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

5:55pm, 11 November 2020

1 Introduction

- **Time** is implicit and discrete, with agents choosing actions at each timestep, and a time resolution constant specifying the simulation time between timesteps
- **State space** is continuous and feature-based techniques are probably more effective than discretisation
- **Actions** may be multi-body and/or multi-effector
- **Action space** is continuous and discretisation typically requires durative actions
- **Observability** is full, meaning that a sensor model is not required (although partial observability can be simulated)
- **Initial state** may be given (meaning a partial policy is sufficient) or may not be given (meaning a complete policy is necessary)
- **Objectives** are goals states with transition costs/rewards, meaning in general that the horizon is indefinite, but a finite-horizon may also be imposed
- **Model** for transitions and objectives is unknown, but can be sampled through interaction with a simulator
- **Game** involving two or more players (agents), including a potentially adversarial design under verficiation (i.e. the ego agent)

2 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 $\mathbb{R}^{\geq 0}$ the subset of non-negative real numbers, 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 $f^{-1}_{B,N} : Y \to 2^X$ defined for each $y \in Y$ as $f^{-1}_{B,N}(y) = \{x \in X \mid f(x) = y\}$.

2.1 SSP Games

A tuple (N, S, A, T, R, G) is a (fully observable, two-player, zero-sum) stochastic shortest path (SSP) game if $N = \{1, 2\}$ is a set of **players**, S is a (possibly infinite) set of **states**, $A = A_1 \times A_2$ is a set of **action profiles** with A_i the (possibly infinite) set of **actions** available to player $i \in N$, $G \subseteq S$ is a (possibly infinite) set of **goal states** for player 1, $T : S \times A \to \Delta(S)$ is a (stochastic) **transition function** such that $T(s, \boldsymbol{a}, s) = 1$ for each $s \in G$ and each $\boldsymbol{a} \in A$, and $R : S \times A \times S \to \mathbb{R}$ is a **reward function** for player 1 such that $R(s, \boldsymbol{a}, s') = 0$ for each $s \in G$, each $\boldsymbol{a} \in A$, and each $s' \in S$. 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 $T(s, \boldsymbol{a})$. An **execution** is a possibly infinite sequence $(s_1, \boldsymbol{a}_1, s_2, \boldsymbol{a}_2, \ldots)$ of states and action profiles. A **history** of length t is a finite execution $h_t = (s_1, a_1, \ldots, a_{t-1}, s_t)$ ending in a state. Let H_t be the set of histories of length t with $D = \{1, 2, \ldots, t_{\max}\}$ 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 (mixed) **strategy** for player $i \in N$ is a probability distribution $\psi \in \Delta(A_i)$. A strategy ψ for player $i \in N$ is a **pure strategy** if $\psi(a) = 1$ for some $a \in A_i$. A (mixed) **policy** for player $i \in N$ is a function $\pi_i : H \to \Delta(A_i)$. Let $\pi_i(h, a)$ denote the probability that player $i \in N$ will 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 **pure** if $\pi_i(h, a) = 1$ for each $h \in H$ and some $a \in A_i$. A pure policy for player $i \in N$ may be written as $\pi_i : H \to A$. A policy π_i for player $i \in N$ is **Markovian** if $\pi_i(h) = \pi_i(h')$ for all $h, h' \in H$ such that $t = t'$ and $s = s'$ where h ends in state $s \in S$ after $t \in D$ timesteps (resp. h' ends in state $s' \in S$ after $t' \in D$ timesteps). A Markovian policy for player $i \in N$ may be written as $\pi_i : S \times D \to \Delta(A_i)$. A Markovian policy π_i for player $i \in N$ is **stationary** if $\pi_i(s, t) = \pi_i(s, t')$ for each $s \in S$ and all $t, t' \in D$, otherwise π_i is **non-stationary**. A stationary policy for player $i \in N$ may be written as $\pi_i : S \to \Delta(A_i)$.

A **policy profile** is a tuple $\boldsymbol{\pi} = (\pi_1, \pi_2)$ where π_i is a policy for player $i \in N$. The (mixed) **strategy profile** for history $h \in H$ according to policy profile $\boldsymbol{\pi}$ is $\boldsymbol{\pi}(h) = (\pi_1(h), \pi_2(h))$. A strategy profile $\boldsymbol{\pi}(h)$ for history $h \in H$ is a **pure strategy profile** for h if $\pi_i(h)$ is a pure strategy for each $i \in N$. Let $\boldsymbol{\pi}(h, \boldsymbol{a}) = \prod_{i \in N} \pi_i(h, a_i)$ be the probability that action profile $\boldsymbol{a} \in A$ will be executed in history $h \in H$ according to policy profile $\boldsymbol{\pi}$.

The **expected value** of policy profile $\boldsymbol{\pi}$ for player $i \in N$ in history $h \in H$ is:

$$V_i(\boldsymbol{\pi}, h) = \sum_{\boldsymbol{a} \in A} \boldsymbol{\pi}(h, \boldsymbol{a}) \sum_{s' \in S} T(s, \boldsymbol{a}, s') \left[R_i(s, \boldsymbol{a}, s') + V_i(\boldsymbol{\pi}, [h, \boldsymbol{a}, s'])\right] \tag{1}$$

where h ends in state $s \in S$. Standard notions of optimality are not well-defined in SSP games when $V_i(\boldsymbol{\pi}, h) = \infty$ for any history $h \in H$, or when $V_i(\boldsymbol{\pi}, h') = -\infty$ for each $h' \in H$. For this reason, solution definitions typically rely on assumptions that the process will (eventually) terminate by reaching a goal state, ensuring finite expected value for a given policy profile. Let $\mathbb{P}(s \mid h, \boldsymbol{\pi}, t)$ denote the probability of transitioning from history $h \in H$ to state $s \in S$ within $t \in \mathbb{N}$ timesteps by following policy profile $\boldsymbol{\pi}$. A policy π_1 for player 1 is **proper** at history $h \in H$ if there exists some $t \in \mathbb{N}$ such that:

$$\mathbb{P}(G \mid h, \pi_1, \pi_2, t) = \sum_{s' \in G} \mathbb{P}(s' \mid h, \pi_1, \pi_2, t) = 1 \tag{2}$$

for any policy π_2 for player 2, otherwise π_1 is **improper** at h. A policy π_1 for player 1 is proper if π_1 is proper at each history $h \in H$, otherwise π_1 is improper.[1] An SSP game is **solvable** if there exists a policy π_1 for player 1 such that π_1 is proper and $V(\pi'_1, h) = -\infty$ for any (improper) policy π'_1 for player 1 such that π'_1 is improper at history $h \in H$. A policy π^*_i for player 1 in a solvable SSP game is an **expectiminimax** policy for player 1 if π^*_i satisfies:

$$\operatorname*{argmin}_{\boldsymbol{\pi}_{-i}} V_i(\pi^*_i, \boldsymbol{\pi}_{-i}, h) \geq \operatorname*{argmin}_{\boldsymbol{\pi}'_{-i}} V_i(\pi'_i, \boldsymbol{\pi}'_{-i}, h) \tag{3}$$

for each policy π'_i for player 1 and each history $h \in H$. It is known that attention in SSP games can be restricted to stationary pure policies [].

[1]The notions of proper and improper policies are undefined for player 2.

3 Framework

Definition 1. *An agent-body-effector model is a tuple* $(N, B, E, f_{B,N}, f_{E,B})$ *where:*

- $N = \{1, \ldots, n\}$ *is a finite set of agents*
- B *is a finite set of bodies with* $f_{B,N} : B \to N$ *a surjection from bodies to agents*
- E *is a finite set of effectors with* $f_{E,B} : E \to B$ *a surjection from effectors to bodies and* $A(e)$ *the non-empty (possibly infinite) set of actions available to* $e \in E$

Corollary 1. $|N| \leq |B| \leq |E|$.

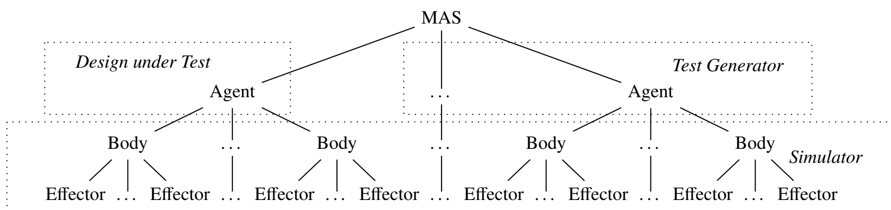


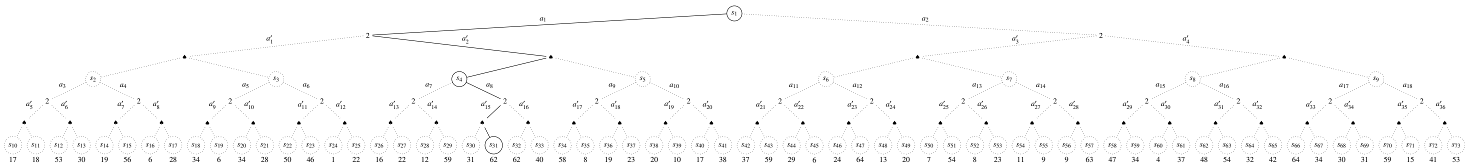
Figure 1: Agent-body-effector model

The set of bodies associated with agents $N' \subseteq N$ is defined as $B(N') = \{b \in f_{B,N}^{-1}(i) \mid i \in N'\}$. The set of effectors associated with bodies $B' \subseteq B$ is defined as $E(B') = \{e \in f_{E,B}^{-1}(b) \mid b \in B'\}$. Thus, the set of bodies B (resp. agents N) in an agent-body-effector model induces a partition of the set of effectors E, as illustrated in Figure 1. The set of (multi-)actions available to effectors $E' \subseteq E$ is defined as $A(E') = \bigtimes_{e \in E'} A(e)$. Thus, there exists a non-empty set of (multi-)actions available to each body (resp. agent).

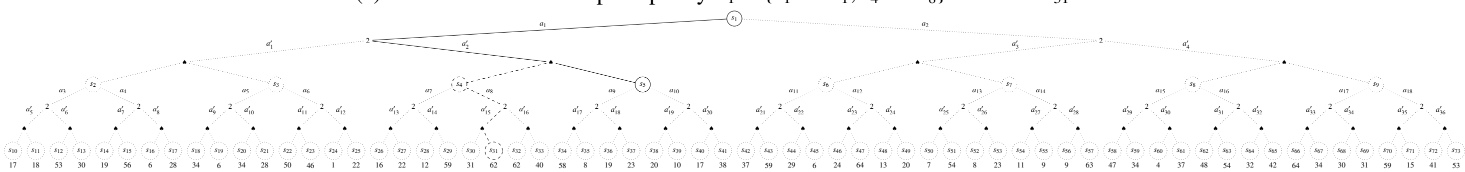
Definition 2. *Let* $(N, B, E, f_{B,N}, f_{E,B})$ *be an agent-body-effector model. An SSP game* $(N, S, A, T, \boldsymbol{R}, G)$ *is a test generation game if:*

- $N = \{1, 2\}$ *with* 1 *the tester agent and* 2 *the ego agent*
- $A_i = A(E(B(\{i\})))$ *is the set of actions available to agent* $i \in N$
- $G \subseteq S$ *is an assertion (or the precondition of an assertion)*
- $(N, S, A, T, \boldsymbol{R}, G)$ *is solvable*

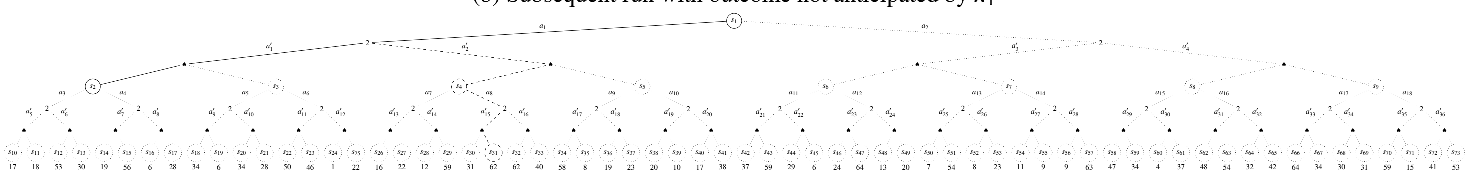
In the context of a test generation game, a test is an optimal policy (e.g. an expectiminimax policy) for player 1. An example of such a policy is shown in Figures 2 and 3. In other words, a test guarantees the triggering of an assertion within a finite number of timesteps while maximizing reward (or minimizing cost) for the tester agent, regardless of actions taken by the ego agent or of any chance outcomes. A test is also applicable (and optimal) for any initial state of a simulation run.



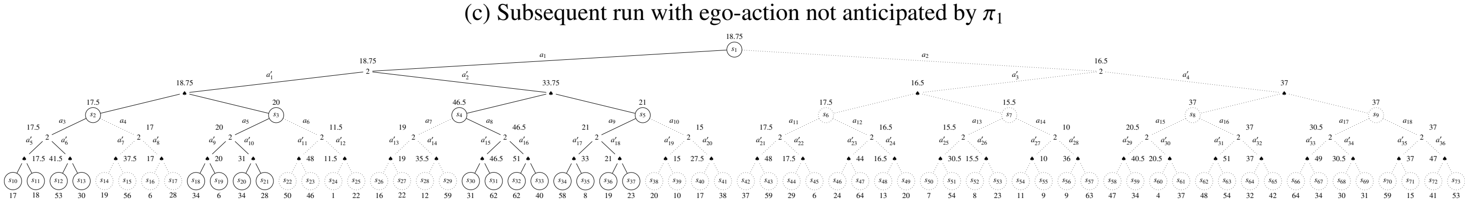
(a) Initial run with incomplete policy $\pi_1 = \{s_1 \mapsto a_1, s_4 \mapsto a_8\}$ such that $s_{31} \in G$



(b) Subsequent run with outcome not anticipated by π_1



(c) Subsequent run with ego-action not anticipated by π_1



(d) Closed expectiminimax policy $\pi_1^* = \{s_1 \mapsto a_1, s_2 \mapsto a_4, s_3 \mapsto a_5, s_4 \mapsto a_7, s_5 \mapsto a_{10}\}$ such that $\{s_{10}, \ldots, s_{13}, s_{18}, \ldots, s_{21}, s_{30}, \ldots, s_{37}\} \subseteq G$

Figure 2: SSP game with initial state $s_1 \in S$ (2 denotes ego decision nodes, ♠ denotes chance nodes, policies are stationary and pure)

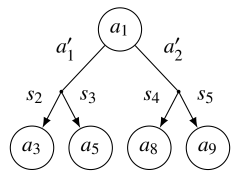
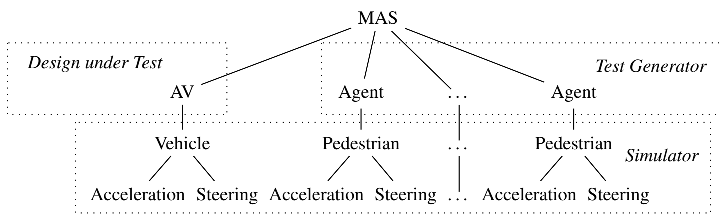
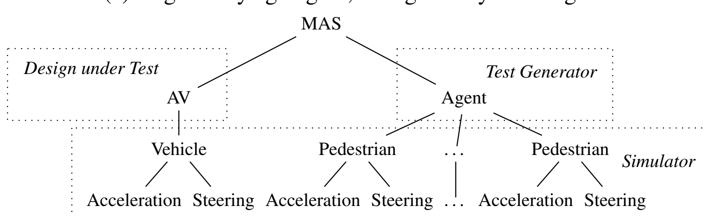


Figure 3: Visualisation of π^* from Figure 2

4 Experiments



(a) single-body ego agent, *n* single-body tester agents



(b) single-body ego agent, 1 multi-body tester agent

Figure 4: Agent-based test generation in CAV-GYM:PEDESTRIANS

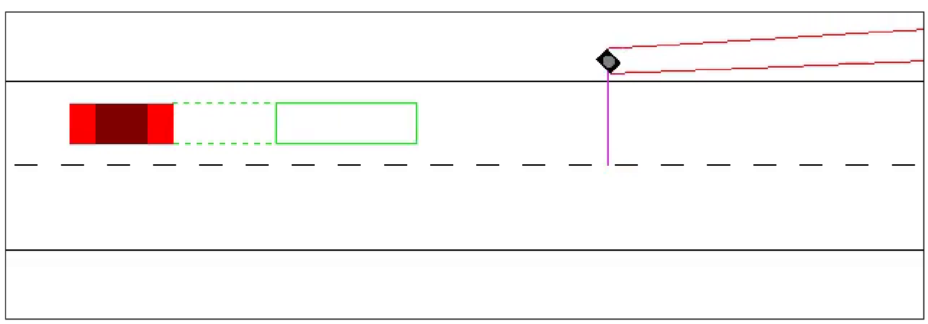


Figure 5: CAV-GYM:PEDESTRIANS

References

Appendix: Kinematics

A state is a tuple $s_i = (p_i, v_i, o_i)$ where $p_i \in \mathbb{R}^2$ is a position (point) denoted $p_i = (p_i^x, p_i^y)$, $v_i \in [v_{\min}, v_{\max}]$ is a velocity with $v_{\min}, v_{\max} \in \mathbb{R}^{\geq 0}$ such that $v_{\min} \leq v_{\max}$, and $o_i \in (-\pi, \pi]$ is an orientation (in radians). Let $b \in \mathbb{R}^{\geq 0}$ be a wheelbase constant. The positions of front and rear wheels $f_i, r_i \in \mathbb{R}^2$ in state $s_i = (p_i, v_i, o_i)$ are:

$$f_i^x = p_i^x + \frac{b}{2}\cos o_i \tag{4}$$

$$f_i^y = p_i^y + \frac{b}{2}\sin o_i \tag{5}$$

$$r_i^x = p_i^x - \frac{b}{2}\cos o_i \tag{6}$$

$$r_i^y = p_i^y - \frac{b}{2}\sin o_i \tag{7}$$

A body has two effectors: throttle and steering. An action is a tuple $a_i = (t_i, e_i)$ where $t_i \in [t_{\min}, t_{\max}]$ is a throttle with $t_{\min}, t_{\max} \in \mathbb{R}$ such that $t_{\min} \leq t_{\max}$, and $e_i \in [e_{\min}, e_{\max}]$ is a steering angle (in radians) with $e_{\min}, e_{\max} \in (-\frac{\pi}{2}, \frac{\pi}{2})$ such that $e_{\min} \leq e_{\max}$. Let $\lambda \in \mathbb{R}^{>0}$ be a time resolution constant. If action $a_i = (t_i, e_i)$ is executed in state $s_i = (p_i, v_i, o_i)$ such that $e_i = 0$, then the successor state is $s_{i+1} = (p_{i+1}, v_{i+1}, o_{i+1})$ where:

$$p_{i+1}^x = p_i^x + v_i\lambda\cos o_i \tag{8}$$

$$p_{i+1}^y = p_i^y + v_i\lambda\sin o_i \tag{9}$$

$$v_{i+1} = \min\{v_{\max}, \max\{v_{\min}, v_i + t_i\lambda\}\} \tag{10}$$

$$o_{i+1} = o_i \tag{11}$$

If action $a_i = (t_i, e_i)$ is executed in state $s_i = (p_i, v_i, o_i)$ such that $e_i \neq 0$, then the successor state is $s_{i+1} = (p_{i+1}, v_{i+1}, o_{i+1})$ where:

$$c_i^x = r_i^x - \frac{b}{\tan e_i}\sin o_i \tag{12}$$

$$c_i^y = r_i^y + \frac{b}{\tan e_i}\cos o_i \tag{13}$$

$$\theta_i = \frac{\operatorname{sgn}(e_i)v_i\lambda}{\sqrt{(c_i^x - p_i^x)^2 + (c_i^y - p_i^y)^2}} \tag{14}$$

$$= \frac{\operatorname{sgn}(e_i)2v_i\lambda}{\sqrt{b^2\left(1 + \frac{4}{(\tan e_i)^2}\right)}} \tag{15}$$

$$p_{i+1}^x = c_i^x + (p_i^x - c_i^x)\cos\theta_i - (p_i^y - c_i^y)\sin\theta_i \tag{16}$$

$$p_{i+1}^y = c_i^y + (p_i^x - c_i^x)\sin\theta_i + (p_i^y - c_i^y)\cos\theta_i \tag{17}$$

$$v_{i+1} = \min\{v_{\max}, \max\{v_{\min}, v_i + t_i\lambda\}\} \tag{18}$$

$$o_{i+1} = \operatorname{arctan2}(\sin(o_i + \theta_i), \cos(o_i + \theta_i)) \tag{19}$$

The point c_i is the centre of rotation for the given state-action pair (with non-zero steering action) and θ_i is the corresponding turn angle. The body kinematics specified by Equations 4–19 are illustrated in Figure 6. If action $a_i = (t_i, e_i)$ is executed in state $s_i = (p_i, v_i, o_i)$ such that $v_i > 0$ and results in successor state $s_{i+1} = (p_{i+1}, v_{i+1}, o_{i+1})$, then it follows that:

$$t_i = \frac{v_{i+1} - v_i}{\lambda} \tag{20}$$

$$\theta_i = \operatorname{arctan2}(\sin(o_{i+1} - o_i), \cos(o_{i+1} - o_i)) \tag{21}$$

$$e_i = \operatorname{sgn}(\theta) \arctan\left(2b\sqrt{\frac{\theta^2}{4v_i^2\lambda^2 - b^2\theta^2}}\right) \tag{22}$$

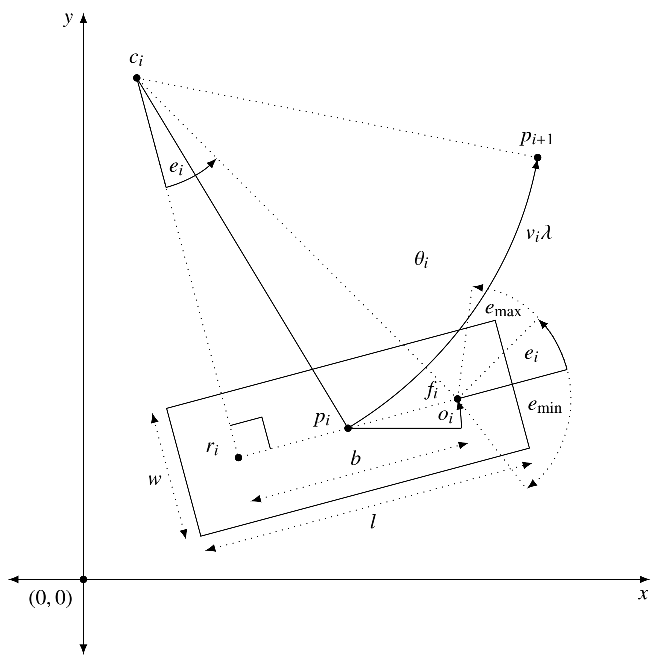


Figure 6: Body kinematics

Appendix: Algorithms

To do: Account for discretisation of continuous action spaces within each agent type (although, strictly speaking, QLearningAgent is the only agent type that requires a finite action space).

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 $\textsc{ChooseAction}(s)$
    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 $\textsc{ChooseAction}(s)$
    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 π : S → A, trigger φ ⊆ S, terminator ψ ⊆ S
function ChooseAction(s)
    if π is active then
        if s ∈ ψ then set π as inactive else return π(s)
    if s ∈ φ then
        set π as active
        return π(s)
    return ∅
```

Algorithm 5: ProgrammedElectionAgent

```
persistent: programmed behaviour π : S → A, terminator ψ ⊆ S, coordinator c
function ChooseAction(s)
    if π is active then
        if s ∈ ψ then set π as inactive else return π(s)
    if elected by c then
        set π as active
        return π(s)
    return ∅
```

Algorithm 6: QLearningAgent

```
persistent: learning rate α ∈ [0, 1], discount factor γ ∈ [0, 1), exploration rate ϵ ∈ [0, 1],
            feature f_j : S × A → ℝ with weight w_j ∈ ℝ for j = 1, ..., m
function ChooseAction(s)
    with probability ϵ do
        return random choice from A(s)
    return random choice from argmax_{a∈A(s)} QValue(s, a)
procedure ProcessFeedback(s, a, r, s')
    q ← (r + γ · max_{a'∈A(s')} QValue(s', a')) − QValue(s, a)
    for each feature f_j do
        w_j ← w_j + α · q · f_j(s, a)
function QValue(s, a)
    return Σ_{j=1}^{m} f_j(s, a) · w_j
```