ is the set of real numbers with $\mathbb{R}^{\geq 0}$ is the set of non-negative real numbers, $\mathbb{Z}^{\geq 0}$ is the set of non-negative integers, and $\mathbb{N}$ is the set of natural numbers. A function $f : X \to Y$ is a surjection if for each $y \in Y$ there exists some $x \in X$ such that $f(x) = y$. Given a function $f : X \to Y$, the inverse image of function f is a function $f_1^{-1} : Y \to 2^E$ defined as $f_1^{-1}(y) = \{x \in X \mid f(x) = y\}$ for each $y \in Y$.

1.1 Normal-Form Games

A normal-form game is a tuple $(N, A, \boldsymbol{u})$ where $N = \{1, \ldots, n\}$ is a finite set of players, $A = A_1 \times \cdots \times A_n$ is a set of action profiles with A_i the set of actions available to player $i \in N$, and $\boldsymbol{u} = (u_1, \ldots, u_n)$ is a profile of utility functions with $u_i : A \to \mathbb{R}$ the utility function for i. A (mixed) strategy for player $i \in N$ is a probability distribution $\psi \in \Delta(A_i)$ with $\boldsymbol{\psi} \in \Delta(A_1) \times \cdots \times \Delta(A_n)$ a (mixed) strategy profile. A strategy ψ for player $i \in N$ is a pure strategy if $\psi(a) = 1$ for some $a \in A_i$. A strategy profile $\boldsymbol{\psi}$ is a pure strategy profile if ψ_i is a pure strategy for each $i \in N$. For convenience, we may denote a pure strategy ψ for player $i \in N$ directly by the action $a \in A_i$ such that $\psi(a) = 1$. Likewise, we may denote a pure strategy profile $\boldsymbol{\psi}$ directly by the action profile $\boldsymbol{a} \in A$ such that $\psi_i(a_i) = 1$ for each player $i \in N$. The expected utility of a strategy profile $\boldsymbol{\psi}$ is then defined as:

$$u_i(\boldsymbol{\psi}) = \sum_{\boldsymbol{a} \in A} u_i(\boldsymbol{a}) \prod_{j \in N} \psi_j(a_j)$$

In game theory, a solution concept is a prediction of how a game will be played. Solution concepts are typically based on the idea that each player will try to maximise their expected utility under the assumption that other players will do the same. A feasible deviation from strategy profile $\boldsymbol{\psi}$ by player $i \in N$ is a strategy profile $\boldsymbol{\psi'} = (\boldsymbol{\psi}_{-i}, \psi')$ where ψ' is a strategy for i. A strategy profile $\boldsymbol{\psi}$ is a Nash equilibrium if no player

$i \in N$ has a feasible deviation $\boldsymbol{\psi'}$ from $\boldsymbol{\psi}$ such that $u_i(\boldsymbol{\psi'}) > u_i(\boldsymbol{\psi})$. Every normal-form game with a finite set of players and a finite set of action profiles is guaranteed to have at least one (not necessarily pure) Nash equilibrium [1].

1.2 Markov Games

A (fully observable) Markov game is a tuple $(N, S, A, T, \boldsymbol{R})$ where $N = \{1, \dots, n\}$ is a finite set of players, S is a set of states, $A = A_1 \times \cdots \times A_n$ is a set of action profiles with A_i the set of actions available to player $i \in N$, $T : S \times A \to \Delta(S)$ is a (stochastic) transition function, and $\boldsymbol{R} = (R_1, \dots, R_n)$ is a profile of reward functions with $R_i : S \to \mathbb{R}$ the reward function for player $i \in N$. A Markov game is zero-sum if $\sum_{i \in N} R_i(s) = 0$ for each state $s \in S$.

A Markov game is composed of a sequence of normal-form games, called stage games, where players seek to optimise their return for the whole game rather than for individual stage games. The current stage game is determined by the current state, and players transition between states by executing action profiles. The game is said to be fully observable because players are assumed to observe the current (shared) state at each decision step. Let $T(s, \boldsymbol{a}, s')$ denote the probability of transitioning to state $s' \in S$ after executing action profile $\boldsymbol{a} \in A$ in state $s \in S$ according to probability distribution $T(s, \boldsymbol{a})$. The probability of transitioning to state $s' \in S$ after executing strategy profile $\boldsymbol{\psi}$ in state $s \in S$ is:

$$T(s, \boldsymbol{\psi}, s') = \prod_{\boldsymbol{a} \in A} T(s, \boldsymbol{a}, s') \prod_{j \in N} \psi_j(a_j)$$

The stage game for state $s \in S$ is a normal-form game $(N, A, \boldsymbol{u})$ where the utility function u_i for player $i \in N$ is defined for each action profile $\boldsymbol{a} \in A$ as the immediate expected reward:

$$u_i(\boldsymbol{a}) = \sum_{s' \in S} T(s, \boldsymbol{a}, s') R_i(s')$$

Markov games subsume several other important frameworks: a Markov game is a repeated game if $|S| = 1$, and is a Markov decision process (MDP) if $|N| = 1$.

Solutions to Markov games are represented as functions, called policies, that map *histories* to strategies. An execution is a possibly infinite sequence $(s_1, \boldsymbol{a}_1, s_2, \boldsymbol{a}_2, \dots)$ of states and action profiles. A history of length t is a finite execution $h_t = (s_1, a_1, \dots, a_{t-1}, s_t)$ ending in a state. Let H_t be the set of histories of length t with $D = \{1, 2, \dots, t_{\max}\}$ the set of decision-steps up to horizon $t_{\max} \in \mathbb{N} \cup \{\infty\}$ and $H = \{h \in H_t \mid t \in D\}$ the set of histories up to $t_{\max}$. A (stochastic or mixed) policy for player $i \in N$ is a function $\pi_i : H' \to \Delta(A_i)$ where $H' \subseteq H$. Let $\pi_i(h, a)$ denote the probability that player $i \in N$ should execute action $a \in A_i$ in history $h \in H'$ according to strategy $\pi_i(h)$. A policy π_i for player $i \in N$ is a deterministic (or pure) policy if $\pi_i(h, a) = 1$ for each $h \in H'$ and some $a \in A_i$. A policy π_i for player $i \in N$ is state-based if $\pi_i(h_t) = \pi_i(h_{t'})$ for all $h_t, h_{t'} \in H'$ such that $t = t'$ and $s_t = s_{t'}$. A state-based policy for player $i \in N$ may be written as $\pi_i : X \to \Delta(A_i)$ where $X \subseteq S \times D$. A state-based policy π_i for player $i \in N$ is stationary if $\pi_i(s, t) = \pi_i(s, t')$ for all $(s, t), (s, t') \in X$, otherwise π_i is non-stationary. A stationary state-based policy

for player $i \in N$ may be written as $\pi_i : S' \to \Delta(A_i)$ where $S' \subseteq S$. A tuple $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ is a policy profile (resp. deterministic policy profile) if each π_i is a policy (resp. deterministic policy) over $H' \subseteq H$. The strategy profile for history $h \in H'$ according to policy profile $\boldsymbol{\pi}$ is $\boldsymbol{\pi}(h) = (\pi_1(h), \ldots, \pi_n(h))$.

Solutions are only well-defined for certain classes of Markov game. Two standard examples are finite-horizon Markov games and infinite-horizon (discounted reward) Markov games. A finite-horizon Markov game is a tuple $(\Gamma, t_{\max})$ where Γ is a Markov game and $t_{\max} \in \mathbb{N}$ is a horizon. An infinite-horizon Markov game is a tuple (Γ, γ) where Γ is a Markov game and $0 \leq \gamma < 1$ is a discount factor. It is known that attention can be restricted to stationary state-based policies in the case of infinite-horizon Markov games, and to non-stationary state-based policies in the case of finite-horizon Markov games. Depending on context, we will assume hereafter that all policies are either stationary or non-stationary state-based policies. For an infinite-horizon Markov game (Γ, γ), the expected value of policy profile $\boldsymbol{\pi}$ for player $i \in N$ in state $s \in S$ is:

$$V_i(\boldsymbol{\pi}, s) = \sum_{s' \in S} T(s, \boldsymbol{\pi}(s), s') \left[R_i(s') + \gamma V_i(\boldsymbol{\pi}, s') \right]$$

For a finite-horizon Markov game $(\Gamma, t_{\max})$, the expected value of policy profile $\boldsymbol{\pi}$ for player $i \in N$ in state $s \in S$, with $t \in D \cup \{0\}$ remaining decision-steps, is:

$$V_i(\boldsymbol{\pi}, s, t) = \begin{cases} \sum\limits_{s' \in S} T(s, \boldsymbol{\pi}(s, t), s') \left[R_i(s') + V_i(\boldsymbol{\pi}, s', t-1) \right] & \text{if } t > 0 \\ 0 & \text{if } t = 0 \end{cases}$$

A feasible deviation from policy profile $\boldsymbol{\pi}$ by player $i \in N$ is a policy profile $\boldsymbol{\pi'} = (\boldsymbol{\pi}_{-i}, \pi')$ where π' is a policy for i. A policy profile $\boldsymbol{\pi}$ is a Nash equilibrium if no player $i \in N$ has a feasible deviation $\boldsymbol{\pi'}$ from $\boldsymbol{\pi}$ such that $V_i(h, \boldsymbol{\pi'}) > V_i(h, \boldsymbol{\pi})$ for each $h \in H'$.

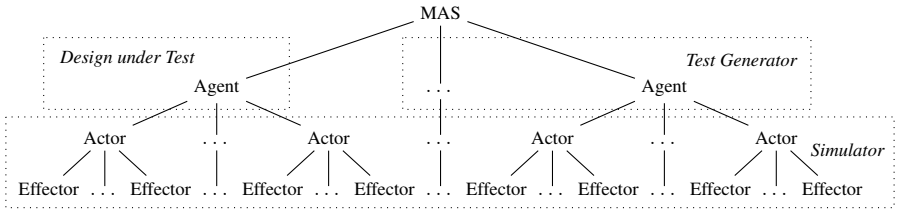


Figure 1: Architecture for agent-based test generation

2 Framework

Definition 1. *A (simulator-based) test generation architecture is a tuple* (N, M, E, f_1, f_2) *where:*

- $N = \{1, \ldots, n\}$ *is a set of agents such that* $n \geq 2$ *with* $1 \in N$ *the ego agent (or design under test)*
- M *is a set of actors with* $f_1 : M \to N$ *a surjection mapping actors to agents*
- E *is a set of effectors with* $f_2 : E \to M$ *a surjection mapping effectors to actors and* $A(e)$ *the non-empty set of actions available to effector* $e \in E$

Corollary 1. $2 \leq |N| \leq |M| \leq |E|$.

The set of actors associated with agents $N' \subseteq N$ is defined as $M(N') = \{m \in f_1^{-1}(i) \mid i \in N'\}$. The set of effectors associated with actors $M' \subseteq M$ is defined as $E(M') = \{e \in f_2^{-1}(m) \mid m \in M'\}$. The set of joint actions available to effectors $E' \subseteq E$ is defined as $A(E') = \times_{e \in E'} A(e)$. Definition 1 is visualised in Figure 1, with the set of actors M (resp. agents N) inducing a partition of the set of effectors E.

Definition 2. *A (simulator-based) test generation game is a tuple* $(G, S, A, T, \boldsymbol{R})$*:*

- $G = (N, M, E, f_1, f_2)$ *is a test generation architecture*
- $(N, S, A, T, \boldsymbol{R})$ *is a Markov game such that* $A_i = A(E(M(\{i\})))$ *is the set of actions available to agent* $i \in N$

Definition 3. *A test generation game* $(G, S, A, T, \boldsymbol{R})$ *is collaborative if* $R_i(s) = R_j(s)$ *for all* $i, j \in N \setminus \{1\}$ *and each* $s \in S$.

Simulator-based test generation constitutes a multi-agent system because it involves a group of (one or more) tester agents playing against a single ego agent: the former is under the control of the test generation system, while the latter is not. The objective of simulator-based test generation is then to design a group of tester agents so as to influence the ego agent towards achieving some test goal. Definition 2 formally models this problem as a Markov game (also known as a stochastic game). Markov games assume that time is composed of discrete decision steps in which agents execute actions simultaneously. Although the transition function T and reward functions

$\boldsymbol{R}$ are specified in Definition 2, these may be unknown in practice (as is the case in multi-agent reinforcement learning). What is important is that a simulator provides feedback to agents in a way that constitutes sampling from T and $\boldsymbol{R}$, allowing agents to approximate T and $\boldsymbol{R}$ through interaction with the simulator.

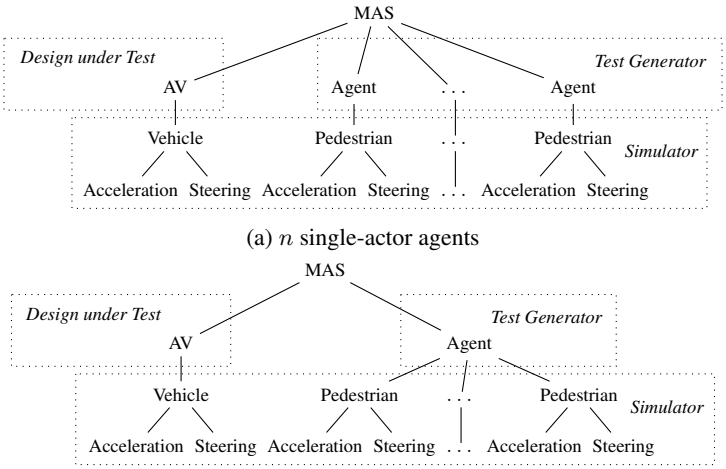
Definition 4. *Let π_1 be a deterministic policy and $\omega \subseteq S$ be a (test) assertion. An interesting test given π_1 and ω is a tuple $(s_1, \boldsymbol{\pi}_{-1})$ where $s_1 \in S$ is an initial state and $\boldsymbol{\pi}_{-1}$ is a deterministic policy profile for the set of agents $N \setminus \{1\}$ such that executing policy profile $(\pi_1, \boldsymbol{\pi}_{-1})$ in state s_1 leads to some state $s \in \omega$ within a finite number of timesteps.*

Note: Assertions could be generalised to sequences of states (e.g. using a termporal logic?) or to executions (e.g. with actions as events?).

Note: We assume that a stochastic policy is always fixed to a deterministic policy by controlling the seed, but this means the test is only interesting with respect to that seed. Is there a more general notion?

Note: How to *repeat* a test when a different ego policy π_1' may cause the assertion to be avoided altogether (e.g. by causing the agents to arrive in history $h \in H$ such that $\boldsymbol{\pi}_{-1}$ is undefined for h)?

3 Experiments



(a) n single-actor agents

(b) 1 multi-actor agent

Figure 2: Agent-based test generation in CAV-GYM:PEDESTRIANS

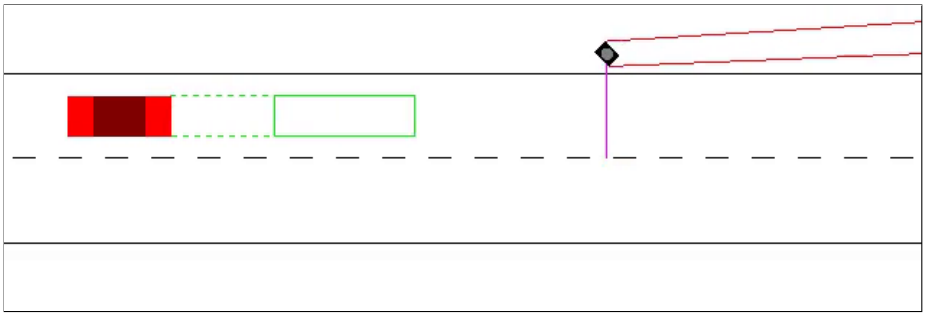


Figure 3: CAV-GYM:PEDESTRIANS

References

[1] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

Algorithm 1: SIMULATOR

persistent: environment e, agents $N = \{1, \ldots, n\}$, terminator $\psi \subseteq S$
begin
  for each *episode* **do**
    $s \leftarrow$ reset e
    for each *timestep* **do**
      $\boldsymbol{a} \leftarrow (\textsc{ChooseAction}_1(s), \ldots, \textsc{ChooseAction}_n(s))$
      $\boldsymbol{r}, s' \leftarrow$ execute $\boldsymbol{a}$ in e
      for each *agent* $i \in N$ **do**
        $\textsc{ProcessFeedback}_i(s, a_i, r_i, s')$
      if $s' \in \psi$ **then break else** $s \leftarrow s'$

Algorithm 2: RANDOMAGENT

persistent: exploration rate $\epsilon \in [0, 1]$
function CHOOSEACTION(s)
  with probability ϵ **do**
    return random choice from $A(s)$
  return $\varnothing$

Algorithm 3: PROGRAMMEDRANDOMAGENT

persistent: programmed behaviour $\pi : S \to A$, terminator $\psi \subseteq S$, exploration rate $\epsilon \in [0, 1]$
function CHOOSEACTION(s)
  if π *is active* **then**
    if $s \in \psi$ **then** set π as inactive **else return** $\pi(s)$
  with probability ϵ **do**
    set π as active
    return $\pi(s)$
  return $\varnothing$

Algorithm 4: PROGRAMMEDREACTIVEAGENT

persistent: programmed behaviour $\pi : S \to A$, trigger $\varphi \subseteq S$, terminator $\psi \subseteq S$
function CHOOSEACTION(s)
  if π *is active* **then**
    if $s \in \psi$ **then** set π as inactive **else return** $\pi(s)$
  if $s \in \varphi$ **then**
    set π as active
    return $\pi(s)$
  return $\varnothing$

Algorithm 5: PROGRAMMEDELECTIONAGENT

persistent: programmed behaviour $\pi : S \to A$, terminator $\psi \subseteq S$, coordinator c

function CHOOSEACTION(s)
  if π *is active* **then**
    if $s \in \psi$ **then** set π as inactive **else return** $\pi(s)$
  if *elected by* c **then**
    set π as active
    return $\pi(s)$
  return $\varnothing$

Algorithm 6: QLEARNINGAGENT

persistent: learning rate $\alpha \in [0, 1]$, discount factor $\gamma \in [0, 1]$, exploration rate $\epsilon \in [0, 1]$,
feature $f_j : S \times A \to \mathbb{R}$ with weight $w_j \in \mathbb{R}$ for $j = 1, \ldots, m$

function CHOOSEACTION(s)
  with probability ϵ **do**
    return random choice from $A(s)$
  return random choice from $\operatorname{argmax}_{a \in A(s)}$ QVALUE(s, a)

procedure PROCESSFEEDBACK(s, a, r, s')
  $q \leftarrow \big(r + \gamma \cdot \max_{a' \in A(s')} \text{QVALUE}(s', a')\big) - \text{QVALUE}(s, a)$
  for each *feature* f_j **do**
    $w_j \leftarrow w_j + \alpha \cdot q \cdot f_j(s, a)$

function QVALUE(s, a)
  return $\sum_{j=1}^{m} f_j(s, a) \cdot w_j$