Reliable Decision-Making in Autonomous Vehicles

Gleifer Vaz Alves, Louise Dennis, Lucas Fernandes, and Michael Fisher

1 Introduction

The deployment of Autonomous Vehicles (AV) on our streets is no longer fictional, but is becoming reality. AV technology will represent a significant change in our economy, society, and daily life [21]. There are already several AV tests that are being undertaken and some of them will lead to practical vehicles on our streets quite soon: in Phoenix (USA) the use of a fully driverless taxi service is expected to commence soon [20]; in Singapore's university district there is the world's first self-driving taxi service, which has been operated by NuTonomy since August 2016 [21]; while there are a large number of cars equipped with autonomous driving technology that are expected on the streets of South Korea by 2020 [21].

Consequently, a lot of work is being carried out concerning different stages of an AV design, for example, detection devices, cameras, sensors, intelligent software, decision-making, liabilities, laws and regulations, and so on. Abstractly, the design of an AV can be divided into two main parts: the high-level and the low-levels [23]. The latter are responsible for the sensors, actuators, detection devices and other

Work partially supported by EPSRC projects EP/L024845 and EP/M027309.

G. V. Alves (✉)
UTFPR – Federal University of Technology, Ponta Grossa, Parana, Brazil
e-mail: gleifer@utfpr.edu.br

L. Dennis · M. Fisher
Department of Computer Science, University of Liverpool, Liverpool, UK
e-mail: L.A.Dennis@liverpool.ac.uk; MFisher@liverpool.ac.uk

L. Fernandes
Samsung R&D (SRBR), Sao Paulo, Brazil
e-mail: lucas.f@samsung.com

A. Leitner et al. (eds.), *Validation and Verification of Automated Systems*,
https://doi.org/10.1007/978-3-030-14628-3_10

similar control elements. The former, however, captures the key decision-making capability that the AV must exhibit now that there is no responsible human 'driver'. This high-level decision-making is software responsible for clearly determining the actions that will be invoked at the low-level.

As highlighted by Herrmann et al. [21] achieving the basic 90% of autonomous driving capabilities is not difficult, but the last 10% is the most challenging task. These 10% includes the most difficult traffic scenarios, especially in urban areas, and the possibility of unexpected or emergency situations. It is here that the AV must make the 'correct' decisions, quickly and reliably. Consequently, we focus here on the high-level decision-making process, where an AV decision-maker is modelled as an intelligent agent [18]. Using this agent-based approach we may write high-level plans for describing the AV decisions and actions and, since these plans are transparent and explicit, we can formally verifying some properties related to this agent's behaviour [11], such as *"it is always true the* AV-agent *will stop in the event of an unexpected emergency"*.

In previous work [16] we have created the first version of a *Simulated Automotive Environment* (SAE), where our AV-agent represented a taxi responsible for driving passengers from a start point to a destination point. Through each taxi route, the agent might face different sorts of obstacles that we divided into two classes: *avoidable* and *unavoidable*. When there is an avoidable obstacle it means the agent can potentially find a detour along its route without crashing in to any obstacle. However, if there is no chance to avoid the obstacle, we say there is an unavoidable obstacle. In such cases, the agent should select the obstacle causing the least (physical) damage to the vehicle, whenever it is possible. The taxi agent in SAE is implemented using GWENDOLEN [8] agent programming language, while the model checker AJPF [9] has been applied towards the formal verification of some related properties concerning the behaviour of the taxi agent. Indeed these same tools (GWENDOLEN and AJPF) are also used in the work here presented.

In this paper, we create an extension of the SAE, named SAE-decision-making, which has a similar approach based on a taxi agent, but where we capture the high-level decisions taking by the AV-agent when there are different levels of emergency (classified here by three different colours): (i) yellow (i.e *avoidable obstacle*); (ii) orange (i.e *harsh environment*); or (iii) red (i.e *unavoidable obstacle*).

As we shall see in this article, our agent uses an approach based on the UK suggestions for driverless vehicles (see [14] and [13]), which state that a human controller has liability concerning the actions taken by the AV. As highlighted by Vellinga [25], in the UK, regulations for testing AVs mention the role of a *test operator*. In [13], it is stated that:

> A test operator will be responsible for ensuring the safe operation of the vehicle at all times (. . .). The test operator (. . .), and be able to anticipate the need to intervene and resume manual control if necessary.

So, in our case, the agent implementation will choose, as a safe and reliable procedure, the following: *the AV-agent will release autonomous control and give it*

back to the human (controller), in case of orange and red levels of emergency (and also when the human is indeed ready to take over).

The remainder of this paper is organised as follows. In Sect. 2 we provide the necessary background on agents, their programming, and their formal verification. Next, Sect. 3 presents the formal definitions of both the environment and agent used in our work, in addition to the agent implementation. Section 4 highlights the properties which have been formally verified by using temporal logic, while in Sect. 5 we provide final remarks.

2 Agents and Formal Verification

In this section, we briefly present some background concepts used in this work. Here we use the notion of a *rational agent* (or intelligent agent) introduced by Bratman [5] and described in detail by Rao and Wooldridge [28]. Here, rational agents are software entities that perceive their environment through sensing, build a model of its *belief* about this environment, and then reason about it. Based on its own mental state, such as its *intentions*, a rational agent can then take actions that may change the environment [28]. A rational agent can be implemented in a number of ways, but we choose to utilise the popular BDI (*Belief, Desire and Intention*) architecture [24].

In our work we use a particular BDI programming language, named GWENDOLEN [8], which captures the BDI concepts in a goal-directed Prolog-like language. Agents programmed in GWENDOLEN have beliefs, reasoning rules, goals, and plans. A plan has three main components [8]:

1. trigger: is the event responsible to unleash a given plan.
2. guard: is a condition that should be checked in order to have an applicable plan.
3. body: has a set of actions and/or plans that are supposed to be executed.

The syntax to represent plans is the following,

```
trigger : { guard } < − body
```

And a plan example written in GWENDOLEN can be as follows,

```
+!emergency_plan { B unexpected_emergency } < − stop;
```

Agent programs constructed using GWENDOLEN language can then be verified using the AJPF tool [9]. AJPF is an extension of the *Java Path Finder* (JPF) program model-checker, used for the formal verification of `Java` programs [26]. The AJPF system [4] was specifically conceived to work with BDI agent programming languages, so that the agent code can be formally verified against given properties. These properties are written in a *Property Specification Language* (PSL) which is

itself based on LTL (Linear Temporal Logic). The syntax for property formulae, ϕ, in PSL is given below, where:

- ag refers to a specific agent in the system;
- f is a classical first-order atomic formula;
- and $\Diamond$ ("eventually in the future") and $\Box$ ("always in the future") are standard LTL operators [17].

$$\phi ::=$$

$$\mathsf{B}_{ag}\ f \mid \mathsf{G}_{ag}\ f \mid \mathsf{A}_{ag}\ f \mid \mathsf{I}_{ag}\ f \mid \mathsf{ID}_{ag}\ f \mid \mathsf{P}(f) \mid \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \phi\,\mathsf{U}\,\phi \mid \phi\,\mathsf{R}\,\phi \mid \Diamond\phi \mid \Box\phi$$

Note that,

- $\mathsf{B}_{ag}\ f$ is true if agent ag believes formulae f to be true,
- $\mathsf{G}_{ag}\ f$ is true if agent ag has a goal to make formula f true,
- *and so on* with A representing actions, I representing intentions, ID representing the intention to take an action, and P representing percepts, i.e., properties that are true in the environment.

An example of a PSL formulae is the following (which is based on the example mentioned in Sect. 1):

$$\Box\, \mathsf{B}_{ag}\ \texttt{unexpected\_emergency} \rightarrow \mathsf{A}_{ag}\ \texttt{stop}$$

3 The AV-Agent: Scenario, Plans, and Environment

As previously mentioned in Sect. 1, the SAE-decision-making is mainly concerned with the high-level decisions taken by our AV-agent as illustrated in Fig. 1.

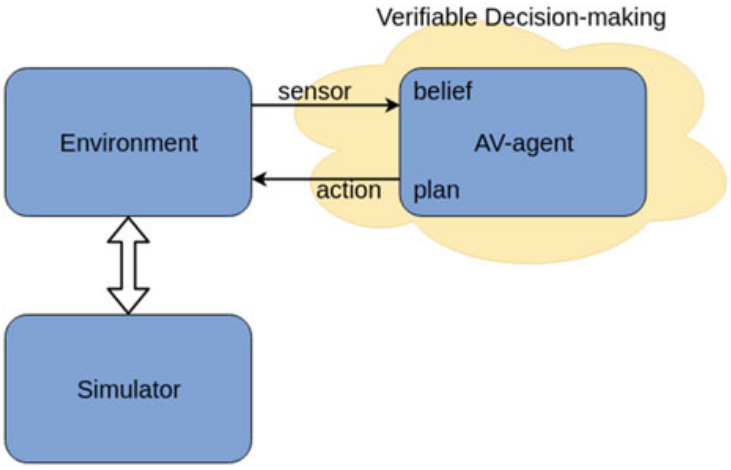

Fig. 1 The general diagram for SAE-decision-making

The SAE-decision-making comprises an environment, the AV-agent, and a (graphical) simulator. The agent captures an autonomous vehicle with basic driving decisions in the environment, where the agent works like a taxi, taking passengers from a starting point towards a destination point and checking its environment constantly for different levels of emergency. This will lead to a range of different actions being considered. Both the environment and the AV-agent can be formally defined as follows.

Definition 3.1 (Environment and Agent) An environment Σ_n is given by the following tuple, such that $n \in \mathbb{N}$:

$$\Sigma_n = (G_r, C_e, P_a, R_d, A_g)$$

where,

- G_r is a square grid[1] of $n \times n$
  - each element from the grid determines a coordinate $(X, Y) \mid (X, Y) \subset G_r$
  - X and Y set the indexes for rows and columns in G_r.
- C_e is a set of i cells, $C_e = \{c_1, c_2, \ldots, c_i\}$, such that $0 \leq i \leq n \times n$.
  - a given cell, c_i, may have one of the following status:
  - `unknown`: it means is an unknown cell, not yet discovered by the agent.
  - `clear`: it means is clear to go through it.
  - `avoidable_obstacle`: the cell has an avoidable obstacle (i.e. it represents a *yellow* level of emergency).
  - `harsh_environment`: the cell has a harsh environment (i.e. it represents an *orange* level of emergency).
  - `unavoidable_obstacle`: the cell has an unavoidable obstacle (i.e. it represents a *red* level of emergency).
- P_a is a set of j passengers, $P_a = \{p_1, p_2, \ldots, p_j\}$, such that $0 \leq j \leq n$.
- R_d is a set of k rides, $R_d = \{r_1, r_2, \ldots, r_k\}$, such that $0 \leq k \leq n$.
  - each ride is given by $r_k = (sp, dp)$
  - sp stands for *starting point*, while dp is the *destination point*
- A_g (or AV-agent) is given by the following tuple:

$$A_g = (B_e, P_l, A_c, D_i)$$

  - B_e is a set of Beliefs of A_g,

[1] To showcase the approach, we take a very simple grid model of the environment.

$$B_e = \{\, \texttt{passenger}, \texttt{ride}, \texttt{position}(X, Y), \texttt{starting\_point}(X, Y), \texttt{destination\_point}(X, Y), \texttt{moving}, \texttt{unknown}, \texttt{clear}, \texttt{avoidable\_obstacle}, \texttt{harsh\_environment}, \texttt{unavoidable\_obstacle}, \texttt{human\_controller\_ready} \,\}$$

  * `human_controller_ready`: means the agent believes the human is capable of controlling the AV and so is ready to take over.

– P_l is a set of plans for A_g,
$$P_l = \{\, \texttt{liability\_controller}, \texttt{autonomous\_control}, \texttt{resume\_manual\_control}, \texttt{finish\_all\_rides}, \texttt{complete\_journey}, \texttt{drive\_to}, \ldots \}$$
– A_c is a set of actions,
$$A_c = \{\, \texttt{drive}, \texttt{navigate}, \texttt{localise}, \texttt{search\_ride}, \texttt{refuse\_ride}, \texttt{parking}, \texttt{sound\_alarm}, \texttt{brakes}, \texttt{slow\_speed}, \ldots \}$$
– D_i is a set of possible directions that A_g may take in the G_r,
$$D_i = \{\, \texttt{North}, \texttt{South}, \texttt{East}, \texttt{West} \,\}$$

□

As it follows, we describe the intended scenario of our approach, as well as, the emergency plans written for the AV-agent. Moreover, we present our graphical simulator, which has been built to illustrate the actions taken by the AV-agent and how these actions reflect on the environment, as generally pictured on Fig. 1 (previously seen at the beginning of this section).

3.1 Scenario

In order to showcase the approach, we will consider here a basic scenario in which an AV must decide what to do in a situation where it detects an obstacle and/or a dangerous environment (e.g., an icy patch of road). In the case of an obstacle, the AV may decide it can plot a safe path around it whereby it considers the obstacle to be avoidable and so retains control.

However it may be that the vehicle is not able to calculate a safe path around the obstacle—for instance it may detect oncoming traffic in the other direction. In such a case the AV may be faced with having to solve a version of the so-called *trolley problem* [19, 3] in which a choice must be made between a number of dangerous options. Since appropriate solutions to the trolley problem remain under debate and, furthermore, in some jurisdictions (e.g., Germany) it may even be illegal to decide in advance by some algorithm what the solution to the trolley problem should be

(see [27], Chapter 6), we here take the view that the AV needs to hand control back to the driver at this point. (Note that work on the formal verification of such *ethical* dilemmas is part of current research [10, 12].)

Unfortunately, while current legislation generally requires the driver of an AV to be ready to assume control at all times, it is widely recognised that it is unrealistic to expect this to always be possible [6]. So, ethically speaking, it is important for the vehicle to have a contingency plan in case the driver does not, or can not, assume control. In this case we follow the recommendation from [7] that instead of attempting to solve the trolley problem itself the AV should instead just engage its emergency stop procedures.

3.2 Emergency Plans

The AV-agent has plans that enable the capability to drive, to undertake rides with passengers, and to successfully complete the journeys. However, we here just describe the plans specifically created to deal with different levels of emergency, which can be seen in Code 1 (which is a fragment from our AV-agent's actual programming).

Notice that if there is a *yellow* emergency level, i.e. the AV-agent believes that there is an avoidable obstacle in the environment, then the action `autonomous_control` should be taken representing the decision that the AV-agent maintains control of the vehicle.

However, if there is an *orange* emergency level, i.e. the AV-agent believes there is a particularly harsh environment and also believes the human who has the liability to control the vehicle is ready, then the autonomous control mode is released and the human regains control. But, if the agent believes the human is not ready to again take over control, actions for sounding an alarm and slowing the vehicle's speed must be taken. In this case, the AV-agent retains autonomous control until it decides the human driver is ready to (safely) regain control.

In the case of a *red* emergency level, i.e. the AV-agent believes there is an unavoidable obstacle and also believes the human who has the liability to control the vehicle is ready to take over control, then autonomous control is released. Nonetheless, if the agent again believes the human is not yet ready to take over, the actions for sounding an alarm and applying the brakes must be taken.

```
GWENDOLEN

:name: AV

:Initial Goals:

liability_controller [achieve]

:Plans:
// levels of emergency

+! liability_controller [achieve] {B unavoidable_obstacles, ~B
    human_controller_ready} <- sound_alarm, brakes;

+! liability_controller [achieve] {B unavoidable_obstacles, B
    human_controller_ready} <- +!resume_manual_control[perform];

+! liability_controller [achieve] {B harsh_environment, B human-
    controller-ready} <- +!resume_manual_control[perform];

+! liability_controller [achieve] {B harsh_environment, ~B human-
    controller-ready} <- sound_alarm, slow_speed, +!autonomous_control[
    perform];

+! liability_controller [achieve] {B avoidable_obstacle} <- +!
    autonomous_control[perform];


+!autonomous_control[perform] <- ...
// set of actions to keep the AV in autonomous mode

+!resume_manual_control[perform] <- ...
// set of actions to the human controller resume the control
```

Listing 1 GWENDOLEN code—fragment of AV-agent's plans

3.3 Agent-Environment Simulator

In order to better represent the environment in which the AV-agent proposed here is inserted, it was developed a graphic simulator to illustrate its actions. Such tool allows to observe the behaviour of an agent in a high-level perspective, by displaying every decision taken and beliefs acquired by the AV-agent in its environment.

Said agent sends messages using a pre established protocol to the simulator, which in turns interprets it translate into new modifications of the simulation. And, to build the simulation, it is used*Java 2D Graphics* library. Meanwhile, all communication is made through a client-server architecture using UDP network protocol.

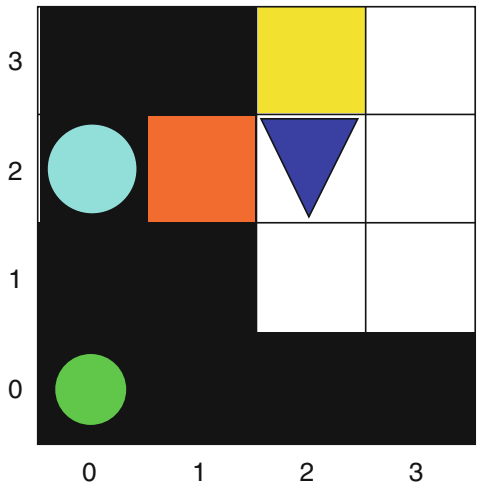

Fig. 2 A possible scenario for the SAE-decision-making simulator

Here, we create a grid to represent the environment and numbered rows and columns to identify coordinates in which the agent can move to. Each coordinate represents a cell where the AV-agent is able to move towards to. As stated before, these cells have states as it follows, `unknown`, `clear`, `avoidable_obstacle`, `harsh_environment`, `unavoidable_obstacle`, represented by, respectively: black, white, yellow, orange and red cells. In turn, the AV-agent is represented by a blue triangle. It is placed in the simulator to show in which direction the agent is moving towards. If a coordinate is the starting point (*sp*) or the destination point (*dp*) for a ride, it will contain a cyan circle or a green circle, respectively.

In Fig. 2 it is displayed a scenario in which the AV-agent located in `(2,2)` is moving south (down), trying to find a way to the starting point `(0,2)` of its next ride. Note that, the agent should be aware of the obstacles in the environment which can be found in its path. As pictured in Fig. 2 there is an avoidable obstacle at `(2,3)` and a harsh environment at `(1,2)`. If everything occur accordingly, after picking up the passenger (at the starting point), the AV-agent will head towards the destination point at `(0,0)`.

Understand that while this simulator provides a high-level representation of the proposed agent and its corresponding actions in the environment, it cannot be used to state its correctness. The AJPF model checker should be used separately from the simulator when the formal verification is conducted.

4 Formal Verifying Agent Decision-Making

In the previous, Sect. 3, we presented our basic AV-agent in GWENDOLEN. Now, using the MCAPL framework we will describe the corresponding formal verification that can be undertaken. As an example, we will concentrate on verification

concerning the plan `liability-controller`, which is indeed responsible for actions related to the reliable decision-making process from the AV-agent.

In the following, we state four properties. Each property is both given in natural language and also described using temporal logic operators, which are used in the formal specification of MCAPL framework.

1. **Red level property**:
   It always the case that the AV-agent, when it believes there is an unavoidable obstacle, but believes the human controller is *not* ready, should apply the brakes.
   - $\Box(B_{AV}$ `unavoidable_obstacle` $\wedge\neg\ B_{AV}$ `human_controller_ready`$) \Rightarrow A_{AV}$ `brakes`
2. **Red/orange level property**:
   It always the case that the AV-agent, when it believes the human controller is ready and either believes there is an unavoidable obstacle or a harsh environment, then should eventually release autonomous mode in order for the human controller to resume manual control.
   - $\Box(B_{AV}$ `human_controller_ready`$\wedge(B_{AV}$ `unavoidable_obstacle` $\vee\ B_{AV}$ `harsh_environment`$)) \Rightarrow \Diamond G_{AV}$ `resume_manual_control`
3. **Orange level property**:
   It is always the case that the AV-agent, when it believes the human controller is *not* ready to take control and believes there is a harsh environment, will slow down its speed.
   - $\Box(\neg B_{AV}$ `human_controller_ready` $\wedge\ B_{AV}$ `harsh_environment`$) \Rightarrow A_{AV}$ `slow_speed`
4. **Yellow level property**:
   It always the case the AV-agent, when believes there is an avoidable obstacle, then at some time should keep the autonomous control.
   - $\Box B_{AV}$ `avoidable_obstacle` $\Rightarrow \Diamond G_{AV}$ `autonomous_control`

Notice these properties are described using the colour code. In the red/orange level property we have a situation which is really a mixture of red and orange levels of emergency since there are two possibilities, one when there is an unavoidable obstacle, and a second one when there is a harsh environment. However, for both situations is indeed important to check whether the human driver is ready to regain control. If all the specified requirements hold, then the goal to resume manual control should be achieved. In addition, we could still have a third possibility namely that we have both an unavoidable obstacle and a harsh environment at the same time. But, when this happens the plan for the unavoidable obstacle is firstly selected since the order established in the GWENDOLEN agent programming (previously shown in Code 1).

5 Conclusion

In this work, we have presented an overview of how we can construct a high-level decision-making mechanism based on an intelligent agent approach for an autonomous system, here specifically an Autonomous Vehicle. Moreover, the decision-making process captured by such an intelligent agent can be formally verified by means of model checking techniques. Here we have used the AJPF model checker, where some properties (written in temporal logic) have been defined in order to identify whether the decision-making process of our AV-agent is indeed a reliable mechanism.

To exemplify the approach, we choose a very simple scenario whereby our AV-agent is implemented in a basic urban traffic environment with three different levels of emergency. According to the emergency recognised by the agent, different actions will be triggered.

One possible extension for the AV-agent implementation can be foreseen by the following scenario: an autonomous vehicle is used on autonomous mode if and only if the vehicle is on the motorway when the vehicle is not on the motorway, then a human driver is required. As a matter of fact, this example is mentioned in a recent Consultation paper from the Law Commission and CCAV (*Centre for Connected and Autonomous Vehicles*) from the UK [22]. This document describes that the aforementioned scenario is most likely to happen, at least in the early stages of AV deployment on the streets.

Notice we could easily adapt the AV-agent implemented in our work by means of adding new beliefs and plans in order to capture the requirements from the scenario mentioned above. That is, when the AV-agent has a belief that *it is on the motorway*, then it should engage the autonomous mode control. But, if the AV-agent believes *it is not on the motorway*, as a result the plan responsible to resume the manual control would be selected.

Furthermore, we have used colours (*yellow*, *orange*, *red*) to abstractly represent the different levels of unexpected situation, or emergency, in our urban traffic environment (deployed in the SAE-decision-making). A similar colour code could be adapted and extended for different environments which might also benefit from the use of Autonomous Systems. An example is the application of autonomous systems in nuclear energy management control [1], where an intelligent agent could be in charge of some autonomous function, but also should be aware of how, and when, to ask for human operator intervention.

In previous work [2], we have presented the first steps towards the formalisation of certain road traffic laws (aka the Rules of the Road) from the Department for Transport in the UK [15]. Generally, such Rules of the Road should be embedded into an intelligent agent in such a way that we can verify whether the agent behaves safely according to the urban traffic laws and regulations. In the future, we will consider the merging of both streams of work into a new extension of SAE system, combining the formalisation from the Rules of the Road and the decision-making

approach presented here. By establishing such a system, we should be able to formally verify whether the AV-agent will always make safe and reliable decisions.

References

1. Aitken, J., Shaukat, A., Cucco, E., Dennis, L., Veres, S., Gao, Y., Fisher, M., Kuo, J., Robinson, T., Mort, P.: Autonomous nuclear waste management. IEEE Intell. Syst. **PP**(99), 1 (2018)
2. Alves, G.V., Dennis, L., Fisher, M.: Formalisation of the Rules of the Road for embedding into an Autonomous Vehicle Agent. In: International Workshop on Verification and Validation of Autonomous Systems, pp. 1–2, Oxford, UK (2018)
3. Bonnefon, J.-F., Shariff, A., Rahwan, I.: The social dilemma of autonomous vehicles. Science **352**(6293), 1573–1576 (2016)
4. Bordini, R.H., Dennis, L.A., Farwer, B., Fisher, M.: Automated verification of multi-agent programs. In: Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, ASE '08, pp. 69–78. Washington, DC, USA. IEEE Computer Society, Piscataway (2008)
5. Bratman, M.E.: Intentions, Plans, and Practical Reason. Harvard University Press, Cambridge (1987)
6. Cunningham, M.L., Regan, M.: Driver inattention, distraction and autonomous vehicles. In: 4th International Driver Distraction and Inattention Conference (2015)
7. Davnall, R.: Solving the single-vehicle self-driving car trolley problem using risk theory and vehicle dynamics. Sci. Eng. Ethics 1–19 (2019). ISSN 1471-5546, https://doi.org/10.1007/s11948-019-00102-6
8. Dennis, L.A.: Gwendolen semantics: 2017. Technical Report ULCS-17-001, University of Liverpool, Department of Computer Science (2017)
9. Dennis, L.A., Fisher, M., Webster, M.P., Bordini, R.H.: Model checking agent programming languages. Autom. Softw. Eng. **19**(1), 5–63 (2012)
10. Dennis, L.A., Fisher, M., Winfield, A.F.T.: Towards verifiably ethical robot behaviour. In: Proc. AAAI Workshop on AI and Ethics (2015)
11. Dennis, L.A., Fisher, M., Lincoln, N.K., Lisitsa, A., Veres, S.M.: Practical verification of decision-making in agent-based autonomous systems. Autom. Softw. Eng. **23**(3), 305–359 (2016)
12. Dennis, L.A., Fisher, M., Slavkovik, M., Webster, M.P.: Formal verification of ethical choices in autonomous systems. Robot. Auton. Syst. **77**, 1–14 (2016)
13. Department for Transport—UK. Testing automated vehicle technologies in public. Available at: https://www.gov.uk/government/publications/automated-vehicle-technologies-testing-code-of-practice (2015)
14. Department for Transport—UK. Regulations for driverless cars. Available at: https://www.gov.uk/government/publications/driverless-cars-in-the-uk-a-regulatory-review (2015)
15. Department for Transport—UK. Using the road (159 to 203)—The Highway Code—Guidance. Available at: https://www.gov.uk/guidance/the-highway-code/using-the-road-159-to-203 (2017)
16. Fernandes, L.E.R., Custodio, V., Alves, G.V., Fisher, M.: A rational agent controlling an autonomous vehicle: implementation and formal verification. In: Bulwahn, L., Kamali, M., Linker, S. (eds.) Proceedings First Workshop on Formal Verification of Autonomous Vehicles. Electronic Proceedings in Theoretical Computer Science, vol. 257, pp. 35–42 (2017)
17. Fisher, M.: An Introduction to Practical Formal Methods Using Temporal Logic. Wiley, Hoboken (2011)
18. Fisher, M., Dennis, L.A., Webster, M.: Verifying autonomous systems. Commun. ACM **56**(9), 84–93 (2013)

19. Foot, P.: The problem of abortion and the doctrine of double effect. Oxf. Rev. **5**, 5–15 (1967)
20. Hawkins, A.J.: A day in the life of a Waymo self-driving taxi—The Verge. Available at: https://www.theverge.com/2018/8/21/17762326/waymo-self-driving-ride-hail-fleet-management (2018)
21. Herrmann, A., Brenner, W., Stadler, R.: Autonomous Driving: How the Driverless Revolution Will Change the World, 1st edn. Emerald Publishing, Bingley (2018). OCLC: 1031123857
22. Law Commission–Centre for Connected and Autonomous Vehicles: Automated Vehicles: a joint preliminary consultation paper. https://www.lawcom.gov.uk/project/automated-vehicles/. Accessed 8 Nov 2018
23. Lincoln, N., Veres, S.M., Dennis, L.A., Fisher, M., Lisitsa, A.: An agent based framework for adaptive control and decision making of autonomous vehicles. In: Proc. IFAC Workshop on Adaptation and Learning in Control and Signal Processing (ALCOSP) (2010)
24. Rao, A.S., Georgeff, M.P.: An abstract architecture for rational agents. In: Proc. International Conference on Knowledge Representation and Reasoning (KR&R), pp. 439–449. Morgan Kaufmann, Burlington (1992)
25. Vellinga, N.E.: From the testing to the deployment of self-driving cars: Legal challenges to policymakers on the road ahead. Comput. Law Secur. Rev. **33**(6), 847–863 (2017)
26. Visser, W., Havelund, K., Brat, G.P., Park, S., Lerda, F.: Model checking programs. Autom. Softw. Eng. **10**(2), 203–232 (2003)
27. Wicks, E.: The right to life and conflicting interests, pp. 1–288. Oxford University Press, Oxford (2010)
28. Wooldridge, M., Rao, A.: Foundations of Rational Agency, Applied Logic Series, vol. 14. Springer, Netherlands (1999)