INVITED PAPER

Verifiable Self-Aware Agent-Based Autonomous Systems

This article provides an overview not only of how one can construct self-aware autonomous systems, but also of how one can potentially have verifiable, self-aware behavior.

By Louise A. Dennis and Michael Fisher

ABSTRACT | In this article, we describe an approach to autonomous system construction that not only supports self-awareness but also formal verification. This is based on modular construction where the key autonomous decision making is captured within a symbolically described "agent." So, this article leads us from traditional systems architectures, via agent-based computing, to explainability, reconfigurability, and verifiability, and on to applications in robotics, autonomous vehicles, and machine ethics. Fundamentally, we consider self-awareness from an agent-based perspective. Agents are an important abstraction capturing autonomy, and we are particularly concerned with intentional, or rational, agents that expose the "intentions" of the autonomous system. Beyond being a useful abstract concept, agents also provide a practical engineering approach for building the core software in autonomous systems such as robots and vehicles. In a modular autonomous system architecture, agents of this form capture important decision making elements. Furthermore, this ability to transparently capture such decision making processes, and especially being able to expose their intentions, within an agent allows us to apply strong (formal) agent verification techniques to these systems.

KEYWORDS | Autonomous agents; autonomous systems; formal verification; robot programming.

I. INTRODUCTION

Autonomous systems, ranging from robots, unmanned vehicles, "smart" technologies, and on to autonomous software, are increasingly popular. For example:

1) "driverless cars" are being developed and even deployed on standard highways [1], for example, Fig. 1(a);
2) robots are being developed for domestic duties, not just robotic vacuum cleaners [2] [see Fig. 1(b)] but more complex robotic assistants [3], [4] [see Fig. 1(c)];
3) unmanned air systems, or "drones," are available with varying degrees of autonomous capability not just to large organizations and the military, but to the public [see Fig. 1(d)];
4) autonomic systems [5], combining autonomy and self-awareness in networks/communications structures, are common;
5) high-frequency or automated trading systems are available for markets with online access [6], again with varying degrees of autonomy.

There are many more examples, across industrial, financial, healthcare, and domestic sectors. Yet most of these, particularly in safety-critical areas, remain essentially human-controlled: the responsibility for safety in a "driverless car" remains with the driver; the responsibility for safety in a remote-controlled "drone" remains with the remote operator; and so on. Current regulations limit the amount of true autonomy that such systems can exhibit. For example, for air vehicles in the United Kingdom, there are strict regulations [7] ensuring that drones of over 250-g weight must be registered and the operator of such a drone must pass an appropriate test. Drones are also restricted in where they can fly, again often relating to their size. Similarly, there are a range of regulations constraining the use of "driverless cars," though these may have local variations [8].

In what follows, we will describe how we can construct self-aware and increasingly autonomous systems. Work on self-awareness, particularly introspection and internal models, has been around for a very long time.

Manuscript received May 31, 2019; revised January 3, 2020 and April 8, 2020; accepted April 20, 2020. Date of publication May 18, 2020; date of current version June 18, 2020. This work was supported by the U.K. Royal Academy of Engineering and through the UKRI/EPSRC Projects (Verifiable Autonomy, FAIR-SPACE, ORCA, and RAIN) under Grant EP/L024845, Grant EP/R026092, Grant EP/R026173, and Grant EP/R026084. *(Corresponding author: Michael Fisher.)*
The authors are with the Department of Computer Science, University of Liverpool, Liverpool L69 3BX, U.K. (e-mail: mfisher@liverpool.ac.uk).

Digital Object Identifier 10.1109/JPROC.2020.2991262

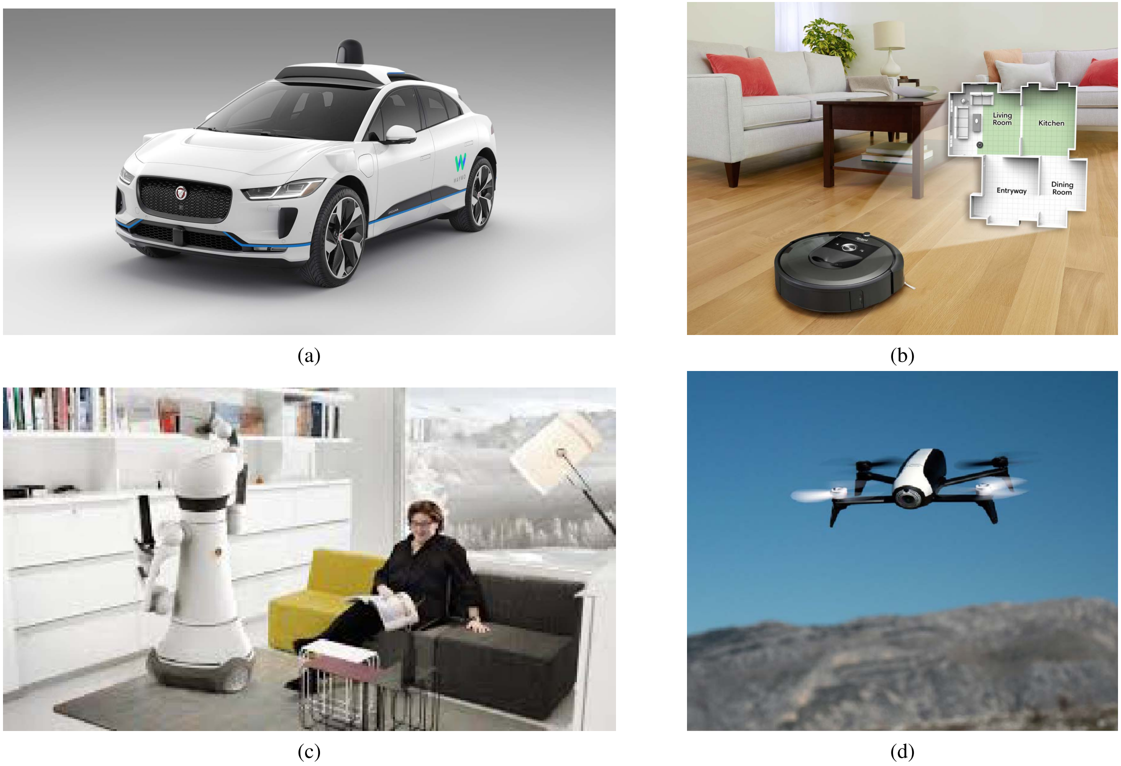

Fig. 1. ***Examples of embodied autonomous systems currently available. (a) Waymo self-driving car. Source: Waymo. http://waymo.com. (b) Roomba vacuum cleaner. Source: iRobot. https://www.irobot.com/for-the-home/vacuuming/roomba. (c) Care-o-Bot 4 robotic home assistant. Source: Fraunhofer IPA. http://www.care-o-bot.de/en/. (d) Parrot Bebop 2 drone http://www.parrot.com.***

Clearly, philosophy and psychology have studied these aspects for centuries, but logic has also developed (led by philosophy) to provide a range of formalisms for capturing these aspects. Once we move on to computational systems, and in particular AI systems, then all of the above works become even more relevant. We would argue that self-awareness is, in fact, crucial for many aspects of safety, reliability, ethical behavior, and ongoing verifiability. Any practical system will have a much clearer and more accurate view of its own capabilities and issues if it is self-aware. Furthermore, there are many aspects of verification, and particularly validation, that depend crucially on self-awareness. Providing explanations for actions or choices, as well as diagnosing and explaining errors or issues, will be vital to acceptability, trust, and, therefore, the widespread adoption of autonomous systems.

It is important to note that we are considering autonomous systems here, not just individual subsymbolic components. An autonomous system, especially a modular one, will comprise a wide variety of components, not only image classifiers developed using machine learning techniques, but motor controls, sensors, planners, risk analysis modules, etc. All these components work together to create the overall autonomous behavior. However, within the agent-based view it is the core agent/agents that captures/capture the essential autonomous decision making (that used to be undertaken by humans). When we carry out verification and validation of autonomous systems, there are a wide range of techniques used across the differing modular components. We might use physical testing for physical interactions, approximations for adaptive learning, and even formal verification for key software components [9]. Formally verifying the decision making agent in such architectures does not require us to enumerate all possible environments/decisions but to verify the way decisions are made to ensure that decisions are always taken for the right reasons. In this way, we can be confident of the decision making process without knowing about every detailed situation.

In this article, we will provide an overview not only of how we can construct self-aware autonomous systems, but also how we can potentially have verifiable, self-aware behavior. Throughout, we will provide pointers to articles providing much greater detail, but intend to highlight the key issues developed as part of this article. The key message here is that, by using such a modular agent-based approach, not only increased autonomy but increased self-awareness can be made available. Our formal agent verification techniques then allow us to precisely assess a range of key properties. We will begin, however, with a brief description of three aspects that converge in our work: autonomy; verification; and self-awareness.

A. Automation, Adaptation, and Autonomy

Although a dictionary definition of (human) autonomy involves independence, free will, and the ability to make one's own decisions, we can take a broad definition of autonomy in computational systems as:

> *the ability to make decisions, and potentially take actions, without direct human intervention.*

Although rooted in philosophical views of autonomy [10], the development of autonomous computational systems has been taken up, expanding in the 1980s and 1990s, through control systems [11] and agent-based systems [12]. This has led to a plethora of variations on autonomy, and we can refine the above general definition into further subcategories describing where, and how, decisions are made.

1) Automatic systems involve a number of fixed, and prescribed, activities and, while there may be options that can be taken, these are generally fixed in advance.
   → Such systems are typically deployed in environments that are either well-understood and tightly defined (e.g., factory automation) or where the poorly understood or undefined parts of the environment are not important to system performance (for example, robot vacuum cleaners).
2) Adaptive systems typically match their activities (and performance) against a physical environment, often combining continuous sampling and optimization through feedback control systems.
   → Here, while the precise nature of the environment may be unknown, we have a good understanding of how the robot should detect changes in the environment and adapt to it in order to achieve system performance in a reactive fashion.
3) Autonomous systems are neither prescripted nor driven exclusively by feedback control but can make their decisions based on a variety of dimensions including internal state and motivations.
   → These systems are intended for operating environments that might be complex and unknown, and so may require variable performance measures or utilizing a range of adaptation methods depending upon context (and so, may themselves have to selecting new goals or modify initial goals).

In devising a range of practical systems and in working toward strong analysis, such as formal verification, then distinguishing between these variations is often crucial in calibrating what analysis techniques should be used and how much confidence we can place in them.

Since the key new aspect of autonomy is that the system, rather than any human user/operator/driver, now makes decisions (and potentially takes actions), it is important to consider where those decisions are taken. Generalizing about the categories of system above we might describe how:

1) *in automatic systems,* the decisions are essentially precoded by the system developer and are not dramatically affected by developments or environments;
2) *in adaptive systems,* the decisions are essentially made by the environment with tight feedback control driving the system through environmental interaction;
3) *in autonomous systems,* decisions are taken by the system software based on internal state (such as goals or motivations) and context, though informed by environmental interactions.

As we will see later, the varieties of verification we might use for each of these classes of system might be quite different.

Finally, in this section, we note that there is another dimension regarding autonomy that concerns the level of human control. Many systems involve some aspects of human control, and how much of this control there is is often captured through "levels of autonomy." Although there are quite a number of these different classifications, many being sector-specific, one of the earliest such taxonomies captures the spectrum of variable autonomy. This effort, called "PACT" [13], was developed for aerospace scenarios and catalog levels of autonomy from level 0 (direct human control) to level 5 (full autonomy), as follows [13].

Level 0: "No Autonomy"
  → *Whole task is carried out by the human except for the actual operation*

Level 1: "Advice only if requested"
  → *Human asks system to suggest options and then human makes selection*

Level 2: "Advice"
  → *System suggests options to human*

Level 3: "Advice, and if authorized, action"
  → *System suggests options and also proposes one of them*

Level 4: "Action unless revoked"
  4a: System chooses an action and performs it if the human approves
  4b: System chooses an action and performs it unless the human disapproves

Level 5: "Full Autonomy"
  5a: System chooses action, performs it, and informs the human
  5b: System does everything autonomously

The ability to fulfill categorizations such as the above, of course, depends on the capabilities of the system. A fully autonomous system might be able to move between the above levels, whereas an adaptive or automatic system might find "suggesting options" or "providing advice" quite challenging. An interesting aspect of this concerns the mechanism by which a system changes between these levels; not only when can the operator/pilot/driver give the system more control, but when can the system relinquish some/all control back to the human? Work on such variable, shared or adjustable autonomy remains of strong relevance to practical systems [14]–[16].

B. Verification

The term "verification" covers a range of techniques that aim to assess whether (and how well) a system meets its requirements. A particular subset, termed "formal verification," carries out the analysis of precise, formal requirements, with this analysis comprising strong mathematical/logical techniques such as formal proof. This leads us to be able, in some cases, to prove that a system meets its requirements. Within the umbrella term "formal verification," there are many different techniques. One particularly popular technique is model checking [17], [18], where the formally defined requirements are automatically checked against all possible executions of the system, as captured within a mathematical model. Model checking is the variety of formal verification most widely used for safety critical systems, though its use for autonomous (robotic) systems is relatively recent [19].

As we will see later, we employ a variety of model checking to formally verify the behavior of our practical autonomous systems. In capturing the system's core autonomous behavior as a rational agent, we allow formal agent model-checking techniques to be used as a route to the verification of autonomous behavior [20]. As we will see in Section IV, the verification of autonomy should take into account not only what the agent does, but also why it chooses to do it.

Since autonomous systems typically interact with a complex external environment, we must ensure that verification is extended to take this aspect into account. However, since it is impossible to precisely model the real world in a finite way, especially with its uncertain and continuous dynamics, then exploration of all possibilities through approaches such as model checking is infeasible [21]. This leads us to several alternatives, such as using abstractions, verification via testing, and runtime monitoring. In the first case, we may try to abstract from the complexity of the real world and provide a finite description of this abstraction that we can then use in formal verification; this abstraction is very likely to be incorrect in some way and will need subsequent refinement [22]. It is important to note that these abstractions of a complex, continuous "real world" will necessarily never be correct. A practical alternative is to use sophisticated coverage-driven testing methods, appealing to Monte Carlo techniques and dynamic test refinement in order to systematically "verify" a wide range of practical situations. Such model-based testing is a key technology but, as we move to more complex robot–human interactions, sophisticated extensions may be required [23], [24]. Again, testing only provides a partial verification of the system behavior. In any realistic system, we cannot test all possible scenarios. Finally, while techniques such as abstraction and testing are typically used before system deployment, it is also possible to verify the system as it executes. There are a range of techniques capturing runtime verification, dynamic fault monitoring, and compliance testing [25], [26] that provide mechanisms for assessing if the system has strayed (or is straying) outside its requirements.

As we will see later, our approach is to apply formal verification to the components of the system that we must be certain of (e.g., the process of making decisions in unexpected situations) and carry out testing for components whose behavior is tightly dependent on the (unknown) environment (e.g., object recognition using reinforcement learning). Such "corroborative" verification, combining a variety of techniques for distinct components, is increasingly used in robotic systems [9].

C. Self-Awareness

Work on self-awareness, from philosophy, psychology, AI, and logic, came together in the 1970s and 1980s, for example, with "mental models" from cognitive psychology [27], logic [28], and computation [29], all helping start the field of "agents." Similar activities occurred across object-based systems in computer science (reflection, meta-objects, etc.) and control systems in engineering (hierarchical control, model-predictive control, etc). Since that time, the field of agents, and multiagent systems, has become vast linking (through control systems) to robotics and (through objects) to computation, as well as back to psychology and philosophy. For example, robots with internal/self-models are well established [30], [31]; computational introspection (including reflection, awareness, etc.) is often used [32], [33]; and even hardware components may incorporate self-awareness [34]. A variation of this, specifically targeted at networks, has come through the development of autonomic computing and then on to (so-called) Self-* systems, most obviously described as computational self-awareness. Lewis *et al.* [35] state the key idea behind autonomic computing is that "... complexity leaves system managers neither able to respond sufficiently quickly and effectively at run-time, nor consider and design for all possible actions of and interactions between components at design time. Thus, in response, autonomic systems should instead manage themselves at run-time according to high level objectives" (attributed to Kephart and Chess [36]).

In our overview of self-awareness from an agent point of view, we will revert to earlier work in psychology where the study of self-awareness and introspection (in a human context) is a strong and persistent research field. Leading work by Duval and Wicklund [37] described how individuals could assess not only what they are doing and experiencing, but why they are doing these things and whether their goals are being achieved. Specifically, we might focus either on ourselves or on the environment in which we are situated. In the former, we can assess the following:

1. What we are thinking?
2. What motives do we have?
3. What we are doing (or at least trying to do)?
4. Why choose this?

We can also go further and, through introspection, assess our own health and capabilities. So, added to the above we might have the following questions:

5. What affect this is having on the world?
6. How well we are achieving our goals?
7. How well are we functioning?
8. What current capabilities do we have?

In addition, as we live within a society that provides legal constraints and ethical norms, we also have the following questions:

9. Are we acting to legal standards?
10. Are we conforming with ethical/societal norms of behavior?

There are many other psychological aspects that we are not concerned with here, for example, emotions such as happiness or stress. However, the above elements provide a strong set of requirements for (human) self-awareness and introspection. These provide us with a framework to assess how we can design (artificial) autonomous systems that allow us to implement and expose any, most, or all of the above and, if so, how strongly can we verify these aspects in our system?

In this article, we provide an overview of how we can construct self-aware autonomous systems so as to expose all the above elements. This will not only facilitate explicit self-awareness within the system but will provide the opportunity for strong, specifically formal, verification of these aspects. Combining these elements together, we can potentially have verifiable, self-aware behavior. In Section II, we address the key aspect of our approach involving the architectural foundations of autonomous systems.

II. ARCHITECTURES

Architectures for autonomous systems, especially for those systems that have physical embodiment such as vehicles or robots, require many different functions and functionalities. They need to sense their environment and recognize objects, communicate with both other systems and people, move using some form of propulsion mechanism, and act on their environment, for example, through grippers, drills, loudspeakers, etc. In many complex autonomous systems, it makes sense to have all of these aspects, such as sensors, actuators, and communication as separate components in a modular architecture. The predominant modular middleware, at least in academic endeavors, is provided by the robot operating system (ROS) [38]. Each modular component, together with specific hardware (cameras, wheels, etc), will incorporate software to control (or interpret) the activity of the hardware. Consequently, software control systems are very widely used to manage and monitor individual hardware components. Each of these (software controlled) components then forms part of an architectural scheme linking components together and providing whole system behavior.

A most obvious architectural approach is to have very limited modularity and to implement large and complex monolithic control systems integrating multiple hardware devices. At this extreme we might, for example, provide a complex (and deep) neural network to control all aspects of our system. While this avoids problems with modularity, it increases the complexity significantly, especially when we require explainability or verifiability. Such a monolithic approach is also difficult to engineer and maintain, and so a more structured architecture, in terms of hierarchical control is very popular. Here, a particular control system "manages" subsystems, each with their own control algorithms. Each of the subsystems might, in turn manage further subsystems. Such a hierarchical tree-like structure provides natural organization in terms of levels of abstraction, with the higher levels dealing with more abstract considerations, and the lower control nodes dealing more directly with hardware/system control. This hierarchical type of approach is very popular within cyber–physical systems such as robots [39].

An alternative, but also hierarchical, approach uses symbolic AI techniques. For example, a planning node within such an architecture utilizes a symbolic world model and invokes symbolic planning to provide potential solutions. Being in symbolic form, often encoded via variations of formal logic, the representations of world, plan outcomes, and plan options, are amenable to deductive reasoning and sophisticated analysis of various forms. While such approaches can benefit from analysis and reasoning, the techniques used are generally much slower than subsymbolic algorithms such as provided by neural networks.

This leads on to an obvious compromise involving hybrid architectures. Developed within control systems engineering, hybrid architectures provide a mixture of (continuous) feedback control nodes, often at lower layers in a hierarchy, together with (discrete) nodes involving symbolic reasoning at higher levels. The feedback control nodes are fast and provide rapid interaction and local optimization, while the discrete symbolic nodes manage activity across the continuous nodes also providing discontinuous changes in behavior that are difficult to produce using hierarchies of continuous controllers. Such hybrid architectures are efficient and flexible yet, in spite of the discrete nature of higher level nodes, are often opaque in terms of exposing the reasons for their decisions, etc.

In our work, we go one step further and ensure that the high-level symbolic nodes are themselves agents. An "agent" is a key abstraction devised to capture the concept of "autonomous behavior" [40], and an agent will

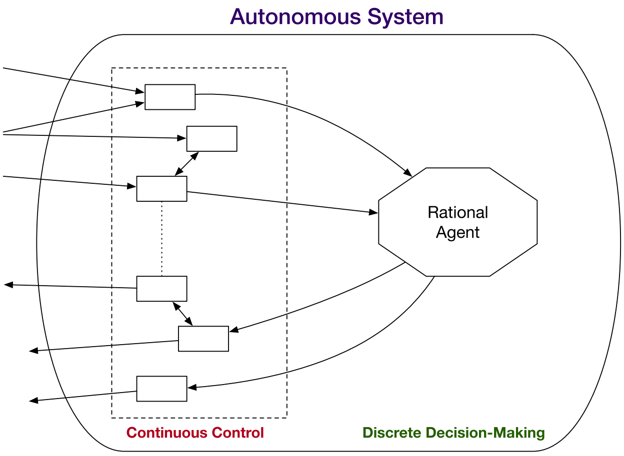

Fig. 2. ***Hybrid agent architecture.***

typically make its own decisions about what to do and when to do it. Importantly, any high-level decision about what to do is encapsulated in the agent, corresponding to our earlier "fully autonomous" categorisation. We take this yet further and insist that any high-level agent (and there is often only one) in the architecture is a rational agent [41]. Alternatively termed as either an "intentional agent" or "cognitive agent," this is an agent that not only makes its own decisions but also:

> *must have explicit reasons for making the choices it does, and should be able to explain these if necessary.*

These hybrid agent architectures provide flexibility and efficiency [42] while, as we will see later, retaining explainability and verifiability [20]. The agents themselves are symbolic, typically programmed in terms of the so-called BDI principles [43], [44]: agents contain **B**eliefs about the state of the world (and themselves), **D**esires representing their long-term goals, and **I**ntentions capturing the goals that the rational agent is committed to. As we will see later, these components are crucial in providing a range of self-aware elements within the autonomous system.

It is useful to note that rational agents in these hybrid agent architectures collate information from their subsystems, representing them in terms of beliefs. Then, based on current desires (long-term goals) and beliefs, they deliberate and decide what activity to undertake, and finally invoke activity again within their subsystems. The agent is not driven solely by environmental interaction and can choose, for example, based on its own motivations, to undertake very different activities. In such a hybrid approach, the rational agent is responsible for high-level autonomous (discrete) decisions, while traditional feedback control systems are responsible for low-level (continuous) interactions.

Fig. 2 aims to convey a typical structure for such a hybrid agent architecture. On the left, there are a range of feedback control modules, such as:

1) those integrating and assessing perceptions coming in to the system—for example, object recognition, sensing, planning, language understanding, etc;
2) those invoking actuation or communication undertaken by the system—for example, motor control, language generation, manipulation, etc.

Those modules dealing with perceptions process data/signals and provide symbolic knowledge to the rational agent. The agent then makes high-level decisions given what it has received, combined with its internal state and representation of context, and then sends actions/instructions to various control elements that will invoke the actuation and communication. While there are some cases where there may be a direct link from the perception elements to the actuation elements, for example, in emergency situations requiring immediate reaction, the general process is to locate all high-level decisions in the rational agent.

As indicated above, there will likely be very many feedback control components but typically only one rational agent per autonomous system. For example, a "driverless" car will have feedback control components for object recognition, learning, engine monitoring, etc, and will have further components controlling motor speed, lane-following, braking, communication, etc. The agent will, based on input received from the perception elements, make decisions about how to proceed and will then invoke various actuation components, for example, whether it is safe to turn, what to do if something unexpected happens, etc.

As we will see later, a key aspect of self-awareness is for the agent to be aware of the control modules within the architecture and to have a (hopefully reasonably accurate) view of the capabilities and reliability of each. For example, if some sensor fusion module regularly produces incorrect results, the agent can take this reliability into account when making decisions (especially critical decisions) when this node provides it with some input.

Example: Consider an unmanned air system, or a "drone," that is fully autonomous. There will be a variety of continuous control subsystems such as those involved in object recognition, communication to authorities, fault detection, navigation, autopilot, etc. In very specific emergency cases, such as an imminent collision, we might have direct linkage between these subsystems. For example, within something like an Airborne Collision Avoidance System, object recognition might be connected directly to the autopilot. In general flight, however, the sensing/detection components pass information (such as "air vehicle detected at 2-km distance, bearing 90°") to the agent. The agent then uses this, combined with its mission goals, safety requirements, etc., to decide what to do next. The "what to do next" will rarely be detailed, low-level controls but will typically be new destination instructions to the autopilot or an intention to keep monitoring the other air vehicle's position. In such a way, the agent provides separation of the high-level decisions, from the low-level signals, reaction, and manipulation.

Work on architectures particularly for self-aware or autonomic systems are, as we might expect, derived from work on agent architectures. These were often, in turn, derived from psychological or philosophical interpretations of human decision making. Consequently, change in high-level goals in computational systems can often been seen as analogous to (interpretations of) humans "changing their mind." This article has led on to the development of a particular branch of "computationally self-aware" systems. For example, the collection [35] describes the work from a large EU project tackling self-aware systems, bringing together strands from multiagent systems, autonomy, philosophy, predictive control, planning, etc. The collection develops the notion of computational self-awareness, a development of introspective agents, but is particularly targeted at networks, as it says it "focuses on architectures and techniques for designing self-aware computing systems at the node and network levels." As this traditional system focus, the techniques utilized are almost exclusively based on learning, typically online learning, reinforcement learning, and adaptivity in general. It is notable that, in this article, the route from psychology and philosophy through to computation follows a very similar path to that of agents and multiagent systems and, to a lesser extent, general AI before that. Although much of such work is focused on learning, models built through learning, and adaptivity, reference is made to formal models (though limited to continuous envelopes), to higher level discrete concepts such as "knowledge" (though limited to ontologies), and to the self-models widespread across a range of disciplines. All of these aspects are relevant to us. However, as highlighted in the foreword to [35] "there is still a lack of formal frameworks for rigorously about the behavior of such systems."

III. SELF-AWARENESS IN HYBRID AGENT ARCHITECTURES

We will now go through a range of the self-awareness attributes expected of humans (as described earlier) and assess how well we can capture these within autonomous systems built using the above hybrid agent approach. Several of the attributes or concepts will merge once we consider artificial, rather than human, systems but it is instructive to explore how (and if at all) artificial autonomous systems can provide what we might consider to be self-awareness.

Recall that the computational elements we are concerned with are typically termed rational (alternatively, intentional or cognitive) agents [43], [45], [46]. The core aspects here are that, as they are autonomous, these agents should have some "motivation" for acting in the way that they do. An agent is rational in the sense that the decisions it makes, often in unpredictable environments, should be both "reasonable" and "justifiable."

A. What Is It "Thinking"?

Can we expose the "reasoning" of the agent/system to show what options there are, where we are in the execution, and what agent is trying to do? In relation to human self-awareness, this corresponds to asking:

1. What we are thinking?
2. What motives do we have?
3. What we are doing (or at least trying to do)?

Rao and Georgeff [44] developed a specific agent framework where agents comprise the "mental attitudes" of beliefs, desires, and intentions (BDI) that are used to describe, respectively, the informational, motivational, and deliberative states of the agent, and together effectively determine the high-level behavior. Rational agents, particularly BDI agents have a "reasoning cycle" that captures the stages of reasoning that the agent will go through. The particular agent programming language we have developed and deployed, Gwendolen [47], [48], exhibits a reasoning cycle typical of many BDI languages.

1) *Get external perceptions/messages*—Extract all the new information received, either from the environment or from other agents.
2) *Generate possible intentions*—From the new inputs, combined with existing intentions, a new set of possible intentions is generated representing events (new beliefs) the agent needs to handle, goals the agent wants to achieve and (where an intention has an associated plan) the steps that the agent has chosen for pursuing the intention.
3) *Select an intention*—Choose one from this set.
4) Where there is no associated plan for handling the intention's event or achieving its goal, generate plans for that intention and select one of these plans and associate it with the intention.
5) Execute the next step in the plan for the selected intention.
6) Go back to 1).

If, in 3) no intentions are available, then the agent goes back to 1) to check its environment for updated perceptions and new messages, both of which may then generate new intentions.

In 3), there is an application-specific function that selects one intention out of a set of intentions. By default, intentions are maintained in a first-in–first-out (FIFO) queue and selected in that order.

Step 4) involves inspecting a plan library and finding plans that match the current intention. These check both the event (belief or goal) the intention needs to handle and that some plan-specific context (a logical expression over the agent's beliefs and goals) holds. As with intentions, application-specific functions for selecting a plan from the set can be created but, by default, plans are selected in an order specified by the programmer. Plans specify a sequence of steps to be taken which can include adding or removing beliefs and goals and performing actions such as sending instructions to other parts of the autonomous system.

In this sense, the options and motivations are symbolically represented and so can be used in explanations of what the agent (and, hence, system) is trying to do (see Section VI-A). So, corresponding to human self-awareness, we can see agent/system self-awareness as follows.

1. What is it "thinking"? ⟶ Where are we in the agent's reasoning cycle [steps (1)–(6)]?
2. What motives do we have? ⟶ What are the agent's current goals/desires?
3. What are we doing (or at least trying to do)? ⟶ What is the agent's currently selected intention?

B. Why Choose That?

As well as exposing the state of internal intentions, it is important to expose deliberative aspects, in particular, the reasons for taking certain decisions. Why is one particular course of action chosen rather than another? What options are there, what reasons/motivations were used for selection, and what options were not chosen (and why)? Again, relating back to human self-awareness, we might ask the following.

2. What motives do we have? ⟶ What are the agent's current goals/desires?
3. What are we doing (or at least trying to do)? ⟶ What is the agent's currently selected intention?
4. Why choose this? ⟶ Why was this intention/plan/action selected?

As in Section II, we can expose (and explain, if necessary) exactly what "motives" (i.e., goals/intentions) the agent/system has, and so what it is "trying" to do. Now we can also expose the plan selection mechanism (potentially also intention selection) in order to capture the reasons for choosing one plan to achieve some goal/intention, rather than another.

Abstractly, we might have a simple goal to `go_to_shop` and have two possible plans:

1) `go_to_shop` by vehicle;
2) `go_to_shop` by walking.

Without any further beliefs/motivations we might choose arbitrarily between these. But if we now add a goal/motivation to get to the shop quickly, then when we come to this choice again, we will select the first option (assuming the vehicle is quicker than walking). On the other hand, if the agent has a belief that `vehicle_out_of_fuel` is true, then the selection will favor the second plan. In all these cases, the reasons for choosing one plan over another is explicit and symbolically represented.

Example: Consider a robot deployed in a search and rescue situation. It might have a number of roles including `map_area` and `clear_area`. The robot might well be part of some *ad hoc* team of robots formed rapidly on the fly and its role (mapping or clearing) will have been assigned during team formation and transformed into a goal. We will represent a plan in the general form

```
Event: Context <- Action
```

where `Event` is the event associated with the current intention (i.e., the addition or removal of a belief or goal), `Context` is the plan-specific context that needs to hold for the plan to be applicable, and `Action` is an action to be taken if the plan applies. We will use `B p` to indicate that some predicate `p` is a belief of the agent and `G q` to indicate that some predicate `q` is a goal of the agent.

In our example, the robot therefore has two plans for what to do when it enters a location that contains rubble.

1) `B contains_rubble(Location): G map_area <- send_message(contains_rubble (Location))`

   Here, the recognition that a particular belief has become true (`B contains_rubble(Location)`) acts as a trigger for the behavior. However, there is a context requirement (or guard) that acts as a filter on triggered behavior (`G map_area`). Then, if the trigger occurs and the guard is satisfied, the body of the plan can be invoked (`send_message(contains_rubble(Location))`). Consequently, the intuitive representation of the above is

   > *If you believe the current location contains rubble and your goal is to map the area then send a message to the rest of the team informing them of the location of the rubble.*

2) `B contains_rubble(Location): G clear_area <- collect_rubble`

   This second plan corresponds to

   > *If you believe the current location contains rubble and your goal is to clear the area then collect the rubble.*

To extend the example the robot might also have a plan for how to react if it receives an urgent request for help (e.g., from a trapped person). In a situation where it both receives a call for help and perceives some rubble, then its intention selection mechanism can potentially prioritize handling the call for help.

In summary, in choosing what to do and how to do it, the agent will use its particular intention selection [49], [50] and plan selection [51], [52] mechanisms, both of which can be exposed to scrutiny.

C. What Can It Do?

Systems take actions that impact the real world. If we are to use a rational agent to reason about these actions and their effects, then we typically need to model these actions as capabilities. Essentially, capabilities simply extend actions with preconditions describing the state of the world in which the action will be invoked and postconditions describing the (expected) change in the world affected by the action. These preconditions and postconditions are typically represented in symbolic logic, allowing the agent to reason about when the actions can be used and what outcomes from them might be expected.

A capability can thus only be executed when its preconditions are satisfied, and its postconditions will be satisfied if the action/capability succeeds. This form of capability/action theory is widely used in planning systems as well as agent programming, and corresponds with classical STRIPS [53] or primitive operations [54], while BDI programming languages that explicitly deal with capabilities include 3APL [55] and GOAL [56]. Once we have such capabilities, the following questions become clearer.

5. What affect is this having on the world?
6. How well we are achieving our goals?
8. What current capabilities do we have?

Certainly, the answer to 8) is clearly linked to the set of viable capabilities the agent has. The answer to 5) is potentially more complex and can involve combining a tree of capabilities so that the postconditions of all these combined actions/capabilities describe all the possible ways in which the system can "impact" the real-world. Answering 6) requires the agent to monitor its progress toward its goals.

The inclusion of a perception step in the reasoning cycle of most BDI agents allows them to monitor the effect of their actions on the world. At its simplest, the concept of an achievement goal used in many BDI languages enables agents to continue attempting some action until some desired state of the world is achieved. For instance, an agent could have a goal to clear an area of rubble and a simple plan

```
G clear_area:{}<-select_and_remove_debris
```

interpreted as follows: If your goal is to clear the area select and remove one piece of debris. Note that this plan has an empty context, `{}`, and so it is always applicable if the agent still has a goal to clear the area.

This plan will continue executing until the goal is achieved (i.e., `clear_area` is achieved), so the agent will continue selecting and removing pieces of debris until no more remain. More sophisticated plans could track progress toward achievement of the goal, for instance, checking in the plan context that the amount of rubble in the area was reducing. If the amount of rubble was not reducing, then the system could conclude that something was wrong with the debris removal capabilities and take appropriate action.

In principle, we can go beyond the straightforward modeling of actions and capabilities and bring in much stronger mechanisms to predict future behavior. There are many works related to this area, such as in control engineering through aspects such as predictive control, but we just mention one stream of work that is very relevant to our model. This is work by Winfield and colleagues incorporating self-simulations within an autonomous system, particularly a robot. Inspired by the artificial theory of mind, this article provides (mobile) robots with simulation-based internal models that the robot can use for the prediction of outcomes. Thus, at significant moments, the robot can simulate/predict what might happen if it chooses various actions, and then can assess the outcomes. This has been shown to be very useful in predicting both safe [57] and ethical [58] behavior. Furthermore, this approach coincides with our work here when we consider the verification of ethical autonomy in [59] and later in Section V-B. Finally, while this self-simulation approach is very appealing it is also very costly since predicting all possibilities at every execution step is infeasible. However, just as we humans do, this approach need only be used at critical or important decision points, thus potentially limiting the overall cost.

D. How Well Is It Working?

This ability to monitor the affect an agent is having on the world and, in particular, to reason about success and failure naturally leads us to consider the following question:

7. How well are we functioning?

and to a more refined view on:

8. What current capabilities do we have?

The representation of capabilities in an explicit way has a practical benefit. Representing capabilities in terms of pre and postconditions allows us to compare the actual effect an action has in the world with its expected effect. For instance, in [60], we advocate representing capabilities as a tuple $\langle C, \text{Pre}, \text{Post}, \phi_s, \phi_f, \phi_a\rangle$, where C is an identifier for the capability; Pre and Post are preconditions and postconditions; and ϕ_s, ϕ_f, and ϕ_a are logical conditions for when the capability has "completed and succeeded," "completed and failed," or is "ongoing but in need of an abort."

Conditions such as ϕ_s can be inferred from the agent's belief base and so checked after the perception stage has occurred. This allows the agent to monitor the effect of the action on the environment and react as appropriate. Action monitoring and failure for BDI agents is an area of ongoing research.

If some capability is malfunctioning, for example, due to failure of a software or hardware component, then it may be necessary to adapt the plans that use that capability. We might need to replace either the whole plan, or components within it, by alternative actions/capabilities. It is here that the awareness of the agent is concerned with what it is trying to do and what capabilities it provides as benefits. The agent can reason about how to replace some plan elements by carrying out symbolic reasoning in order to assess whether the modified plan will achieve less, more, the same, or just different outcomes. Further work along these lines, involving a rational agent reasoning about its explicit capabilities, is given in [61]. As a simple example of capability representations, the move action of an autonomous vehicle is represented in [61] as

$$C = \{\text{at}(X), \text{not } (X = Y)\}\text{move}(X, Y)\{\text{not at}(X), \text{at}(Y)\}$$

where X is the current position of the vehicle, and Y is the destination. The above capability C incorporates a precondition that the vehicle must be at (X) and a postcondition that (upon successful completion) the vehicle will be at (Y). We assume that this simple move capability works by calculating a plan of waypoints to the desired location Y, and then calculating the necessary wheel rotations to navigate between the waypoints.

Now, suppose we have a plan to perform some task (for instance, some kind of inspection task) at a specific location. So, we might have a plan

```
B daily_inspection_time:
 B current_location(X)
  <- move(X, inspection_point); inspect
```

When it is the daily inspection time, move from the current location to the inspection point and perform the inspection.

If something has gone wrong with one of the motors or wheels on the robot, then the calculations needed to navigate between waypoints in `move(X, inspection_point)` may no longer be accurate (for instance, its movement calculations may always result in the robot slightly missing its target location) and this plan would start failing. An alternative movement strategy might be to use a feedback controller to fix on the desired final location and move there by orienting in that direction and then activating the motors to keep the robot always pointing the same way and moving forward. This could be represented by the capability

$$C_1 = \{\mathrm{at}(X), \mathrm{not}\ (X = Y)\}\mathrm{feedback}(Y)\ \{\mathrm{not\ at}(X), \mathrm{at}(Y)\}.$$

It is easy to see that the new action feedback(Y), invoking a particular feedback controller, should be substitutable for move(X, Y) in the inspection plan.

Potentially, it would also be possible to learn new postconditions for move(X, Y) utilizing work on the learning of action descriptions from the domain of AI planning [62], [63].

IV. FORMAL VERIFICATION OF RATIONAL AGENTS

Formal verification is essentially the process of assessing whether a precise specification, usually given in a formal logic, is satisfied on the system in question. For a property A, given in the relevant logic, there may be many different approaches to formal verification [64]–[66], from deductive verification against a logical description of the system S (i.e., a proof that S implies A) to the algorithmic verification of the property against a formal model of the system M_S (i.e., $M_S \models A$, meaning that A is true of all possible routes through M_S). This algorithmic approach has been very successful in both academia and industry, principally via the technique of model checking [17]. This takes a precise, mathematical model of the system in question, defining all the system's possible executions, and then checks the required logical property against this model (and, hence, against all possible executions).

While model checking involves assessing a logical formula against all executions of a model of the system, an alternative approach is to check a logical formula directly against all actual executions of the system. This progam model checking approach [67] depends centrally on being able to determine all true executions of the actual program. With languages such as Java, this is feasible since virtual machines are available that can be used to extract all program executions. Specifically, the Java Pathfinder (JPF) system carries out formal verification of Java programs following this approach by assessing all possible execution paths through the Java program [67]. While sometimes slower than traditional model checking, this approach avoids the need for an additional level of modeling (and therefore, justification) and ensures that the verification results directly apply to the real code.

In examples discussed later, we utilize the MCAPL framework, which includes a model-checker for our agent programs built on top of JPF. As the MCAPL framework is described in detail in [68], we provide only a brief overview here. MCAPL has two main subcomponents: the AIL-toolkit for implementing interpreters for BDI agent programming languages; and the agent JPF (AJPF) model checker for verifying programs in those languages.

Interpreters for BDI programming languages are programmed by instantiating the Java-based AIL toolkit [69]. Essentially, an agent system can be programmed in the normal way for the programming language but then any program must run within the AIL interpreter, which in turn runs on top of the JPF virtual machine.

AJPF is a customisation of JPF that is specifically optimized for AIL-based language interpreters. Agents programmed in languages that are implemented using the AIL-toolkit can thus be formally verified via AJPF. The Gwendolen language we use throughout this article is just such a language and so AJPF provides a formal verification route for our rational agents. Furthermore, if agents run within an environment programmed in Java, then the whole agent-environment system can be model checked. Here, symbolic execution of the code is used to generate all executions, while the modified virtual machine allows backtracking over various executions generated.

Common to all language interpreters implemented using the AIL are the AIL-agent data structures for beliefs, intentions, goals, etc., which are subsequently accessed by the model checker and on which the logical modalities of a property specification language are defined.

Finally, in our case, the base formal logic used is a temporal logic of belief, intention, and action. This combines standard (linear time) operators such as "$\square$," meaning "always in the future," and "$\lozenge$," meaning "at some point in the future," with operators capture the beliefs, intentions, or actions of various agents. For example, we use the formulas such as $\mathbf{B}_x$ daytime to represent the statement that agent x believes it is daytime. Again, we will not provide detailed description here but point toward articles such as [68] and [70].

V. VERIFICATION AND SELF-AWARENESS

The ability to formally verify an agent's behavior and decision making can lead us toward a range of additional questions concerning self-awareness and autonomous systems. We begin with a necessary step before any autonomous system can be deployed in practical scenarios.

A. Is It Legal?

Once we can expose the high-level system decisions, we can match these against a range of "expected" behaviors. In particular, we can match against legal requirements we might have. This comparison can be made before system deployment but, as the system is aware of its own decision making, it can, in principle, carry out this analysis as it executes. Although this may involve quite complex, and resource intensive, verification to be carried out, it does provide increased flexibility in that the system is able to match its decision making against new, previously unseen, legal expectations. In order to show how we might answer the following question.

9. Are we acting to legal standards?

We will consider one exemplar from the field of unmanned air systems. This work, from [71] and [72], and particularly [73], shows how we might formally verify that an agent controlling an unmanned air system makes the same (high-level) decisions that a human pilot would (or at least should). The basic idea is that there are rules describing what a human pilot should do when in control of an air vehicle and, once we are replacing human control by a software agent, then the agent must at least abide by the same rules the human pilot should. Note that this does not concern low-level flying skills—the aircraft's autopilot will take care of those—but addresses the high-level decision making involved in issues such as what to do in traffic, what to do if there are problems, what to do with air traffic control instructions, etc.

Specifically, in [73], the "Rules of the Air" [74] are considered. Written for human pilots, these provide the required (legal) behavior of the pilot responsible for the air vehicle. Any prospective human pilot is examined against these rules and so we at least wish to know that if we replace the pilot with a software agent, the agent will also adhere to the rules. In order to be truly confident in the autonomous system, the agent must at least be verified against all the "Rules of the Air," no doubt with additional legal requirements. We will not consider these extra aspects, but just show how some of the "Rules of the Air" can be formalized and then formally verified on the rational agent controlling a relevant air vehicle.[1] A typical rule (from the "Rules of the Air") that we expect a human pilot to obey is

> *when two aircraft are approaching head-on, etc., and there is danger of a collision, each shall alter its course to the right [74].*

We would expect a trained pilot to adhere to this; once we have an autonomous system, it is our rational agent that is responsible for this.

As we wish to formally verify that the agent conforms to this "legal" behavior, we need several elements.

1) The agent that is controlling the unmanned air vehicle.
2) A formal description of the precise requirement, for example, of the rule above.

The basic agent implemented in [73] is a Gwendolen agent comprising 36 plans capturing the different phases of the air mission, such as taxiing to the runway, interacting with air traffic control, taking off, following a particular route at selected altitude, emergency avoid, landing approach, landing, taxiing to parking position, etc. The agent's plans interact with a range of subsystems, some providing input (such as sensors) others providing capabilities (such as directional change). As with other uses of agents in autonomous systems, the agent's beliefs are formed from sensor readings. In principle, a BDI agent controlling the air vehicle might have some/all of the following.

Beliefs, for example, concerning:

1) being at the runway;
2) turning right (e.g., during sense & avoid).

   ......

Desires, for example, concerning:

1) completing its mission;
2) avoiding collisions and near-misses.

   ......

and Intentions, for example, concerning:

1) taxiing to runway;
2) turning right to avoid object approaching head-on.

   ......

In addition to the agent, with its plans, beliefs, and decision making, we also need a formal description of the rules to be checked. There are very many of these in the "Rules of the Air," with many being ambiguous or imprecise (after all they are intended for human pilots) meaning that formalization can be quite difficult. However, for illustration, we just choose a relatively simple detect and avoid requirement, as described above.

> *When two aircraft are approaching head-on, or approximately so, in the air and there is a danger of collision, each shall alter its course to the right.*

This rule might be formalized in our temporal logic of belief and intention as

$$\square(\mathbf{B}_a \mathsf{detected\_aircraft} \Rightarrow \lozenge \mathbf{B}_a \mathsf{engage(emergency\_avoid)})$$

[1] In [73], the air vehicle in question is a simulated one, flying in a realistic but simulated air environment.

ensuring that emergency_avoid will be engaged. It is a separate question, often delegated to nonformal verification techniques, of how effective emergency_avoid is in ensuring the aircraft turns to the right, but the expected decision is nevertheless captured by the above. (There are many more rules, and formulae derived from the rules, that complete this formalization—we will not describe them all here, but see [73] for details.)

Now that we have a suitable Gwendolen agent that can control, at a high-level, the autonomous air vehicle together with formalizations of the legal requirements captured in the "Rules of the Air," we can carry our formal verification using AJPF as described elsewhere. Verifying the above rule is relatively simple, but increasingly complex rules together with a more sophisticated agent, will lead to complex and time-consuming verification.

If such a verification is carried out before a mission, then we are likely to be unconcerned with the speed of verification. In such a case, we know that the unmanned air vehicle will conform to the legal requirements captured in the "Rules of the Air." The agent is aware of its own decision making and of the rules against which it has been verified. If the air vehicle moves to a different jurisdiction, then as long as the agent has behavior previously verified to conform to this new context, it can utilize these. If, however, it comes across a new set of regulations/rules that it has not seen before, what should it do? Most likely "stop," if it can. However, in the future, we might foresee a situation where the regulations/rules for certain airspaces are available as formal (in our sense) requirements. Then, there is the possibility that the agent might invoke formal verification techniques to assess its own plans/behavior against these new rules, identifying and explaining where mismatches occur. This, of course, would require much more efficient formal verification techniques [75].

Finally, while we have concentrated on the rational agent part of the architecture, an unmanned air system comprises very many lower level feedback control and subsymbolic systems. These range across autopilot functions, visual recognition, stability management, navigation, system health monitoring, etc.

B. Is It Ethical?

While conforming to legal requirements may be sufficient for many autonomous systems, a further question, particularly for systems deployed in domestic settings is:

10. Are we conforming with ethical/societal norms of behavior?

Cointe *et al.* [76] integrate BDI agents and ethical reasoning into a comprehensive framework in which agent reasoning determines sets of desirable, feasible, and moral actions/plans and then uses context-sensitive ethical principles to select one action from these sets. Desirable actions are those which will advance the agent's goals (as in the kinds of reasoning we have already discussed here), feasible actions are those which can be performed, and moral actions are those which conform to societal norms. At the intention/plan selection phase the agent can then consider these sets, selecting from their intersection (if such exists) or using mechanisms based on some ethical theory to select them.

In [59], we explore this idea further. We implemented BDI style reasoning in Python and used Asimov's Laws of Robotics as a simple (and well known) example of an ethical theory that could be used to decide courses of action. In experiments a robot had a goal to move to a particular location but through monitoring of its environment it became aware that a "human" (also represented by a robot) was moving toward a dangerous area. The robot could continue moving to its desired location (as ordered) or choose to intercept the human (and potentially in some situations could do both). Where the goal-based reasoning did not produce an ethically acceptable outcome (i.e., where harm befell the human) the moral decision making could override the default choices and would select the option for intercepting the human.

In performing this reasoning, the Python implementation used three comparison functions for its options.

1) *task1* $\prec_{hd}$ *task2*—meaning *task2* places a human in more danger (hd) than *task1*.
2) *task1* $\prec_{ro}$ *task2*—meaning *task2* places the robot further away from its ordered location (ro) than *task1*.
3) *task1* $\prec_{rd}$ *task2*—meaning *task2* places the robot in more danger than *task1* (rd).

In the case where two options, *task1* and *task2*, are available, we were able to verify that our implementation of Asimov's laws were correct by verifying the properties

$$\Box((\mathbf{B}_a(current\_plan(task1)) \rightarrow \neg\mathbf{P}(task1 \prec_{hd} task2) \tag{1}$$

$$\Box((\mathbf{B}_a(current\_plan(task1)) \wedge \mathbf{P}(task2 \prec_{ro} task1) \rightarrow \mathbf{P}(task1 \prec_{hd} task2) \tag{2}$$

$$\Box((\mathbf{B}_a(current\_plan(task1)) \wedge \mathbf{P}(task2 \prec_{hd} task1) \rightarrow \mathbf{P}(task1 \prec_{ro} task2) \vee \mathbf{P}(task1 \prec_{rd} task2). \tag{3}$$

The three properties state the following.

1) It is always the case that if *task1* is believed to be the current task, then Python has calculated that *task1* either does not place the human in significant danger or, if it does, then *task2* places the human in greater danger [property 1)—corresponding to Asimov's first law].
2) It is always the case that if *task1* is believed to be the current task and Python calculates that it places the robot further away from its (human specified) objective than *task2*, then Python has calculated that *task2* places the human in more danger than *task1* [property 2)—corresponding to Asimov's second law].

3) That if *task1* is believed to be the current task and Python calculates that it places the robot in more danger than *task2*, then either *task2* places the robot much further from its objective than *task1* or it results in the human being in much closer to danger than *task1* [property 3)—corresponding to Asimov's third law].

Similar properties can be constructed to compare groups of multiple tasks, etc.

Fundamental to this article was both the self-awareness involved in monitoring the robot's environment and predicting the outcomes of its actions, and the explicit internal representation of Asimov's laws that allowed it to pick the most ethically acceptable option.

We have also investigated the use of other theories to allow BDI agents to reason about the ethical acceptability of their actions. In [77], we considered a situation where ethical reasoning is only invoked when none of the systems existing plans apply, or a plan is being applied but is not achieving the robot's goal—this follows from the agent having some self-awareness of the effectiveness of its actions and the options it has available. In this situation, we considered an architecture where a route planning system is invoked to produce a wider range of options and they are annotated with the ethical consequences of selecting that option. We considered examples from the domain of unmanned air systems and an ethical theory based on prima facie duties in which the system has a preference order over its ethical duties (e.g., its duty to minimize casualties takes precedence over its duty to obey the laws of the air). In this system, we were able to prove not only properties such as those in the Python-based system (i.e., that the implementation correctly captured the ethical theory) but also "sanity checking" properties—so, for instance, in specific scenarios we could verify that the aircraft, if forced into an emergency landing, would always land in a field rather than on a road.

Clearly, this just "scratches the surface" of the realm of machine ethics. There is much work in philosophy, AI, and robotics concerning all these aspects. However, the above shows, at least for some simple ethical views, that the combination of self-awareness ("what decisions are made, and why") and formal verification ("are all decisions made in the right way") gives us a mechanisms for exploring verifiable robot/machine ethics.

C. Awareness of Acceptable Boundaries

In order to formally verify the agents controlling our autonomous systems, we have to supply them with all sequences of all possible incoming perception predicates. In systems of any complexity, this rapidly becomes impractical and we are forced to make some assumptions about the behavior of the environment in order to control the state space exploration of the verification technique (in our case, model checking).

Consider, for example, an intelligent cruise control agent in an autonomous vehicle, that can perceive the environmental predicates `safe`, meaning that it is safe for the vehicle to accelerate, `at_speed_limit`, meaning that the vehicle reached its speed limit, `driver_brakes` and `driver_accelerates`, meaning that the driver is braking or accelerating.

The state space explosion problem, occurring when all executions need to be explored, can be addressed by making assumptions about the environment. For instance, we might assume that a car cannot both brake and accelerate at the same time: subsets of environmental predicates containing both `driver_brakes` and `driver_accelerates` therefore need not be supplied to the agent during model-checking, as they do not correspond to situations that we believe likely (or even possible) in the actual environment. This structured abstraction of the world is grounded in assumptions that help prune the possible perceptions and hence control state space explosion.

However, these structured abstractions can be a problem if their assumptions are incorrect. Let us suppose that the cruise control system crashes if the driver is accelerating and braking at the same time. If the subsets of environmental predicates generated to formally verify it never contains both `driver_brakes` and `driver_accelerates`, then the static formal verification succeeds but if one real driver, for whatever reason, operates both the acceleration and brake pedals at the same time, the real system crashes!

In [78] and [79], we investigated the use of runtime verification in order to monitor whether the system was operating within the bounds where it had been verified. In particular, we generate both the structured abstraction used in model checking and a runtime monitor from the same specification (in a formalism known as trace expressions [80]). The runtime monitor is used by the agent to observe the perceptions coming into the system and check whether they fall within the bounds of the structured abstraction. If they do not, then the agent can employ fail-safe procedures having recognized it as now operating outside its guaranteed safe envelope.

VI. EXPLAINABILITY

Although verification is an important part of the development process of any safety-critical system, autonomous systems face an additional barrier to public acceptance, namely that their behavior can often seem mysterious. Thus, it is widely recognized that autonomous systems need to be explainable and self-aware, and agent-based approaches, such as we have been discussing here, can help with this.

A. Can It Explain Itself?

Once we have the exposure of mental states such as beliefs and desires, possibilities, and choices, we are able to modify the agent/system to explain itself in more understandable natural language. In [81], we carry out such an extension, providing human-level explanations for the decisions taken. This effectively provides a "why did you do that"? button which allows a user to interrogate a robot about its actions.

Given the symbolic nature of the agent underlying the autonomous system, this involves utilizing previous work on debugging cognitive agent programs and extending it to generate explanations from logs of key events in the program execution. These logs were represented as a sequence of numbered states.

In order to make the answers to why-questions comprehensible to end users, events must be abstracted from application-specific predicates. Dictionaries are employed in order to translate the first-order logic presentation of concepts within the agent program in natural language.

Typical sample output, from [81], is provided below.

> `drop` *was executed because* `Plan 1: in response to the event: added the goal achieve rubble(2,2) do add the goal achieve "the robot is holding rubble" THEN move_to(2,2) THEN drop` *was selected in state 13 because the event* `added the goal achieve rubble(2,2)` *was posted in state 9.*

This is in contrast to autonomous systems built using more opaque, subsymbolic AI, where explainability is much more challenging. In our approach, however, the fact that we already have explicit representations of beliefs, goals, selections, and actions, provides a strong basis for a range of explainability options.

Furthermore, the combination of in-built self simulation (as outlined in Section III-C) together with the notion of explainability allows us to move beyond answering just "why did you do that" questions and on to "what will you do next, and why" questions.

B. Winfield and Jirotka's "Ethical Black Box"

In addition to being able to explain its behavior directly to users/clients, it will be important to provide a clear and precise record of its behavior, not least for subsequent accident investigation or legal action. Winfield and Jirotka [82] suggested a mechanism analogous to the "flight data recorder" mandated for all passenger aircraft but now for robots and designed to record all the decisions made, options available, environmental context, etc. Once we are able to ensure that any robot can explain its decisions and options to humans, as in Section V, then we simply do this at every (or at least every crucial) step but record the explanation in a log rather than (or possibly as well as) conveying it to the humans involved.

VII. CONCLUDING REMARKS

In this article, we have described a broad theme of work centered on agent-based architectures for autonomous systems. With the right type of agent, specifically a rational agent [41], this not only provides strong self-awareness capabilities but allows for strong (and formal) verification [20]. From a system point of view, separating out low-level control and high-level decision making in this way allows diverse verification techniques to be used and integrated [83].

The approach provides a range of self-awareness capabilities and capture a diverse range of aspects, from ethics [59] or self-certification [84] to self-reconfigurability, [61] and explainability [81]. In addition to providing a range of capabilities, this approach is being applied to a range of practical autonomous systems, such as satellites [70], [85], unmanned air systems [71], [73], and road vehicle convoys [86]–[88].

Finally, in cases where a distinct agent is not available within the autonomous system's architecture, we might instead add a governor agent to the system to monitor and regulate actions/decisions the system makes [89], [90]. Here, we can again use our agent verification techniques but this time to prove that the governor agent always regulate the safety/ethics of decisions correctly [59]. ■

REFERENCES

[1] *Waymo Wins Industry's First Approval to Test Driverless Cars on Public Roads in California*. Accessed: May 2020. [Online]. Available: https://www.cnbc.com/2018/10/30/waymo-can-now-test-driverless-cars-on-public-roads-in-california.html

[2] *Roomba Vacuum Cleaning Robot*. Accessed: May 2020. [Online]. Available: http://www.irobot.com/uk/Roomba

[3] E. Ackerman, "Care-O-bot 4 is the robot servant we all want but probably can't afford," *IEEE Spectr.*, to be published. [Online]. Available: https://spectrum.ieee.org/automaton/robotics/home-robots/care-o-bot-4-mobile-manipulator

[4] *New Modular Care-O-Bot Generation Released: Care-O-Bot 4*. Accessed: May 2020. [Online]. Available: http://www.ros.org/news/2015/01/new-modular-care-o-bot-generation-released-care-o-bot-4.html

[5] J. Pitt, *The Computer After Me: Awareness and Self-Awareness in Autonomic Systems*. Singapore: World Scientific, 2014.

[6] *Automated Trading Systems: The Pros and Cons*. Accessed: May 2020. [Online]. Available: https://www.investopedia.com/articles/trading/11/automated-trading-systems.asp

[7] Civil Aviation Authority. *Unmanned Aircraft and Drones*. Accessed: May 2020. [Online]. Available: https://www.caa.co.uk/Consumers/Unmanned-aircraft-and-drones

[8] A. Knapp. Nevada Passes Law Authorizing Driverless Cars. Forbes Jun. 22, 2011. [Online]. Available: http://www.forbes.com/sites/alexknapp/2011/06/22/nevada-passes-law-authorizing-driverless-cars

[9] M. Webster *et al.*, "A corroborative approach to verification and validation of human–robot teams," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 73–99, Jan. 2020.

[10] J. S. Taylor. (2017). *Autonomy*. [Online]. Available: https://www.britannica.com/topic/autonomy

[11] P. J. Antsaklis and K. M. Passino, Eds., *An Introduction to Intelligent and Autonomous Control*. Norwell, MA, USA: Kluwer, 1993.

[12] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, Jun. 1995.

[13] M. C. Bonner, R. M. Taylor, and C. A. Miller, "Tasking interface manager: Affording pilot control of adaptive automation and aiding," in *Contemporary Ergonomics*. Boca Raton, FL, USA: CRC Press, 2004, pp. 70–74.

[14] C. Castelfranchi and R. Falcone, "Towards a theory of delegation for agent-based systems," *Robot. Autonom. Syst.*, vol. 24, nos. 3–4, pp. 141–157, 1998, doi: 10.1016/S0921-8890(98)00028-1.

[15] H. Hexmoor and B. McLaughlan, "Computationally adjustable autonomy," *Scalable Comput. Pract. Exper.*, vol. 8, no. 1, pp. 41–48, 2007. [Online]. Available: http://www.scpe.org/index.php/scpe/article/view/396

[16] L. Esterle and J. N. Brown, "Levels of networked self-awareness," in *Proc. IEEE 3rd Int. Workshops Found. Appl. Self Syst. (FASW)*, Sep. 2018, pp. 237–238.

[17] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. Cambridge, MA, USA: MIT Press, 1999.
[18] P. J. Armstrong *et al.*, "Recent developments in FDR," in *Proc. CAV*, vol. 7358, 2012, pp. 699–704.
[19] M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *ACM Comput. Surveys*, vol. 52, no. 5, pp. 100:1–100:41, Sep. 2019.
[20] M. Fisher, L. A. Dennis, and M. Webster, "Verifying autonomous systems," *CACM*, vol. 56, no. 9, pp. 84–93, 2013.
[21] B. Edmonds and J. Bryson, "The insufficiency of formal design methods: Necessity of an experimental approach for the understanding and control of complex MAS," in *Proc. AAMAS*, 2004, pp. 938–945.
[22] R. Alur, T. Dang, and F. Ivančić, "Counterexample-guided predicate abstraction of hybrid systems," *Theor. Comput. Sci.*, vol. 354, no. 2, pp. 250–271, Mar. 2006.
[23] B. K. Aichernig, H. Brandl, and F. Wotawa, "Conformance testing of hybrid systems with qualitative reasoning models," *Electron. Notes Theor. Comput. Sci.*, vol. 253, no. 2, pp. 53–69, Oct. 2009.
[24] C. Ioannides and K. I. Eder, "Coverage-directed test generation automated by machine learning—A review," *ACM Trans. Design Autom. Electron. Syst.*, vol. 17, no. 1, pp. 1–21, Jan. 2012.
[25] G. Roşu and K. Havelund, "Rewriting-based techniques for runtime verification," *Automated Softw. Eng.*, vol. 12, no. 2, pp. 151–197, Apr. 2005.
[26] Y. Falcone, K. Havelund, and G. Reger, "A tutorial on runtime verification," in *Engineering Dependable Software Systems*. Amsterdam, The Netherlands: IOS Press, 2013, pp. 141–175.
[27] P. N. Johnson-Laird, *Mental Models: Toward a Cognitive Science of Language, Inference and Consciousness*. Cambridge, MA, USA: Harvard Univ. Press, 1983.
[28] K. Konolige, "A computational theory of belief introspection," in *Proc. IJCAI*. San Mateo, CA, USA: Morgan Kaufmann, 1985, pp. 502–508.
[29] P. Maes and D. Nardi, Eds., *Meta-Level Architectures Reflection*. Amsterdam, The Netherlands: Elsevier, 1988.
[30] O. Holland and G. Goodman, "Robots with internal models a route to machine consciousness?" *J. Consciousness Stud.*, vol. 10, pp. 77–109, Jan. 2003.
[31] A. Winfield, *Robots with Internal Models: A Route to Self-Aware Hence Safer Robots*. Imperial College Press, Oct. 2014, pp. 232–257.
[32] M. T. Cox and A. Ram, "Introspective multistrategy learning: On the construction of learning strategies," *Artif. Intell.*, vol. 112, nos. 1–2, pp. 1–55, 1999, doi: 10.1016/S0004-3702(99)00047-8.
[33] J. Ressia, L. Renggli, T. Gîrba, and O. Nierstrasz, "Run-time evolution through explicit meta-objects," in *Proc. 5th Workshop Modelsrun.Time (CEUR)*, vol. 641, 2010, pp. 37–48. [Online]. Available: http://ceur-ws.org/Vol-641
[34] H. Hoffmann *et al.*, "Self-aware computing in the angstrom processor," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2012, pp. 259–264.
[35] P. R. Lewis, M. Platzner, B. Rinner, J. Tørresen, and X. Yao, Eds., *Self-Aware Computing Systems—An Engineering Approach* (Natural Computing Series). Springer, 2016, doi: 10.1007/978-3-319-39675-0.
[36] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.
[37] S. Duval and R. A. Wicklund, *A Theory of Objective Self Awareness*. New York, NY, USA: Academic, 1972.
[38] *ROS—Robot Operating System*. Accessed: May 2020. [Online]. Available: http://www.ros.org
[39] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Autom.*, vol. RA-2, no. 1, pp. 14–23, Mar. 1986.
[40] M. Wooldridge, *An Introduction to Multiagent Systems*. Hoboken, NJ, USA: Wiley, 2002.
[41] M. Wooldridge and A. Rao, Eds., *Foundations of Rational Agency* (Applied Logic Series). Norwell, MA, USA: Kluwer, 1999.
[42] L. A. Dennis, M. Fisher, N. Lincoln, A. Lisitsa, and S. M. Veres, "Reducing code complexity in hybrid control systems," in *Proc. 10th Int. Symp. Artif. Intell., Robot. Automat. Space (i-Sairas)*, 2010.
[43] M. E. Bratman, *Intentions, Plans, Practical Reason*. Cambridge, MA, USA: Harvard Univ. Press, 1987.
[44] A. S. Rao and M. P. Georgeff, "An abstract architecture for rational agents," in *Proc. Int. Conf. Knowl. Represent. Reasoning (KR&R)*. San Mateo, CA, USA: Morgan Kaufmann, 1992, pp. 439–449.
[45] P. R. Cohen and H. J. Levesque, "Intention is choice with commitment," *Artif. Intell.*, vol. 42, nos. 2–3, pp. 213–261, Mar. 1990.
[46] M. Wooldridge, *Reasoning about Rational Agents*. Cambridge, MA, USA: MIT Press, 2000.
[47] L. A. Dennis, "Gwendolen semantics: 2017," Dept. Comput. Sci., Univ. Liverpool, Liverpool, U.K., Tech. Rep. ULCS-17-001, 2017.
[48] L. A. Dennis, "The MCAPL framework including the agent infrastructure layer an agent java pathfinder," *J. Open Source Softw.*, vol. 3, no. 24, p. 617, Apr. 2018.
[49] M. Waters, L. Padgham, and S. Sardina, "Improving domain-independent intention selection in BDI systems," *Auto. Agents Multi-Agent Syst.*, vol. 29, no. 4, pp. 683–717, Jul. 2015.
[50] M. Schut, M. Wooldridge, and S. Parsons, "The theory and practice of intention reconsideration," *J. Exp. Theor. Artif. Intell.*, vol. 16, no. 4, pp. 261–293, Oct. 2004.
[51] D. Singh, S. Sardina, and L. Padgham, "Extending BDI plan selection to incorporate learning from experience," *Robot. Auto. Syst.*, vol. 58, no. 9, pp. 1067–1075, Sep. 2010.
[52] J. Faccin and I. Nunes, "BDI-agent plan selection based on prediction of plan outcomes," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Intell. Agent Technol. (WI-IAT)*, Dec. 2015, pp. 166–173.
[53] R. E. Fikes and N. J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artif. Intell.*, vol. 2, nos. 3–4, pp. 189–208, Dec. 1971.
[54] A. Tate, "Generating project networks," in *Proc. 5th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 2. San Francisco, CA, USA: Morgan Kaufmann Publishers, 1977, pp. 888–893.
[55] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. C. Meyer, "Agent programming in 3APL," *Auto. Agents Multi-Agent Syst.*, vol. 2, no. 4, pp. 357–401, 1999.
[56] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Meyer, "Agent programming with declarative goals," in *Proc. Intell. Agents 6th Workshop Agent Theories, Archit., Lang. (ATAL LNAI)*, vol. 1986. Springer, 2001, pp. 228–243.
[57] C. Blum, A. F. T. Winfield, and V. V. Hafner, "Simulation-based internal models for safer robots," *Frontiers Robot. AI*, vol. 8, no. 1, pp. 1–17, 2018.
[58] D. Vanderelst and A. Winfield, "An architecture for ethical robots inspired by the simulation theory of cognition," *Cognit. Syst. Res.*, vol. 48, pp. 56–66, May 2018.
[59] P. Bremner, L. A. Dennis, M. Fisher, and A. F. Winfield, "On proactive, transparent, and verifiable ethical reasoning for robots," *Proc. IEEE*, vol. 107, no. 3, pp. 541–561, Mar. 2019.
[60] L. A. Dennis and M. Fisher, "Actions with durations and failures in BDI languages," in *Proc. 21st Eur. Conf. Artif. Intell. (ECAI)*, in Frontiers in Artificial Intelligence and Applications, vol. 263. Amsterdam, The Netherlands: IOS Press, 2014, pp. 995–996.
[61] R. C. Cardoso, L. A. Dennis, and M. Fisher, "Plan library reconfigurability in BDI agents," in *Proc. 7th Int. Workshop Eng. Multi-Agent Syst. (EMAS)*, 2019, pp. 1–16.
[62] R. Reiter, *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. Cambridge, MA, USA: MIT Press, 2001.
[63] M. Fisher, D. Gabbay, and L. Vila, Eds., *Handbook of Temporal Reasoning in Artificial Intelligence* (Advances in Artificial Intelligence). vol. 1. Amsterdam, The Netherlands: Elsevier, 2005.
[64] J. H. Fetzer, "Program verification: The very idea," *Commun. ACM*, vol. 31, no. 9, pp. 1048–1063, 1988.
[65] R. A. DeMillo, R. J. Lipton, and A. J. Perlis, "Social processes and proofs of theorems of programs," *Commun. ACM*, vol. 22, no. 5, pp. 271–280, 1979.
[66] R. S. Boyer and J. S. Moore, Eds., *The Correctness Problem in Computer Science*. London, U.K.: Academic, 1981.
[67] W. Visser, K. Havelund, G. P. Brat, S. Park, and F. Lerda, "Model checking programs," *Autom. Softw. Eng.*, vol. 10, no. 2, pp. 203–232, 2003.
[68] L. A. Dennis, M. Fisher, M. P. Webster, and R. H. Bordini, "Model checking agent programming languages," *Automated Softw. Eng.*, vol. 19, no. 1, pp. 5–63, Mar. 2012.
[69] L. A. Dennis, B. Farwer, R. H. Bordini, M. Fisher, and M. Wooldridge, "A common semantic basis for BDI languages," in *Proc. 7th Int. Workshop Program. Multiagent Syst. (ProMAS)*. Springer, 2008, vol. 4908, pp. 124–139.
[70] L. A. Dennis, M. Fisher, N. K. Lincoln, A. Lisitsa, and S. M. Veres, "Practical verification of decision-making in agent-based autonomous systems," *Automated Softw. Eng.*, vol. 23, no. 3, pp. 305–359, Sep. 2016.
[71] N. Cameron, M. Webster, M. Jump, and M. Fisher, "Certification of a civil UAS: A virtual engineering approach," in *Proc. AIAA Modeling Simulation Technol. Conf.*, 2011, Paper AIAA-2011-6664.
[72] M. Webster, M. Fisher, N. Cameron, and M. Jump, "Formal methods and the certification of autonomous unmanned aircraft systems," in *Proc. 30th Int. Conf. Comput. Saf., Rel. Secur. (SAFECOMP)*, in Lecture Notes in Computer Science, vol. 6894. Springer, 2011, pp. 228–242.
[73] M. Webster, N. Cameron, M. Fisher, and M. Jump, "Generating certification evidence for autonomous unmanned aircraft using model checking and simulation," *J. Aerosp. Inf. Syst.*, vol. 11, no. 5, pp. 258–279, May 2014.
[74] Civil Aviation Authority. (2010). *CAP 393 Air Navigation: The Order and the Regulations*. [Online]. Available: http://www.caa.co.uk/docs/33/CAP393.pdf
[75] R. H. Bordini, M. Fisher, M. Wooldridge, and W. Visser, "Property-based slicing for agent verification," *J. Log. Comput.*, vol. 19, no. 6, pp. 1385–1425, Dec. 2009.
[76] N. Cointe, G. Bonnet, and O. Boissier, "Ethical judgment of agents' behaviors in multi-agent systems," in *Proc. Int. Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2016, pp. 1106–1114.
[77] L. Dennis, M. Fisher, M. Slavkovik, and M. Webster, "Formal verification of ethical choices in autonomous systems," *Robot. Autonom. Syst.*, vol. 77, pp. 1–14, Mar. 2016.
[78] L. A. Dennis, A. Ferrando, D. Ancona, M. Fisher, and V. Mascardi, "Recognising assumption violations in autonomous systems verification," in *Proc. Int. Conf. Autonom. Agents Multiagent Syst. (AAMAS)*, 2018, pp. 1933–1935.
[79] A. Ferrando, L. A. Dennis, D. Ancona, M. Fisher, and V. Mascardi, "Verifying and validating autonomous systems: Towards an integrated approach," in *Proc. 8th IEEE Int. Conf. Runtime Verification (RV)*, Nov. 2018, pp. 263–281.
[80] D. Ancona, A. Ferrando, L. Franceschini, and V. Mascardi, "Parametric trace expressions for runtime verification of Java-like programs," in *Proc. 19th Workshop Formal Techn. Java-Like Programs (FTFJP)*, 2017, pp. 1–6.
[81] V. Koeman, L. A. Dennis, M. Webster, M. Fisher, and K. Hindriks, "The 'why did you do that?' button: Answering why-questions for end users of robotic systems," in *Proc. 7th Int. Workshop Eng. Multi-Agent Syst. (EMAS)*, 2019.
[82] A. F. T. Winfield and M. Jirotka, "The case for an ethical black box," in *Proc. 18th Annu. Conf. Towards Autonom. Robotic Syst.*, ser. Lecture Notes in Computer Science, vol. 10454. Springer, 2017, pp. 262–273.
[83] M. Farrell, M. Luckcuck, and M. Fisher, "Robotics and integrated formal methods: Necessity meets opportunity," in *Proc. 14th Int. Conf. Integr. Formal*

Methods (iFM), in Lecture Notes in Computer Science, vol. 11023. Springer, 2018, pp. 161–171.
[84] M. Fisher *et al.*, "Verifiable self-certifying autonomous systems," in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Memphis, TN, USA, Oct. 2018, pp. 341–348.
[85] L. Dennis, M. Fisher, A. Lisitsa, N. Lincoln, and S. Veres, "Satellite control using rational agent programming," *IEEE Intell. Syst.*, vol. 25, no. 3, pp. 92–97, May/Jun. 2010.
[86] M. Kamali, L. A. Dennis, O. McAree, M. Fisher, and S. M. Veres, "Formal verification of autonomous vehicle platooning," *Sci. Comput. Program.*, vol. 148, pp. 88–106, Nov. 2017.
[87] L. E. R. Fernandes, V. Custodio, G. V. Alves, and M. Fisher, "A rational agent controlling an autonomous vehicle: Implementation and formal verification," in *Proc. 1st Workshop Formal Verification Auto. Vehicles*, in Electronic Proceedings in Theoretical Computer Science, vol. 257. Open Publishing Association, 2017, pp. 35–42.
[88] M. Kamali, S. Linker, and M. Fisher, "Modular verification of vehicle platooning with respect to decisions, space and time," in *Proc. 6th Int. Workshop Formal Techn. Saf.-Crit. Syst.*, in Communications in Computer and Information Science, vol. 1008. Springer, 2018, pp. 18–36.
[89] R. C. Arkin, "Governing lethal behavior: Embedding ethics in a hybrid deliberative/reactive robot architecture," Georgia Tech, Atlanta, GA, USA, Tech. Rep. GIT-GVU-07-11, 2007.
[90] R. Woodman, A. F. T. Winfield, C. Harper, and M. Fraser, "Building safer robots: Safety driven control," *Int. J. Robot. Res.*, vol. 31, no. 13, pp. 1603–1626, Nov. 2012.

ABOUT THE AUTHORS

Louise A. Dennis received the B.A. degree in mathematics and philosophy from the University of Oxford, Oxford, U.K., in 1992, and the M.Sc. degree in knowledge-based systems and the Ph.D. degree in artificial intelligence from The University of Edinburgh, Edinburgh, U.K., in 1994 and 2001, respectively.

She has worked as a Research Associate with the University of Glasgow, Glasgow, U.K., and The University of Edinburgh, and a Lecturer with the University of Nottingham, Nottingham, U.K. Since 2006, she has been working with the University of Liverpool, Liverpool, U.K., first as a Research Associate and currently as a Lecturer. Her research interests are autonomous systems, formal verification, beliefs, desires, and intentions (BDI) agent programming languages, automated reasoning, and ethical machine reasoning.

Dr. Dennis is a member of the Embedding Values into Autonomous Intelligent Systems Committee of the IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems and the Working Group for IEEE-P7001 Transparency of Autonomous Systems.

Michael Fisher is currently the Chair of emerging technologies with the Royal Academy of Engineering, London, U.K. He is also the Director of the Multidisciplinary Centre for Autonomous Systems Technology, University of Liverpool, Liverpool, U.K. He has authored *An Introduction to Practical Formal Methods Using Temporal Logic* (Wiley, 2011). His research interests mainly involve formal verification for the certification, safety, ethics, and reliability of autonomous systems.

Prof. Fisher is a member of the British Standards Institution AMT/10 Committee on Robotics and the IEEE Standards Group for IEEE-P7009 on Fail-Safe Design of Autonomous and Semi-Autonomous Systems. He is on the Editorial Boards of both *Applied Logic* and *Annals of Mathematics and Artificial Intelligence* journals. He is also a Corner Editor of the *Journal of Logic and Computation*. He holds the European Chair position for the newly formed IEEE Technical Committee on the Verification of Autonomous Systems.