A Multi-Agent Systems Approach to Test Generation for Simulation-based Autonomous Vehicle Verification

Greg Chance[1], Abanoub Ghobrial[1], Severin Lemaignan[2], Tony Pipe[2], Kerstin Eder[1]

***Abstract*—Simulation-based verification is beneficial for assessing otherwise dangerous or costly on-road testing of autonomous vehicles (AV). This paper addresses the challenge of efficiently generating effective tests for simulation-based AV verification using software testing agents. The multi-agent system (MAS) programming paradigm offers rational agency, causality and strategic planning between multiple agents. We exploit these aspects for test generation, focusing in particular on the generation of tests that trigger preconditions of assertions. On the example of a key assertion we show that, by encoding a variety of different behaviours respondent to the agent's perceptions of the test environment, the agent-based approach generates twice as many effective tests than pseudo-random test generation, while being both scalable and efficient. Moreover, agents can be encoded to behave naturally without compromising the effectiveness of test generation. Our results suggest that generating tests using testing agents significantly improves upon random and simultaneously provides more realistic driving scenarios.**

***Index Terms*—Test Generation, Simulation, Autonomous Driving, Verification, Multi-Agent System, Testing Agent**

I. Introduction

VERIFICATION is the process used to gain confidence in the correctness of a system with respect to its requirements [6]. Testing is a technique that can be used to achieve this by showing that the intended and actual behaviours of a system do not differ and detecting failures against the requirements in the process [32].

Using simulation to test autonomous driving functions in safety critical scenarios benefits from full control over the environment, where road layouts, weather conditions, a variety of road users and other driving scenario parameters can be directed to achieve specific test targets. These tests may aim to provide evidence to regulators of the functional safety of the vehicle or its compliance with commonly agreed upon road conduct, such as the Vienna convention [33], typically implemented at national level as a set of rules [29], road traffic laws and penalties [30].

Verification of complex systems is challenging. In semiconductor design, for example, it has long been recognised that verification can take up to 70% of the design effort [3], with the largest part still being achieved with simulation-based techniques. The testbench is the code used to drive a stimulus sequence into the Design under Verification (DUV) while observing input protocols. It also records coverage and checks the DUV's response. The testbench provides a completely closed environment from the DUV's perspective. Simulators are used to execute testbenches. Automation plays a critical role in achieving verification targets efficiently and effectively.

Coverage-driven verification is a systematic, goal-directed simulation-based verification method [2] that offers a high degree of automation and is capable of exploring systems of realistic detail under a broad range of environment conditions. Because exhaustive simulation is intractabe due to the vast parameter space, the remaining challenge is in strategically selecting the (ideally smallest set of) test cases that result in the highest level of confidence in the design's correctness. Automating test generation has been the focus of research for decades, giving rise to a variety of coverage-directed stimulus generation techniques that exploit formal methods, genetic programming and machine learning [17].

Compared to semiconductor design verification, AV verification faces even bigger challenges, including automatic test generation. It is well known that few of the valid tests are actually interesting from a verification point of view. Estimates vary, but demonstrating AV safety with a confidence of 95% that the failure rate is at most 1.09 fatalities per 100 million miles driven would take 275 million miles, equivalent to 12.5 years for a fleet of 100 AVs [18]. This figure is based on the number of roads fatalities in the US, and would need to be adjusted taking into consideration local statistics on road safety, e.g. the number of road fatalities per billion vehicle-km in the UK has been half that of the US in 2018 [13]. Ways must be found to test the scenarios of interest without needing millions of miles of driving or billions of miles of simulated driving [22]. In particular, simulation-based testing offers the opportunity to increase the number of otherwise rare events [21] in order to determine whether the AV handles such rare events appropriately.

Considering the AV as a DUV, the challenge is to generate tests that interact with the AV over a period of time, thereby creating an environment in which the AV needs to respond to the received stimulus while making progress towards its destination. As such, the AV can be classed as a *responder* DUV, i.e. a DUV that reacts to lower-level stimulus observed on its interfaces with the surrounding environment in order to maintain legally correct driving behaviour and follow the social norms associated with road traffic. This paper investi-

[1]Abanoub Ghobrial, Greg Chance and Kerstin Eder are with the University of Bristol, Bristol, UK {greg.chance, abanoub.ghobrial, kerstin.eder}@bristol.ac.uk

[2]Severin Lemaignan and Tony Pipe are with the Bristol Robotics Laboratory, University of the West of England, Bristol, UK {severin.lemaignan, tony.pipe}@brl.ac.uk

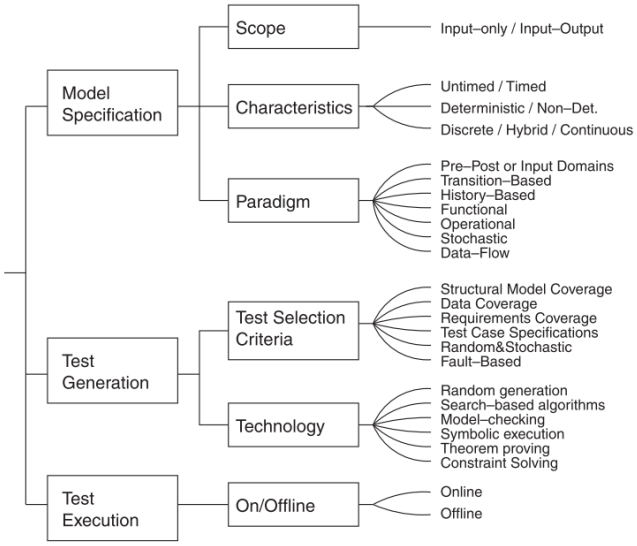

Fig. 1. Taxonomy of model-based test generation, from [32].

gates the benefits of introducing *agency* into the verification environment in order to address the challenges of verifying the responder DUV. Each software agent is tasked with specific goals that aim to achieve verification objectives, e.g. reaching coverage targets. A set of software agents can then be directed to interact, coordinating their behaviour in response to the AV's observed actions in order to increase the likelihood of rare events occurring during simulation to reach coverage targets faster. Our key research question is: what are the benefits of using agent-based test generation for the verification of AVs in simulation? In particular, we are interested in how agent-based test generation compares to pseudo-random test generation techniques wrt. the following criteria for a 'good' test case, which are inspired by [11]: *effectiveness* in detecting defects, *efficiency* in minimising the number if tests required to achieve verification goals, *economy* in terms of resource usage during verification as well as during the analysis of the verification results, and also *robustness* towards changes. Our results suggest that generating tests using testing agents significantly improves upon random and simultaneously provides driving scenarios that are more realistic than those obtained by random test generation.

We regard agent-based test generation as a contribution to the well-established model-based test generation paradigm. A taxonomy of model-based test generation from [32] is given in Figure 1. The agent-based technique creates two new entries in that taxonomy, a new Paradigm under Model Specification, *Agent-based*, and a new Technology under Test Generation, *Agency*, which includes reactive reasoning, causality and strategic planning between multiple agents. In this paper we use an agent-based model to specify the test environment of the AV. Agency is given to the key dynamic entities in the test environment that interact with the AV, in our case pedestrians. Agency, implemented through a belief-desire-intention agent interpreter [8] is then employed to generate tests based on the multi-agent system that represents the test environment.

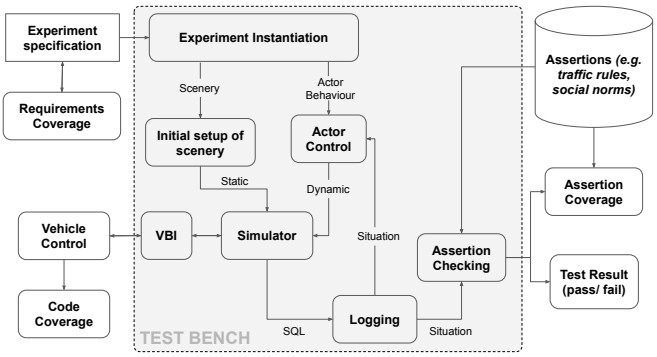

Fig. 2. Testbency Architecture.

Note that other test generation techniques, such as random generation, which we use as baseline for evaluation, or model checking [7], can also be applied to an agent-based model.

This paper is structured as follows. In the next section we introduce the terminology we will adopt throughout the paper and present the testbench architecture used in our experiment. In Section III-A we review related work on test generation for simulation-based AV verification and introduce the basics of multi-agent systems. Section IV presents our case study, which is centred around test generation for a collision avoidance scenario. Results are presented and discussed in Section V. We conclude in Section VI and give an outlook on future work.

II. Terminology and Testbench Architecture

We will adopt the terminology defined in [31], where *scene* refers to all static objects including the road network, street furniture, environment conditions and a snapshot of any dynamic elements. Dynamic elements are the elements in a scene whose actions or behaviour may change over time, these are considered actors and may include other road vehicles, cyclists, pedestrians and traffic signals. A *situation* is defined as the subjective conditions and determinants for behaviour at a particular point in time. The *scenario* is then defined as a temporal development between several scenes which may be specified by specific goals and values.

The proposed testbench, see Fig. 2, is driven by a specification for the experiment which defines the scene and scenario including all dynamic actors. The experiment or test case specification specifies the test inputs, i.e. the execution conditions for an item to be tested [28]. The vehicle behaviour interface (VBI) connects the AV controller (vehicle control) to the simulator. It provides the simulator with the driving decisions of the AV and forwards updates on the scene to the AV controller. A geospacial database logs the AV and all other actors to enable post-simulation assertion checking.

Given the experiment specification, tests may be generated in a variety of ways that differ mainly in their effectiveness and efficiency. Manually generated tests are typically effective in achieving verification objectives, but are considered expensive due to the high engineering cost involved. Random methods are usually employed at the early stages of testing to build up

coverage quickly. They suffer, however, from a high number of invalid tests being generated and, even when constrained to produce only valid tests, the tests generated are often not interesting wrt. the verification objectives, in our case this refers to exercising the collision-avoidance decision making logic of the AV. Model-based test generation offers an alternative that produces valid and interesting tests at the cost of developing a model that faithfully encodes the behaviour of the test environment. This model can then be explored in a variety of ways, see Figure 1. Our paper explores how the agency that is naturally present in the multi-agent system that represents the dynamic actors in a scene can be used for model-based generation of test scenarios in the context of simulation-based AV verification.

One could argue that the simulation environment presented to the AV should be as close to the real world it seeks to emulate as the most realistic proving ground. Such efforts seek to reduce the *reality gap*, providing the most likely scenarios and actor behaviour. This can be achieved by monitoring real traffic scenarios, e.g. tracking individual vehicles, cyclists and pedestrians, and building behavioural models for each of the tracked entities [5]. On the other hand, the creation of edge cases, i.e. events that are rare under normal circumstances, yet critical for gaining confidence in the correct behaviour of the AV, is important to reach verification objectives in a timely manner. This is exactly what the agent-based test generation approach introduced in this paper is aimed at.

III. Background

A. Related Work

An overview of the challenges currently faced in software testing of AVs is given by Koopman, highlighting the importance of fault injection into the testing domain [20] and how to decide what aspects of the system need to be tested in the areas of operational design domains, event detection and vehicle manoeuvres [19]. Describing a driving scene in natural language has been shown to be an effective framework for scenario generation. This can also be automated based on formalised rules and knowledge [4] or even evolved from ISO safety standards [24]. Hallenbach et al. take the approach to test generation of developing composite metrics for traffic quality and using them to determine if a scenario is 'critical' or not and therefore worth exploring further [16]. Generating targeted test cases can be a more efficient way to achieve verification objectives. For example, Mullins et al. [25] describe a test generation method that is focused on exploring the transitions between decision making performance modes of the AV, with the aim to find tests that exercise the complete decision making logic. Saigol et al. [27] discuss the need for *smart actor control* which they expect may lead to interesting scenarios when these interact with the AV a testbench framework for automated testing. Rocklage et al. [26] approach the problem of test generation from the viewpoint of the other traffic participants, using a trajectory planner to ensure coverage over a fixed list of scene parameters. Exploiting agency for test generation, however, has not yet been investigated.

B. Multi-Agent Systems

Georgeff and Lansky were key in the development of the belief-desire intention (BDI) agent programming approach [15] and also in the early work of multi-agent systems [14]. In a multi-agent system, multiple intelligent agents interact in order to collectively solve problems that are too difficult or impossible for individual agents to achieve. A multi-agent system includes a set of software agents and their environment. Agents can be equipped with varying degrees of agency, starting from passive agents, such as obstacles, including active agents that have goals of low complexity, such as pedestrians that act as part of a group, to rational agents that are capable of reasoning based on a cognitive model of the situation, their perception of the surrounding environment, and a set of rules they can use for strategic planning and communication with other agents in the system. The individual agents are considered *autonomous* and in control of their behaviour within the multi-agent system as they pursue their goals. This results in self-organisation and self-direction towards achieving a common goal.

Combining the BDI framework with an automated test generation approach leads to the concept of a software agent capable of generating tests. Such intelligent, agent-based test generation has first been applied in the human-robot interaction domain [1], where a coverage-driven test generation approach was supplemented with reinforcement learning to improve the efficiency and effectiveness of testing the critical part of a robot-to-human handover task within a collaborative manufacturing scenario. Test agents have also been proposed by Enoiu et al. [10] for regression testing, although they are used more for test selection from a library of tests, rather than for test generation. These agents use inter-agent messaging to decide what tests to execute and with what prioritisation.

IV. Case Study

This section describes a case study to explore the proposed agent-based test generation method.[1] Our aim is to generate tests that exercise an assertion that requires the AV to avoid collisions with other road users, provided that an entity, a pedestrian in this case, intrudes in the path of the AV in such a way that a collision can be avoided by the AV either by braking or manoeuvring. In Fig. 3 this is illustrated by the 'Precondition Zone', i.e. the area of interest in which the assertion is activated. Intrusions that fall within the 'Stopping Distance' of the AV, 12m at 30km/h [29], are not considered interesting as they result in unavoidable collisions and tests of this type are of limited use. A single assertion is chosen so that we can study the fundamental properties of agent-based test generation in a simple setting. Moreover, we do not consider the test result (pass or fail), i.e. the behaviour of the AV in response to the assertion being activated.

A. Test Environment

The testing environment, see Fig. 3 is a straight two-lane road 99m long and each lane is 6m wide. Pavements are on

[1]The code used is available at github.com/TSL-UOB/CAV-MAS.

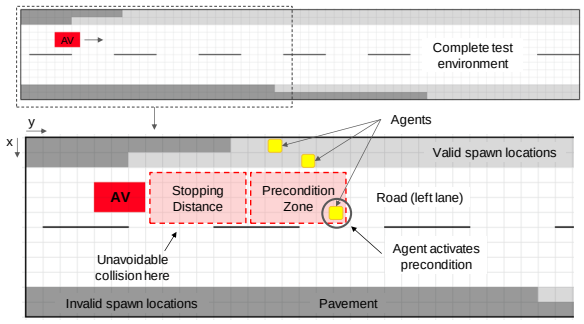

Fig. 3. Test environment at full scale (upper) and in detail (lower) including valid and invalid spawn locations for the pedestrians and the position and direction of the AV.

both sides of the road which are three meters in width giving a total size is 18m x 99m. The AV velocity is 9.0 m/s (equivalent to ~20 mph) and the pedestrian velocity is 1.4 m/s (equivalent to ~3 mph). The map is discretised with 1.5 meter resolution for simple division into the AV and pedestrians velocities. Thus, in the discretised world the AV velocity is 6 cell/sec and the pedestrians velocity is 1 cell/sec. The total number of map cells is 12 x 66 = 792 cells.

The AV travels along the left hand lane of the road starting at cell $y = 0$ and travelling to the end, cell $y = 66$. If the assertion is triggered or the AV reaches the end of the road then the test is restarted. The AV occupies an area 4.5x3m equivalent to 4x2 cells. There are no other vehicles and the right hand side of the road is unoccupied.

The environment is initialised with the agents spawned randomly on any valid pavement area. When the environment is reset new random locations are set for the agents. Control over random locations are set using a fixed seed based on the experiment number and so initial spawn locations are repeatable which ensures valid comparison between experiments.

The precondition is activated if an agent is found in the *precondition zone* of the AV which extends 6 cells forwards of the 'stopping distance' of the vehicle, Fig.3. Some areas of the environment are impossible for the agents to intersect the AV and are therefore invalid spawn locations, shown in Fig. 3.

B. Test Agents

To assess if the agent-based model of test generation is effective it is compared to random methods and repeated sufficiently for statistical power. A simple scene is chosen containing a straight road with a pavement either side represented in a grid world with a finite (1.0s) time interval. For this example pedestrian and test agent are synonymous. At the start of a test pedestrians are randomly located onto the pavement area and the AV is represented as a vehicle that drives at constant speed in a straight line and will not brake or turn. The test agents have differing behaviours from simple random to more complex and directed, which is the focus of this section.

The different agent behaviours can be split into two main classes: random and directed, see Table I. For the random class, the *random* is where the test agent can make any random movement at each tick where the available actions are: do nothing (stand still), move up, down, left or right. For the *constrained random* the pedestrian is instantiated walking along the pavement before the simulation starts and has only one action; to randomly choose when to cross the road. The constrained random is included so that a comparison between crossing the road at a targeted or random time can be made.

The second class of behaviours are based on perceptions and these agents are initialised walking along the pavement as in constrained random. The *proximity* behaviour instructs the agent to cross the road when the AV is within a certain radius. Using the *intersect* behaviour, the agent calculates an intersection time and chooses to cross the road if the pedestrian can reach the intersection point before the AV has passed. The *election* behaviour ranks the probability of each agent to enact the intersect behaviour and elects the agent with the highest likelihood of success.

The *proximity* behaviour describes an agent, for this assertion, with suicidal tendencies that will step into the road whenever a car is nearby but is a highly improbably model of pedestrian actions. The *proximity* behaviour also suffers from the 'zombie effect' where all nearby pedestrians are drawn to the passing vehicle which may lead to the coverage point required but through an improbable scenario. The *intersect* behaviour is a more refined version of the proximity behaviour but may still result in multiple agents attempting to intersect the AV simultaneously whereas the *election* behaviour should limit this to a single agent.

The number of agents is also a factor to consider for this demonstration, too few and the likelihood of an agent triggering the assertion will be low and entirely dependent on the initial starting position in the test environment due to differences in speed. The maximum number of agents could be considered all the available grid locations (1.5m spacing) on the pavement without overlap. As the number of agents, nA, tends to this maximum the probability of hitting the coverage point using a random class increases significantly as more agents appear in the road with an increase in computation expense. As nA increases using more intelligent behaviour methods, suitable agents to trigger the assertion precondition should be found more readily resulting in shorter and hence more efficient tests. The range of nA explored was 1-20.

C. Scoring

Each agent behavioural method will be repeated 1000 times and the number of successful tests counted. A successful test is generated when a pedestrian intrudes into the *precondition zone* of the AV that may potentially cause an intersection. In an attempt to impose more realistic behaviour on the agents, a scoring system is used that penalises certain actions. A living cost is imposed on the agents to promote shorter tests and a penalty is given for agents that are in the road.

The scoring system is:

- A reward of 100 is given if the agent enters the *precondition zone*.
- A penalty of -1 for each time step.
- A penalty of -5 for each time step spent in the road.

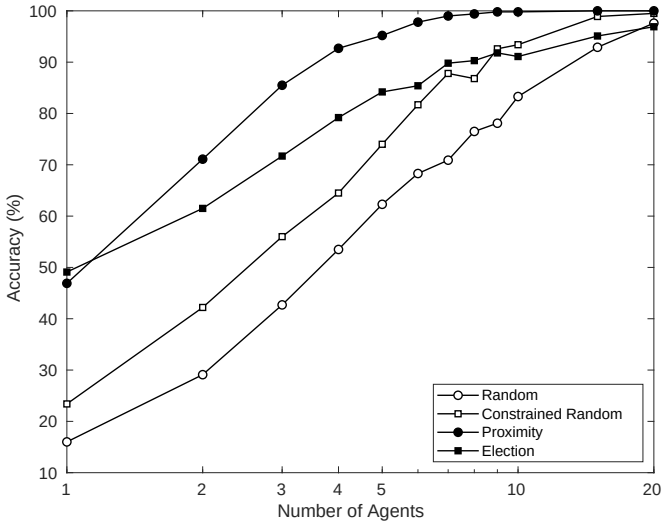

Fig. 4. Accuracy of agent to generate a test case that meets the precondition.

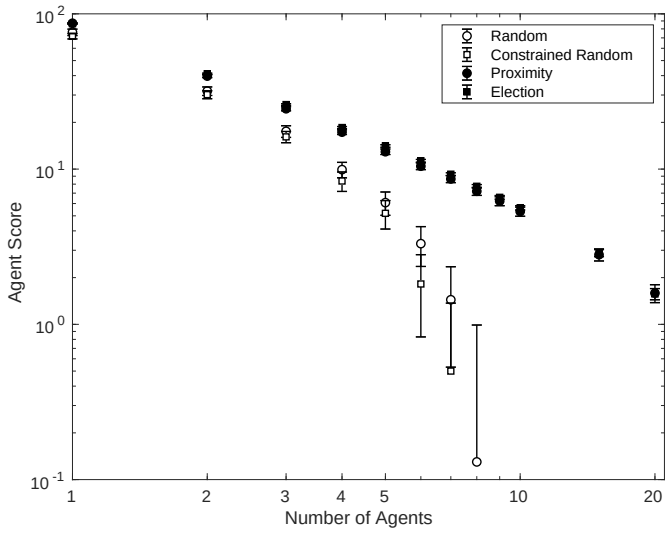

Fig. 5. The average agent score for successful tests with 95% confidence intervals for each agent type.

V. Results

The results are evaluated based on 5 criteria; Accuracy is the ratio of successful tests the agents generated over all tests, Score is a measure of how natural the agents behaved, Combined Score combines accuracy with score, and Time is how long the agents took to generate the test in both simulation ticks and wall clock time. Both Score and Time have distributions associated with them and therefore confidence intervals are provided.

A. Test Accuracy

Test accuracy, defined as the number of tests that have activated the precondition for the assertion as a ratio of all tests, are shown for each agent type in Fig. 4. The *random* and *constrained random* have lower accuracy compared to the directed agents when $nA < 10$. For high nA the accuracies converge mostly due to a saturation of agents. Fig. 4 also shows that for $nA = 1$ a directed agent outperforms a random agent by more than 2:1. The *election* agent generates a slightly higher accuracy than the *proximity* agent but only by 2.2%. However, for $nA = 2$ and above the *proximity* agent has a 10% increase over the election agent which is because this agent has multiple attempts to trigger the precondition whereas the election agent only has one.

B. Test Score

For tests that activate the precondition the average agent score is compiled including 95% confidence interval range, see Fig. 5. The maximum theoretical score of any agent is 94 which includes 100 points for a successful test subtracting a living cost of 1 and road penalty of 5 but requires the agent to spawn adjacent to the *precondition zone*. For a single agent scores are similar as only successful tests are included in the scoring. As the number of agents increases the random class diverge from the directed class and the variance in the random agent score increases. This indicates that the directed agents have more realistic actions than the random class.

As nA increases the random agents are more often found in the road and hence average score drops, whereas the directed agents are only found crossing the road when they deem fit and are on the pavement at all other times keeping the score much higher and resulting in more realistic test cases. No significant difference between the directed class is seen in the score even with a high number of agents indicating both represent a similar level of realistic behaviour, at least within the limited scope of this example.

C. Combined Score

As the score above (section V-B) only shows tests that meet the precondition, it can be easy to interpret these results as overly optimistic so a *combined score* is provided which is given by (score * accuracy / 1000). This combined measure promotes scores that are attached to high accuracies, describing agents that can generate useful tests with natural pedestrian behaviour. Time could also ave been included as a denominator here but is already accounted for in the living cost penalty. The normalised results, Fig 6, show that the directed agents are over twice as effective within this new combined definition than random agents for $nA = 1$ although this advanced drops rapidly with increasing agent numbers to reach a steady gap of around 12%. The *constrained random* agent outperforms the *random* for $nA < 4$ beyond which there is little difference. So if random is your choice, you might as well just use completely random unless you want to use low agent numbers. The *election* agent has the highest combined score for $nA = 1$ but higher agent numbers are favour the *proximity* agent up to $nA = 4$ beyond which there is no notable difference between the directed agent types.

D. Agent CPU Time

For each test generated that met the precondition, the CPU time taken to execute the agent action was averaged over the 1000 runs and compared across different agent types and

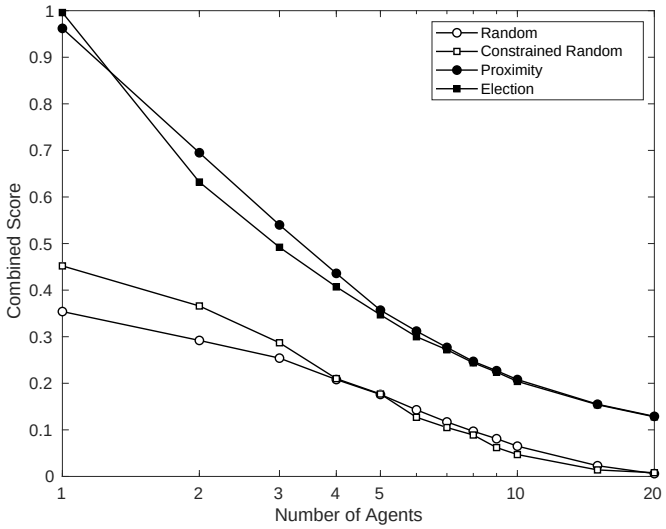

Fig. 6. The combined accuracy and score for each agent type.

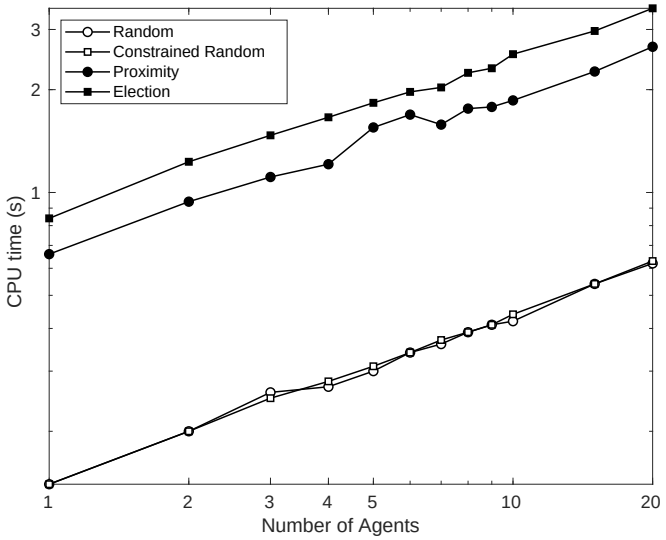

Fig. 7. The CPU time taken to execute the agent actions averaged over 1000 runs.

numbers, Fig. 7. This is essentially comparing the resources required to execute the actions of different agent types. The more complex agents have additional CPU overhead compared to random but this difference is relatively unchanged with increasing agent numbers.

E. Time to Generate Test

For each successful test generated, the number of simulation ticks was averaged over 1000 runs and compared across different agent numbers, Fig. 8. The directed agents show improved performance over the random class for $nA = 1$ by approximately a single simulation tick and this trend continues as agent numbers increase. By $nA = 20$ the directed agents are 1.85 simulation ticks faster than the random agents. Overall the *proximity* agents find tests in the shortest number of simulation steps.

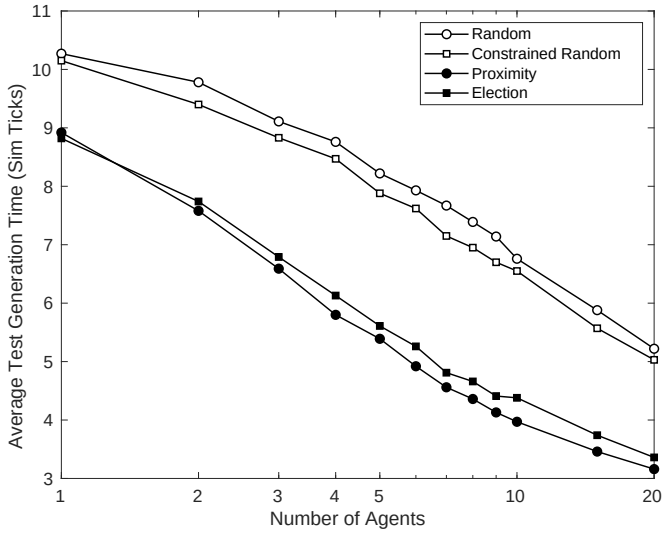

Fig. 8. The average time taken for an agent to find a successful test, t_g.

F. Results Analysis

The results are assessed against the criteria of a 'good' test case as described in [11].

- *Effective*: How effective the method is at generating test cases that detect faults with the responder. The Accuracy metric, defined above, shows how often the agent generates a test case that satisfies the precondition of the assertion and will therefore reveal, or is likely to reveal, defects of the DUV. Fig. 4 shows that a small number of directed agents are around twice as effective as random ones and over three times as effective as a single agent.
- *Exemplary*: An exemplary test case will test the DUV against multiple assertions, an aspect of our future work, see Section VI.
- *Economic*: How costly the test case is to run, analyse and debug. The resource cost (CPU time) to execute the agent actions, Fig 7 is 4-5 times higher for the directed agents than random, see heading t_c in Table I for $nA = 3$. This may be a significant factor in the decision to use an intelligent agent unless abundant CPU resources are to hand. However, the test generation time (t_g), Fig 8, indicates that on average the directed agents find successful test cases faster than random in terms of simulation ticks. Therefore, although more resource is allocated to the directed agents, overall they find test cases faster due to better efficiency, i.e. better return on investment (ROI).
- *Evolvable*: The maintenance required to adapt the test case to software changes or *scenes* in our case. In the case study each test generated is of a different scene as the agent position is randomly generated each time. Therefore the directed agents show adaptability to new scenes and this attribute is accounted for within the accuracy and could be easily shown to succeed with different road layouts and vehicle speeds, see Section VI.

The results show that by nearly all the metrics and parameter combinations discussed, (accuracy, natural behaviour score,

TABLE I
TEST AGENT SUMMARY TABLE SHOWING DESCRIPTION AND NUMBER OF LINES OF CODE (LOC) FOR EACH AGENT AND SAMPLE OF RESULTS: ACCURACY, COMBINED SCORE (s_c), CPU TIME (t_c) AND TEST GENERATION TIME (t_g) FOR $nA = 3$.

Behaviour	**Agent Description**	LOC	Accuracy (%)	s_c (points)	t_c (s)	t_g (ticks)
Random	Randomly perform action (up, down, left, right, stop).	23	42.7	0.254	0.09	9.11
Constrained Random	Walk along pavement, randomly cross the road	82	56.0	0.287	0.08	8.83
Proximity	Walk along pavement, Cross road when AV in range	86	85.5	0.540	0.37	6.79
Election	As in Proximity but elect a single agent to cross	235	71.7	1.470	0.49	6.59

test generation time) the directed agents outperform the random agents. This shows that even a small amount of direction can be a distinct advantage over random techniques.

But what was the cost of developing these agent behaviours and is there a limited pay-off to gain significantly superior intelligence for really complex agent behaviour? Comparing the lines of code for each agent can help to weigh the investment in agent complexity, see lines of code (LOC) in Table I. This shows that for a small investment the *random* agent with 23 lines of code can be improved significantly with (86 lines) to improve it's combined score and test generation time (t_g). To improve the combined score further with the *election* agent took 10x the size of the *random* code with lower overall accuracy but better combined score, suggesting this was not worth the investment.

This simple scenario would suggest that the level of agent complexity should be considered carefully as a simpler level of intelligence could be more beneficial than very complex one which may also required additional investment in debugging.

VI. CONCLUSION AND FUTURE WORK

The MAS programming paradigm offers rational agency and strategic planning to software agents that have been exploited in this research for test generation. On a small example we show that, by encoding a variety of different behaviours respondent to the agent's perceptions of the test environment, the agent-based approach generates twice as many effective tests than a pseudo-random approach. Furthermore, agents can be encoded to behave more realistically without compromising their effectiveness. Our results suggest that generating tests using testing agents is a promising avenue of research and has been shown to significantly improves upon random whilst simultaneously providing more realistic driving scenarios.

Future work will consider how agents behave when multiple assertions exist in a variety of different scenes. As discussed in [9], agents could have their goal selection modified based on coverage feedback which fits in with the feedback reward of many such architectures, e.g. MDP [23] and back-propagation for ANN [12]. Abstracting agent perceptions to a feature based representation would ensure the agent state space can scale to large physical maps and is adaptable to new features as more assertions are added. Including personality to agents is also another avenue that could provide insightful as a tuning parameter [34] to explore edge cases.

ACKNOWLEDGEMENT

This research has in part been funded by the ROBOPILOT and CAPRI projects. Both projects are part-funded by the Centre for Connected and Autonomous Vehicles (CCAV), delivered in partnership with Innovate UK under grant numbers 103703 (CAPRI) and 103288 (ROBOPILOT), respectively.

REFERENCES

[1] D. Araiza-Illan, A. G. Pipe, and K. Eder. "Intelligent agent-based stimulation for testing robotic software in human-robot interactions". In: *ACM International Conference Proceeding Series* (2016), pp. 9–16.

[2] D. Araiza-Illan, D. Western, A. Pipe, and K. Eder. "Coverage-Driven Verification —". In: *Hardware and Software: Verification and Testing*. Ed. by N. Piterman. Springer International Publishing, 2015, pp. 69–84.

[3] W. M. Arden. "The international technology roadmap for semiconductors perspectives and challenges for the next 15 years". In: *Current Opinion in Solid State and Materials Science* 6.5 (2002), pp. 371–377.

[4] G. Bagschik, T. Menzel, and M. Maurer. "Ontology based Scene Creation for the Development of Automated Vehicles". In: *IEEE Intelligent Vehicles Symposium, Proceedings*. Vol. 2018-June. 2018, pp. 1813–1820.

[5] F. Behbahani et al. "Learning from demonstration in the wild". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 775–781.

[6] J. Bergeron. *Writing testbenches functional verification of HDL models*. Springer Science Business Media, 2012.

[7] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. "Verifying Multi-agent Programs by Model Checking". In: *Autonomous Agents and Multi-Agent Systems* 12.2 (Mar. 2006), pp. 239–256. URL: https://doi.org/10.1007/s10458-006-5955-7.

[8] R. H. Bordini, J. F. Hübner, and R. Vieira. "Jason and the Golden Fleece of agent-oriented programming". In: *Multi-agent programming*. Springer, 2005, pp. 3–37.

[9] K. Eder, P. Flach, and H. W. Hsueh. "Towards automating simulation-based design verification using ILP". In: *Lecture Notes in Computer Science* 4455 LNAI (2007), pp. 154–168.

[10] E. P. Enoiu and M. Frasheri. "Test Agents: The Next Generation of Test Cases". In: *2nd IEEE Workshop on NEXt level of Test Automation* (2019).

[11] M. Fewster and D. Graham. *Software test automation*. Addison-Wesley Reading, 1999.

[12] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson. "Learning to communicate with deep multi-agent reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 2137–2145.

[13] I. T. Forum. *Road Safety Annual Report*. https://www.itf - oecd . org / sites / default / files / docs / irtad - road - safety - annual - report - 2018_2.pdf. Accessed: 2019-11-26. 2018.

[14] M. Georgeff. "Communication and interaction in multi-agent planning". In: *Readings in distributed artificial intelligence*. Elsevier, 1988, pp. 200–204.

[15] M. P. Georgeff and A. L. Lansky. "Reactive reasoning and planning." In: *AAAI*. Vol. 87. 1987, pp. 677–682.

[16] S. Hallerbach, Y. Xia, U. Eberle, and F. Koester. "Simulation-Based Identification of Critical Scenarios for Cooperative and Automated Vehicles". In: *SAE International Journal of Connected and Automated Vehicles* 1.2 (2018).

[17] C. Ioannides and K. I. Eder. "Coverage-Directed Test Generation Automated by Machine Learning – A Review". In: *ACM Trans. Des. Autom. Electron. Syst.* 17.1 (Jan. 2012), 7:1–7:21.

[18] N. Kalra and S. M. Paddock. "Driving to safety How many miles of driving would it take to demonstrate autonomous vehicle reliability". In: *Transportation Research Part A Policy and Practice* 94 (2016), pp. 182–193.

[19] P. Koopman and F. Fratrik. "How many operational design domains, objects, and events?" In: *CEUR Workshop Proceedings* 2301 (2019), pp. 1–4.

[20] P. Koopman and M. Wagner. "Challenges in Autonomous Vehicle Testing and Validation". In: *SAE International Journal of Transportation Safety* 4.1 (2016), pp. 15–24.

[21] P. Koopman and M. Wagner. "Toward a Framework for Highly Automated Vehicle Safety Validation". In: *SAE Technical Paper Series* 1 (2018), pp. 1–13.

[22] K. Korosec. *Waymo's self driving cars hit 10 million miles*. https://techcrunch.com/2018/10/10/waymos-self-driving-cars-hit-10-million-miles. Accessed: 2019-11-26. 2019.

[23] M. L. Littman. "Markov games as a framework for multi-agent reinforcement learning". In: *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[24] T. Menzel, G. Bagschik, and M. Maurer. "Scenarios for Development, Test and Validation of Automated Vehicles". In: (2018). URL: http://arxiv.org/abs/1801.08598.

[25] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta. "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles". In: *Journal of Systems and Software* 137 (Mar. 2018), pp. 197–215.

[26] E. Rocklage, H. Kraft, A. Karatas, and J. J. Seewig. *Automated Scenario Generation for Regression Testing of Autonomous Vehicles*. Oct. 2017.

[27] Z. Saigol and A. Peters. "Verifying automated driving systems in simulation framework and challenges". In: *25th ITS World Congress* September (2018), pp. 17–21.

[28] I. Standards Board. *IEEE Standard Glossary of Software Engineering Terminology*. 1990, pp. 1–84.

[29] *The Highway Code*. https://www.gov.uk/guidance/the-highway - code. Accessed: 2019-11-25. Department of Transport, UK, 2015.

[30] *UK Road Traffic Act 1988*. http://www.legislation.gov.uk/ukpga/1988/52/contents. Accessed: 2019-10-03.

[31] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer. "Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving". In: *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC* 2015-Oct (2015), pp. 982–988.

[32] M. Utting, A. Pretschner, and B. Legeard. "A taxonomy of model-based testing approaches". In: *Software Testing, Verification and Reliability* 22.5 (2012), pp. 297–312.

[33] *Vienna Convention on Road Traffic*. https://treaties.un.org. Accessed: 2019-10-03.

[34] A. Zoumpoulaki, N. Avradinis, and S. Vosinakis. "A Multi-Agent Simulation Framework for Emergency Evacuations Incorporating Personality and Emotions". In: *Hellenic Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 2010.* 2010, pp. 423–428.