Chapter 7

Verifiable Autonomy and Responsible Robotics

Louise Dennis and Michael Fisher

Abstract The move towards greater autonomy presents challenges for software engineering. As we may be delegating greater responsibility to software systems and as these autonomous systems can make their own decisions and take their own actions, a step change in the way the systems are developed and verified is needed. This step involves moving from just considering what the system does, but also why it chooses to do it (since decision-making may be delegated). In this chapter, we provide an overview of our programme of work in this area: utilising hybrid agent architectures, exposing and verifying the reasons for decisions, and applying this to assessing a range of properties of autonomous systems.

1 Introduction

Autonomy is surely one of the key technological themes for the twenty-first century. We expect to soon see fully autonomous vehicles, robots, and software, all of which will need to be able to make their own decisions and take their actions, without direct human intervention. Depending on our levels of trust in such technologies, this potential future may be exciting or frightening, and possibly both! Exciting in that autonomous systems can allow us to devise support and tackle tasks well beyond current possibilities; frightening in that the direct control of these *autonomous* systems is now taken away from us.

How do we know that they will work? How do we know that they are safe? And how can we trust them? All of these are currently impossible to answer since we cannot be sure that such systems will never deliberately cause harm. Yet, without such guarantees, increased autonomy could be unlikely to be accepted by engineers, allowed by regulators, or trusted by the public.

L. Dennis · M. Fisher (✉)

Department of Computer Science, University of Manchester, Manchester, UK
e-mail: louise.dennis@manchester.ac.uk; michael.fisher@manchester.ac.uk
<https://www.cs.man.ac.uk/~dennisl>; <https://web.cs.manchester.ac.uk/~michael>

© The Author(s) 2021

A. Cavalcanti et al. (eds.), *Software Engineering for Robotics*,
https://doi.org/10.1007/978-3-030-66494-7_7

189

As we can never precisely describe, or model, the real world, we are never able to say that a system will be absolutely safe or will definitely achieve something; instead, we can only say that it *tries* to be safe and tries to carry out its tasks to the best of its ability. This distinction is crucial: we can only prove that an autonomous system never decides to do the wrong thing and always aims to act safely, we cannot guarantee that accidents will never happen.

However this step, in itself, is important—if an accident occurs, did the system intentionally cause the accident, or did it just make a mistake? Examining the decision-making process, and particularly the *reasons* for such decisions, becomes crucial.

Traditionally, practical autonomous systems comprise complex and intricate controllers, often involving open and closed-loop control systems of many varieties (parametric, neural, fuzzy, etc.). However, since the reasons for any choice of action are often opaque, such systems are hard to develop, control, and analyse.

We utilise a *hybrid agent architecture*, wherein we can separate the discontinuous changes (i.e. ‘deliberate’ choices) into a distinct ‘agent’, leaving the low-level control aspects to traditional methods. These low-level elements range across adaptive control, image classification, sensor fusion techniques, manipulation, etc. A further aspect is that this agent must have, and be able to explain, explicit reasons for making the choices it does [21, 31].

Once we have such a hybrid agent architecture, the reasons for particular choices are exposed and transparent. This is vital to regulators who need to be able to interrogate the exact motivations for decisions, for engineers who need clear and precise structure in systems decisions, and the public who may be convinced of a robot’s trustworthiness if they can access the internal reasoning processes. Using this approach we can also directly tackle complex questions of safety, ethics, legality, etc. [10, 54]. Without this fundamental step, no amount of testing or simulation is likely to satisfy system producers, the public, or government regulators. This will involve formally verifying the agent’s (discrete, high-level, finite) decision-making and that decisions are taken for the right reasons.

In this chapter, we provide an overview of our programme of work in this area: utilising hybrid agent architectures, exposing and verifying the reasons for decisions, and applying this to assessing a range of properties of autonomous systems. Our long-term aim is to ensure that practical and useful (effective) autonomous systems can be constructed that can truthfully (verifiably) explain their intentions. Since truly autonomous robots will have at least some, and possibly quite significant, responsibilities, this will be crucial. This chapter has close links with, and relevance to, Winfield’s work (Chap. 6). Indeed, we have collaborated closely in developing formal verification for his ethical governors; see Sect. 8.2. Furthermore, the ability of our systems to explain, reliably and before acting, what they will do and why they make these choices solves one of the core issues in explainability, as highlighted by Winfield and Jirotko’s work on an *ethical black box* [56]; see Sect. 7.1.

In what follows, we will begin by exposing the key issues with autonomous systems. We follow this with a description of our approach and then explanations of how this solves many of the issues highlighted earlier.

2 What Is the Problem?

2.1 *Concerns About Autonomy*

Once the decision-making process is taken away from humans, even partially, can we be sure what autonomous systems will do? Will they be safe? Can we ever trust them? What if they fail? This is especially important once robotic devices, autonomous vehicles, etc., increasingly begin to be deployed in safety-critical situations.

Example from Popular Culture We have seen in science fiction films such as *Terminator* and *I, Robot* typical examples of robots with sinister intent. In such films, there is an assumption that, just as with humans, we cannot accurately expose the intentions that these robots have. However, if we construct our robot's software carefully, then we certainly can assess the *reasons* for making certain choices.

2.2 *No Psychiatrists for Robots?*

As we move towards increased autonomy, we need to assess not just *what* the robot will do, but *why* it chooses to do it. As above, the common view is that we cannot reason about why the robot makes the decisions it does, and specifically cannot capture its *intent*. However, with an *autonomous system* that we have built, we should, if we are careful with its construction, be able to examine its internal programming and find out exactly

1. *what* it is 'thinking',
2. what *choices* it has, and
3. why it *decides* to take particular options.

Consequently, we explicitly program autonomous systems, such as robots, using an intentional approach and ensure that these are both transparent and exposed. Returning to the *Terminator* example above, think how different (and perhaps uninteresting) such films would have been if robots had to truthfully explain their intentions. If we could guarantee, through proof for example, that decisions were always made for the right reasons and that these reasons were truthfully displayed, then think how quickly you would become confident, and possibly more trusting, of such robots.

3 Background

Although robots have been in use for many years, these have been typically part of automated systems (such as production lines) that have very fixed tasks to undertake. Furthermore, these robotic systems are regularly monitored by human operators. But, this situation will change. Increasingly, there is a strong need for robotic systems that can not only continue working for long periods of time but can do so without close human supervision. It is only when these systems become autonomous that we will see significant change and efficiency improvements.

However, current robotic systems, such as driver less cars, unmanned air vehicles, domestic robots, manufacturing robots, etc., are far from being autonomous. They remain closely controlled by human operators, drivers, or pilots in most crucial situations.

Why is this? Essentially, there is a step change that is needed to move from analysis of standard cyber-physical systems, typically controlled by humans, to truly autonomous systems that make their decisions and have their own responsibilities. This often presents a barrier to the development of technology.

We would like to have autonomous home-care robots that can significantly improve social care for the elderly, driver less cars that fundamentally improve accident rates on our roads, nuclear robots that can reliably decommission a complex power-station without danger to humans, etc. All of these types of systems can strongly impact, and enhance, both societies and economies.

The main barriers to the development of such systems are not technological, but ethical, legal, and social. Again understanding why decisions are made in autonomous systems is central.

3.1 *Autonomy*

We begin with a straightforward definition of *autonomy*:

- the ability of a system to make its own decisions and to act on its own, and to do both without direct human intervention.

With this definition, we are particularly focussing on systems where *software* has a key role in decision-making.

Who Makes the Decisions? Even within the above definition of ‘autonomy’, there are important variations concerning how this autonomy (or autonomous decision-making) is achieved.

- **Automatic:** involves several fixed, and prescribed, activities; there may be options, but these are generally fixed in advance and rely on fairly minimal input from the environment—for example, to check the positioning of a target is within acceptable bounds.

- **Adaptive:** improves its performance/activity based on feedback from its environment—typically developed using tight continuous control and optimisation (e.g. a feedback control system) and generally modelled using differential equations.
- **Autonomous:** decisions made based on the system’s (belief about its) current situation at the time of the decision—the environment may still be taken into account, but the motivations and beliefs within the system are important.

Especially when we come to (formal) verification, distinguishing *between* these variations is often crucial. If we think about who or what makes the decisions, then we have very different forms of, and targets for, verification:

- in *automatic systems*, the decisions are encoded by the system developer and so the exact options can be verified before deployment;
- in *adaptive systems*, the decisions are essentially made by the environment with tight feedback control driving the system through environmental interaction, to which the system merely reacts, and so verification here requires a model (always imprecise) of the environment; and
- in *autonomous systems*, decisions are taken by the software based on internal goals and motivations while taking into account beliefs about the environment, and so here verification should be applied particularly to the core decision-making software (agent).

Our system architectures will combine these elements and so analysis will require verification of all these different forms. So, we next revisit a range of suitable verification techniques that we might use for the above components.

3.2 Verification

The aim of verification is to ensure that a system matches its requirements; formal verification takes this further, not only having precise formal requirements but carrying out a comprehensive mathematical analysis of the system to ‘prove’ whether it corresponds to these formal requirements.

Essentially, verification is the process of assessing, sometimes formally, that we have built the system satisfying its requirements/specification.

As we will see later, we capture the system's core autonomous behaviour as a *rational agent*. Once we do this, formal verification techniques for autonomous decision-making become possible. Any verification technique here needs to take into account not only what the agent does but why it chooses to do it [31]. In this case, a logical proof about the agent's choices is required. Note that, in legacy cases, where a distinguished decision-making agent is *not* a core part of the architecture, we might be able to formally verify any governor component usually used to regulate the safety/ethics of decisions within a robotic system [5, 58].

Varieties of Formal Verification Verifying that an agent, A , matches a (formal) requirement, R , for that agent can be carried out in many ways.

- *Proof*: the behaviour of the agent is itself formally described by the logical formula, φ_A , and verification involves *proving* that this behaviour, φ_A , always implies that the requirement, R . (Essentially, $\vdash \varphi_A \Rightarrow R$.)
- *Traditional Model-Checking*: where R is checked against a representation of all possible execution paths of the agent, A .
- *Synthesis*: from R generate an automaton implementing A that guarantees to only produce runs satisfying R .
- *Dynamic Fault Monitoring (aka Runtime Verification)*: where executions *actually* generated by the agent, A , as it is running are checked against R .
- *Program Model-Checking*: all execution paths the program implementing the agent, A , can take are checked against R .

We also note that *Testing*, whereby R is checked on a subset of the possible executions of the agent A , is also a popular, though not exhaustive, route. As we will see later, the above varieties of verification will be used for the different components within the system.

Formal verification, through model-checking, of agent-based autonomous systems is still at a relatively early stage [7, 8, 30, 46] though the techniques and tools reported in this chapter are at the forefront of this work, specifically the MCAPL framework and AJPF program model-checker [20, 21, 28].

3.3 Trustworthiness

The concept of trust in autonomous systems is quite complex and subjective [33]. However, the *trustworthiness* of an autonomous system usually comprises at least two aspects:

1. *Reliability*—does system work reliably and predictably?
2. *Beneficiality*—is the system working for our benefit, for example, is it trying to help us, not harm us?

Even for (non-autonomous) cyber-physical systems, providing evidence of reliability is problematic. This often involves mathematical models of the system and

its environment, both of which will necessarily be wrong, in the sense that they do not capture the physical world. In the case of autonomous systems, we are particularly unclear about exactly ‘how wrong’ these models are, especially once we deploy them in unknown and dynamic environments. To assess the reliability of a system, we will also need to be clear about exactly what we expect it to do (and so what its requirements are). While this captures the trustworthiness of a standard cyber-physical system, once we move to autonomous systems, then the concept of *beneficiality* becomes important.

Since we are delegating activity and responsibility to an autonomous system, and especially where this might involve some critical activities, we must be sure that the system will behave as we would want even when not directly under our control. It is here that ‘why’ the system will choose to make certain decisions becomes crucial. In complex environments we cannot predict all the decisions that must be made (and so cannot pre-code all the ‘correct’ decisions), but we can ensure that, in making its decisions, an autonomous system will carry them out ‘in the right way’.

Example A domestic robot helper might have a standard mode that prohibits contact with humans. Thus this robot in our home keeps out of our way. In the (uncommon) case where the single home occupant is unconscious and a fire has started in the house, then the robot should carry out reasoning (and assess the risks) hopefully deciding to (gently) carry the home occupant out of the house.

Clearly, for this example we could hard-wire a policy but there will always be scenarios that we have not considered (and so set rules for). In making its decisions, the robot’s primary goal is to keep the occupant safe.

Providing evidence for the beneficial activities of the system is a challenge, particularly for systems built from opaque components, such as deep-learning modules. However, as we will see later in this chapter, if we build the system appropriately, exposing the reasons for decisions, and then apply formal verification techniques, we can have much greater confidence in the system, especially its beneficiality.

4 Autonomous Robotics

Robots are already here: machines that clean our houses are well established [49] (see also Fig. 7.1), and factory automation has been available for a while with robots making our cars, our clothes, even our food [35]! But now a new generation of robots is being designed that have the potential to affect safety. Unlike factory robots, they have no cages around them, and unlike robot floor cleaners, they are multi-functional with a range of sophisticated capabilities.

These robots (e.g. Care-O-bot, Fig. 7.2) are increasingly *autonomous*—they are designed to make their own decisions, and even act, *without* human intervention. These decisions may be essential in critical situations: maybe the robot is assisting a person at home who has an accident, or is assisting a person at work who does

Fig. 7.1 Roomba vacuum cleaner. Source: iRobot: <https://www.irobot.com/for-the-home/vacuuming/roomba>.

Fig. 7.2 Care-O-bot 4 robotic home assistant. Source: Fraunhofer IPA: <http://www.care-o-bot.de/en/>.

not see imminent danger. These are not an isolated class, since unmanned vehicles (see Figs. 7.3 and 7.4) and broad AI decision systems might all be required to take critical decisions.

Fig. 7.3 Parrot Bebop 2 drone. Source: <http://www.parrot.com>.

Fig. 7.4 Waymo self-driving car. Source: Waymo: <http://waymo.com>.

As such systems become increasingly autonomous, so the need to capture the core autonomous decision-making (where possible) becomes crucial. This enhances explainability, but, if we can then apply formal verification techniques, then we can impact upon trustworthiness of these new classes of robotic system.

Fig. 7.5 Modular architecture. Copyright Dennis and Fisher, printed with permission in this chapter.

4.1 Architectures

In constructing autonomous systems such as autonomous robots, software engineering is a key aspect since the ‘autonomous behaviour’ is primarily encapsulated within software. There is a range of (software) architectural styles used for these systems, but with several underlying themes as described below.

4.1.1 Robot Architectures: Modularity

First, *modularity* has now been widely adopted, both because of developing international standards (ISO/DIS 22166)¹ and through the widespread use of de facto standards such as the *Robot Operating System* (ROS) [45]. Such modularity separates key architectural elements and permits improved design, analysis, and maintenance. See Fig. 7.5 for a representation of a typical modular architecture.

In addition to permitting and supporting *inter-operability* amongst sub-components, such modularity provides two specific benefits for us. First, it provides a natural basis for compositional verification and so improves viability and efficiency of, for example, formal verification in that we need only verify smaller components. A secondary benefit is that modular architectures allow heterogeneous components to be employed. These may range across agents incorporating symbolic reasoning, deep learning-based object recognition, adaptive control for grasping, etc. All these components will have very different verification techniques applied to them, so the modularity of the architecture also supports the heterogeneous nature of verification across complex autonomous robotics [27].

4.1.2 Robot Architectures: Transparency

Second, there is a strong move towards *transparency* to ensure that designers, users, and assessors have a clear idea of what is in each component. This is

¹<https://committee.iso.org/home/tc299>.

Fig. 7.6 Transparent modular architecture. Copyright Dennis and Fisher, printed with permission in this chapter.

Fig. 7.7 Architecture with module specifications. Copyright Dennis and Fisher, printed with permission in this chapter.

particularly captured by standards such as the IEEE’s forthcoming P7001 standard on *Transparency of Autonomous Systems* [37] and through research on *Explainable AI* (see also Sect. 7.1). So, rather than just imposing modularity, we do not want many, or even any, of our modules to be *black boxes*. We require modular components to be *transparent*, i.e. we must be able to inspect and assess the internal behaviour of the module; see Fig. 7.6.

4.1.3 Robot Architectures: Verifiability

Being able to see each component’s code still might not help us—understanding adaptive behaviour just from a component’s implementation can be hard/impossible. For example, seeing the coding of a complex deep-learning network will likely not aid us in understanding exactly what it is doing. Instead, we usually require a concise, and ideally *formal*, description of the anticipated behaviour of each component.

In Fig. 7.7, each component’s specification describes what it is expected to do. Once we have this, we can select the most appropriate verification technique to assess how well it does, indeed, achieve its specification.

Software Engineering for Robotic systems is a complex task, and so modularity is often enhanced with descriptions (even metadata) of the intended behaviour, for example, Genom [34] (see Chap. 8). The formalisation of these descriptions paves the way for compositional reasoning, for example, through assume-guarantee tech-

niques [39]. Furthermore, our use of heterogeneous techniques is useful here [27, 55]. For example, a deep-learning vision component might most productively be verified by testing, an agent might be verified by program model-checking, a motion planning component might be verified by theorem proving, and so on.

4.2 Verification of Cyber-Physical Systems

Before turning to the formal verification of the discrete ‘agents’ of the type we described above, we provide a comment about the verification of more traditional cyber-physical systems. If we do not care about exactly why decisions have been made, only assessing what the system will do, then our systems usually fall within the automatic or adaptive categories described earlier. To carry out formal verification, especially of adaptive systems, we need a formal model of the environment in which the system resides. This formal model is usually quite complex with differential equations mixed with discrete computations (and will, of course, be wrong). This explains why the majority of formal verification in this area utilises hybrid system verification [3, 11, 43, 44] which in turn often relies on approximation [1, 2, 40].

5 Verifiable Autonomy

In this section, we will describe both our architectural basis and the verification techniques, in order to provide stronger verification of autonomous systems, especially their decision-making.

5.1 Robot Architectures: Responsibility

Following on from the discussion of software architectures in the last section, there is one further architectural aspect that will be crucial to us: the identification of key decision-making responsibilities, usually into identifiable and verifiable components.

As our systems become increasingly autonomous, it is important to be clear *where* key decisions are made and who (whether human or software) is *responsible* for these decisions. Furthermore, how is responsibility distributed, shared, and reinforced? This leads us to further architectural requirements wherein areas of responsibility, and mechanisms for changing them, are explicitly highlighted in decision-making components (agents), as in Fig. 7.8.

Autonomous systems must make their own decisions about what to do, and so such decisions are made in the software that controls the system. In our approach,

Fig. 7.8 Architecture with agent responsibilities. Copyright Dennis and Fisher, printed with permission in this chapter.

the high-level decision-making is captured as a *rational agent* [31, 60]. These software agents reason in a simplified, but structured, manner.

They have a set of ‘goals’ they wish to achieve, a set (often quite large) of ‘beliefs’ about the state of the world, and a set of ‘plans’ describing their known routes to achieving such goals. On top of these, agents typically have deliberation mechanisms that help decide which goals can be tackled, how to tackle them, and which are the most important to tackle at present [59]. All these aspects are then modified by the agent’s evolving context and its dynamic beliefs about the world. Such agents are the core of our approach to responsible autonomous systems.

5.2 Hybrid Agent Architectures

The requirements for *reasoned* decisions and explanations, as well as self-awareness (see later), have led on to our development of *hybrid agent architectures* [42] combining:

1. one or more *rational agents* for *high-level* autonomous decisions, and
2. a range of traditional *feedback control systems* for *low-level* activities,

These can be more *understandable*, easier to *program* and *maintain*, and, often, more *flexible* so providing benefits over monolithic control architectures [19]. An overview is provided in Fig. 7.9, and in subsequent sections we refine this basic concept to provide transparency and verifiability, self-awareness, fault tolerance and reconfigurability, and generality.

Our approach is that

- we should be certain what the autonomous system **intends** to do and how it **chooses** to go about this

and so the core *rational agent*

- must have explicit **reasons** for making the choices it does, and should expose/-explain these when needed.

Fig. 7.9 Overview of hybrid agent architecture. Copyright Dennis and Fisher, printed with permission in this chapter.

Example: From Pilot to Rational Agent Within a modern aircraft, the *Autopilot* software can essentially fly the aircraft ‘on its own’

- keeping on a particular path,
- keeping flight level and steady under environmental conditions,
- planning routes around obstacles, etc.

In many cases the *human* pilot will then be concerned with high-level decisions, such as

- where to go to,
- when to change route,
- what to do in an emergency, etc.

Once we think of constructing a fully autonomous version of this, the *rational agent* now makes the decisions the pilot used to make.

5.3 Verifying (Rational) Agents

As we saw in Sect. 3.2, we have many options of how to carry out formal verification, but for rational agents, we commonly choose to use *program model-checking* [52] whereby a logical specification is checked against the *actual* agent code that is used in the robot/system. We have developed a toolkit for such verification [16], based on the Java Pathfinder program model-checker [38, 51].

In Java Pathfinder, Java code is symbolically executed (through a modified Java virtual machine), and, as all the possible executions are explored, each is checked against the agent requirements. In Fig. 7.10, we can see (on the left) a backtracking interpreter exploring all possible branches in a program’s execution while (on the right) an automaton assesses the required properties on each execution path

Fig. 7.10 Representation of Java Pathfinder exploration, from [29]. Copyright Dennis and Fisher, printed with permission in this chapter.

explored. This combination of (backtracking) symbolic execution and a monitoring automaton in parallel is termed ‘*on the fly*’ model-checking [36].

5.4 Agent Java Pathfinder (AJPF)

There are many different agent programming languages and agent platforms based, at least in part, on the rational agent view above. These languages are essentially rule-based, goal-reduction languages, with the extra aspect that deliberation, the ability to change between goals and change between plan selection strategies at any time, is a core component. Consequently, they allow the representation and programming of high-level reasoning mechanisms that have many similarities with basic practical reasoning in humans [9]. The basic agent programming language we use is called Gwendolen [15], a language particularly built on the concepts of beliefs, goals (alternatively, *desires*), and intentions [48].

Gwendolen is built on Java and so Java Pathfinder (JPF) can be enhanced to verify Gwendolen programs rather than just Java programs. This requires the extension of the program model-checking approach with the additional agent aspects of Gwendolen (beliefs, plans, goals, etc., as above).

In addition to exposing the structure of the system, we need formal properties to specify. While standard formal specifications often utilise algebraic or temporal specifications, we now need to extend these with agent aspects. To describe these extensions, we will begin with standard temporal logics [29]. While classical propositional logic is typically used to capture *static* and unchanging situations,

we turn to propositional temporal logics to describe how the state changes as time progresses. As an example, the formula ' $this \wedge \bigcirc that \wedge \Diamond the_other$ ' means that *this* is true now (in the current state), *that* is true in the next state, and *the_other* is true at *some* state in the 'future'.

Clearly, for describing requirements of our rational agent, we should choose an appropriate way to extend temporal logic. However, there are very many options here, such as: *logics of knowledge*, *real-time temporal logics*, *probabilistic logics*, *cooperation or coalition logics*, etc. No simple logic exactly captures the behaviours of rational agents, and so, in realistic scenarios, we need to *combine* several logics. In our case (and this is typical of many rational agent logics [47]), we, at least in principle, combine:

- a temporal logic, to represent basic dynamic activity;
- a modal logic of *belief* (of the KD45 variety), to represent information [26]; and
- a modal logic of *intention* (of the KD variety), to represent motivation [47].

Combining logics can become very complex, very quickly, so, in practice, we actually use a restricted combination of simple linear temporal logic combined with (non-nested) representations of belief and intention, both referring to goals and plans that the agent has.

Example Specification A typical statement we might have concerning human-robot interaction:

- If a robot 'believes' that a human is directly in front of it, and it has a goal to get to a room beyond the human, then it should *never* choose a plan that brings it into proximity with the human, unless there is no alternative, in which case the robot may even decide to wait for the person to move or instead may decide to drop or revise its goal.

This can be formalised in the logic combining time, intention, and belief (as well as some formalisation of plans and actions).

Formal requirements can then be formally verified of our rational agent, allowing us to analyse the core decision-making in our autonomous system.

5.5 Heterogeneous Verification

Recall that our hybrid agent-based architecture is as shown in Fig. 7.11. Now, we can employ different verification techniques to different components within our architecture.

- We can *formally verify* the agent's decision-making and so can be certain about the way decisions are made.
- We can *simulate and test* or alternatively *monitor* the feedback control components. This either provides us with a probabilistic estimate (testing) or informs us when 'bad' behaviour occurs (monitoring).

Fig. 7.11 Hybrid agent architecture interacting with environment. Copyright Dennis and Fisher, printed with permission in this chapter.

- We can *practically test* the whole system. Although this gives relatively weak verification evidence, it provides important confidence to the developers, regulators, or users that the system *can* work in realistic scenarios.

So, when we verify autonomous robotic systems constructed using our hybrid agent architecture, we can say *for certain* what it will decide to do once presented with a particular scenario, but cannot guarantee that such scenarios will be recognised accurately. Similarly, once the agent decides what to do, we cannot guarantee that this will be carried out effectively.

Our agent essentially replaces the high-level, human decision-making so formal verification here has an important role in certification.

5.6 Problems: Requirements

As described earlier, if we are to carry out verification, and particularly *formal* verification, of a component, we need to be clear what specification we are verifying against. This leads us to the ongoing problem of *where* these requirements come from. There are two variations on this. In one version the requirements we are given of a system or component may be *extremely* vague and under-specified:

- Please verify the system is safe!
- Please verify the system is secure!
- Please verify the system is usable!
- Please verify the system is nice!

Not only are many of these aspects subjective ('nice', 'usable', etc.) and so more relevant to (user) validation techniques, but no further detail of terms such as 'safe' or 'secure' is given. There might be a link to a relevant standard (where this exists), but such standards are also vague and ambiguous.

A second version of the problem is where we just have *far too many* requirements. There may be precise *safety* requirements and precise *security* requirements.

Furthermore, there may be local regulations to incorporate or, especially where fully autonomous systems are used, *ethical* requirements. Then, if an autonomous robotic system (such as a domestic robot or a driver less car) is owned by a person, then they will likely require their preferences to be incorporated and so on through a variety of additional constraints.

Here the problem is that the number and size of requirements become vast. Especially when we carry out formal verification, this can be beyond the capabilities of contemporary verification tools. Furthermore, we might need quite complex mathematical apparatus even to represent all of these (originally, natural language) requirements. If we are carrying out formal verification on logical descriptions, this often means combining a wide range of logics, and this can easily push verification systems beyond their limits in terms of computational complexity of the combined logics considered or practical complexity of the size of the targets.

6 Applications: ‘Act as Humans Should’

A typical sequence of moving from automation to autonomy is to:

1. begin with an automated system (often with fixed behaviour);
2. move to human (and, probably, remote) control of the system;
3. and then gradually replace human decisions with autonomous ones.

It is step (3) where we particularly need verification of the autonomous behaviours. Specifically, we need to ensure that the decisions the autonomous system makes correspond to what the human operators would have done in step (2).

Once we can expose the high-level system decisions, we can match these against a range of ‘expected’ behaviours. In particular, we can match against specific legal requirements we might have. This is not always possible, especially if we have no prescribed behaviours for human operators, drivers, or pilots, but in some cases can be useful. Below we outline obvious example areas from the realm of autonomous vehicles.

6.1 Towards UAS Certification

This work, from [12, 53] and particularly [54], summarised below 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 ‘Rules of the Air’ [13] describe what a human pilot should do when in control of an air vehicle. If we replace human control by a rational agent, then clearly the agent should follow the rules prescribed for a human pilot; see Fig. 7.12. Note that the aircraft’s *autopilot* will take care of low-level flight skills, with the agent taking care of the higher-level decisions, e.g.

Fig. 7.12 Providing certification evidence for UAS. Copyright Dennis and Fisher, printed with permission in this chapter.

- when two aircraft are approaching head-on ... and there is danger of a collision, each shall alter its course to the right [13].

This type of requirement is then translated to a formal specification. For example, in our temporal logic of belief and intention (see Sect. 5.4, where ‘ $\Box$ ’ means ‘it is always the case’, $\Diamond$ means ‘at some future moment’, and **B** means ‘the agent believes’):

$$\Box(\mathbf{B} \text{ detected_aircraft} \Rightarrow \Diamond \mathbf{B} \text{ engage}(\text{emergency_avoid}))$$

ensuring that **emergency_avoid** will be engaged. This can then be verified of the agent. Note that the question of how effective **emergency_avoid** is in ensuring the aircraft turns to the right is assessed differently (through testing) as is the reliability of aircraft detection. Note also that the ‘**B**’ modality is necessarily weak; we cannot be certain about any aspect of the environment so belief is the best we can hope for.

There are many ‘Rules of the Air’, with many being ambiguous or imprecise (after all they are intended for human pilots) meaning that formalisation can be quite difficult. However, we would expect a trained pilot to adhere to these, and so if we have an autonomous system, it is our rational agent that should be responsible for following these rules.

In [54] we have developed a Gwendolen agent that can control, at a high level, a (simulated) autonomous air vehicle and have formalised a selection of the ‘Rules of the Air’. We used AJPF to verify that the agent’s decisions match the formal requirements. Note that, while verifying the above rule is relatively simple,

Fig. 7.13 Assessing ‘Rules of the Road’ in vehicles. Copyright Dennis and Fisher, printed with permission in this chapter.

increasingly complex rules together with a more sophisticated agent will lead to complex verification, requiring us to improve the verification efficiency [57].

6.2 ‘Driver less’ Car Analysis?

In road vehicles, technology is moving beyond adaptive cruise control, lane-keeping, and automated parking, towards more autonomous capabilities. Once we introduce software to control the vehicle, for example, when a driver is taking a rest, we will need much more sophisticated software analysis. Just as a rational agent controlling an autonomous air vehicle might be verified with respect to the ‘Rules of the Air’ (see the previous section), the rational agent controlling an autonomous road vehicle can be verified against the ‘Rules of the Road’ [4]. This follows a similar approach to the UAS work, as shown in Fig. 7.13.

In fully autonomous vehicles, the reliability of the agent will be crucial. This reliability is under-specified at present, but must eventually be able to cope with unexpected events, failures, aggressive road users, etc. Thus, the verification of suitable properties concerning the decision-making will be vital. (As ever, capturing these requirements, beyond prescribed ‘Rules of the Road’, will be difficult.)

7 Applications: Self-Awareness

7.1 Explainability

In some systems, especially those with opaque components, explainability can be difficult. Hence the popularity of the field of Explainable AI [25]. However, if we have constructed our autonomous system using a hybrid agent architecture, then the rational agent

1. has symbolic representations of its motivations (goals, intentions) and beliefs
2. reasons about these to decide what to do, and
3. records all the other options explored (and reasons for not taking them).

And so, it essentially has a trace of reasoned decisions. Then with a little work [41], we can provide explainability and transparency in decision-making. This allows us to provide

- recording facilities—such as Winfield/Jirotko’s *ethical black box* [56] (see also Chap. 6);
- interactive capabilities—such as a ‘why did you do that?’ button [41]; and
- advanced capabilities—such as a ‘what will you do next, and why?’ button.

In our approach, we do not use any *learning* components for high-level decision-making. Although it may be possible to incorporate them, work remains to be done on explaining new, learned behaviours.

7.2 Self-Awareness

Just as in psychology, where *self-awareness* involves internal models, we can enhance and implement introspection and self-awareness in autonomous systems by providing a range of internal models; see Fig. 7.14. The elements here comprise the following models:

- An *Interaction Model* describing the form of, and requirements for, interaction with operators, humans, inspectors, etc.
- A *Safety Model* capturing the assumptions and guarantees extracted from the system certification process, involving how techniques such as safety cases and fault identification identify safety aspects.
- A *Self Model* describing the components, architecture, and effectiveness of the system itself.
- A *Task Model* describing the goals, tasks, and schedules for the robotic system’s activities.

All of these are available to the agent and all are useful for different aspects.

Fig. 7.14 Internal models accessible from the agent; from [32]. Copyright Dennis and Fisher, printed with permission in this chapter.

Example Robots that must continue working for long periods will have failures in some of their subsystems. These failures could range from minor to severe, and, in the former case, we require the system to continue operating, though perhaps at reduced efficiency. In order to recognise such failures and, potentially, provide replacement behaviours, the agent must

- be aware of system components and their expected behaviours,
- be able to monitor its own performance, and
- be able to autonomously *reconfigure* its software.

All of these elements depend on an up-to-date and accurate *Self Model*. Each model could be a separate architectural component, or multiple models of these forms might be directly accessible by the rational agent. Furthermore, this form of internal structure provides possibilities for carrying out formal verification of activities related to self-awareness [18].

7.3 Awareness of Acceptable Boundaries

If we have a clear view of the envelope of behaviour for which our systems approved, then this can be captured within the *Safety Model*. Monitoring system behaviour can highlight when the system steps beyond (or is close to stepping beyond) legal limits.

A more general version of this involves capturing the behaviour boundaries and assessing when the agent moves outside these boundaries. When we carry out formal verification of the agent, there will be some assumptions about the environment in which the agent is embedded. In [23, 28] we showed the use of *Runtime Verification* to monitor whether our system is operating within the bounds where it had been verified. Specifically, when the original formal verification is carried out, a runtime monitor is also generated that can be used by the agent to allow it to check if it is reaching (or has passed) its verified envelope.

8 Applications: Beyond the Predictable

While conforming to legal requirements may be sufficient for many autonomous systems, a further question, particularly for systems deployed in domestic settings, is what happens in unpredictable situations or in situations where some violation is unavoidable.

Recall the example from Sect. 3.3, where a robot should rescue a person from a burning house. Although there may be a very strict rule about human contact, the robot helper decides to ‘break’ this and save the human. We are here beginning to veer into ethical, rather than straight-forwardly legal, issues.

Cointe et al. [14] integrate rational agents and ethical reasoning into a comprehensive framework in which agent reasoning determines sets of *desirable*, *feasible*, and *moral* actions and then uses context-sensitive ethical principles to select one action from these sets. We are not so comprehensive but, in the sections below, highlight how some ethical decision-making can be implemented and verified within our approach.

8.1 Simple Ethical Ordering

In [22] we considered a situation, where ethical reasoning is only invoked when none of the system’s 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 minimise casualties takes precedence over its duty to obey the Rules of the Air). In this system, we were able to verify that the aircraft, if forced into an emergency landing, would always land in a field rather than on a road.

This approach used straight-forward ethical priority ordering such as

save life >> save animals >> save property

Then the agent simply assesses the options and takes the most *ethical* one with respect to that ordering. Once the agent decisions take these *ethical concerns* into account, then we can extend formal verification to also assess these. For example, we can formally verify that

if a chosen course of action violates some substantive ethical concern, A

then the other available choices all violated some concern that was equal to, or more severe than, *A*.

Though simple, this 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 mechanism for exploring verifiable robot/machine ethics.

8.2 *Ethical Governors (See also Chap. 6)*

In [10] we explored the ethical issues further. We 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 an agent.

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 towards 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.

This approach was based on the use of an ethical governor utilising an internal simulation-based model [50]. As the ethical governor is essentially a rational agent, we verify this agent against ethical requirements and properties. In Fig. 7.15, we see how the ethical governor intercepts the actions the robotic system is deciding upon and either prunes or orders this set of options before returning them to the robot to choose. In this way, the ethical governor can remove any particularly unwanted options. It is important, however, for the ethical governor to have mechanisms for analysing the outcomes of various options and it is here that Winfield et al. utilise internal simulations [50].

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

8.3 *Multiple Ethical Theories*

Finally, we just mention our work in [6, 24]. Here, we allow and verify the modification of ethical theories and priorities as our autonomous systems change contexts. We provide hybrid agent architecture capable of ethical reasoning that allows agents to change their reasoning as contexts (both physical and social) change.

Fig. 7.15 Ethical governor and its verification. Copyright Dennis and Fisher, printed with permission in this chapter.

9 Responsibility

As we have discussed, an *autonomous* system must be able to make its own decisions, and even act, without direct human control. At this point, it becomes important to be sure what the system will choose to do, as it may make this choice without human input. We must not only be able to assess the autonomous decision-making but, in doing so, should be able to convince ourselves that the system will always endeavour to act responsibly and ethically. We term this *responsible autonomy*.

Irresponsible Autonomy Although we might aim for trustworthy and responsible systems, in many cases we cannot ensure these. Either:

1. we know for certain what the system is aiming to do and this will be unacceptable in some form (typically, unethical or irresponsible); or
2. we do not know what the system is exactly doing, what it is aiming to do, or why it makes the choices it does.

Both are bad! In the first case, we are clear about what the system is doing but we have chosen to program it to act badly. In the second case, we do not even know for certain what the system is doing. To distinguish these, we might term the first variety as *unethical* and the second as *irresponsible*.

A related aspect concerns how much we can, or should, rely on these components. Especially with autonomous systems in safety-critical scenarios,

- it is vital that we use strong verification techniques for any component we rely on.

We can be sure what it is trying to do (as above). As a corollary,

- we should never fully **rely** on any component that has not been formally verified.

It is vital that we design and build autonomous systems in ways that are more amenable to strong verification and provide transparency about intention.

Example [17] A robot is holding a hammer, but then drops it on a person's head. Either this was a *deliberate* act and the robot aimed to injure the person, or it was an accident and was due to some failure in, for example, its gripper control. From the viewpoint of public trust, the difference between these two options is *huge*. So, if we can formally verify the robot's decision-making processes to show that it will *never* decide to take an action it believes will cause harm, then this is a significant step on the path to increasing public confidence in this technology.

10 Concluding Remarks

The key, new aspect in *autonomous systems* is that the system is able to *decide for itself* about the best course of action to take. This is a significant change in the roles of engineered systems and requires very different approaches to those traditionally used for (human-controlled) cyber-physical systems. Our approach to this problem is to advocate systems architectures in which the core, high-level decision-making capabilities can be captured as a *rational agent*. Then, by *formally verifying* the rational agent, we verify not what the system does, but what it *tries* to do and why it decided to try! This issue of 'why' decisions are made is crucial. It becomes essential to be able to identify exactly *why* the system took this choice as this may be important in verification or certification; society may be more forgiving of entities that at least *try* to do the right thing.

Once we have delegated decision-making to the agent, then being able to ensure that it will make its future decisions for the right reasons is vital. Our work described here represents initial steps towards providing stronger guarantees of autonomous systems behaviour and so helping trustworthiness and responsibility.

Acknowledgements The work described in this chapter has been supported by EPSRC, over several projects on reconfigurable autonomy (EP/J011770), verifiable autonomy (EP/L024845), sensor systems (EP/N007565), and robotics for hazardous environments, specifically nuclear (EP/R026084), offshore (EP/R026173), and space (EP/R026092). This work was also supported by a Royal Academy of Engineering *Chair in Emerging Technology*. Finally, we would like to thank the organisers of, and participants in, the RoboSoft meeting at the Royal Academy of Engineering in London in November 2019 for their encouragement and feedback.

Apart from Figures 7.1, 7.2, 7.3, and 7.4, all images are copyright Dennis and Fisher, reproduced with permission in this chapter.

References

1. A. Abate, M. Prandini, J. Lygeros, S. Sastry, Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica* **44**(11), 2724–2734 (2008)
2. A. Abate, J.-P. Katoen, A. Mereacre, Quantitative automata model checking of autonomous stochastic hybrid systems, in *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control (HSCC)* (ACM, New York, 2011), pp. 83–92

3. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine, The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.* **138**(1), 3–34 (1995)
4. G.V. Alves, L. Dennis, L. Fernandes, M. Fisher, Reliable decision-making in autonomous vehicles, in *Validation and Verification of Automated Systems: Results of the ENABLE-S3 Project*, ed. by A. Leitner, D. Watzzenig, J. Ibanez-Guzman (Springer International Publishing, Cham, 2020), pp. 105–117
5. R.C. Arkin, Governing lethal behavior: embedding ethics in a hybrid deliberative/reactive robot architecture, in *Proc. 3rd ACM/IEEE International Conference on Human Robot Interaction (HRI)* (ACM, New York, 2008)
6. M. Bentzen, F. Lindner, L. Dennis, M. Fisher, Moral permissibility of actions in smart home systems, in *Proceedings of the FLoC 2018 Workshop on Robots, Morality, and Trust through the Verification Lens* (2018)
7. R.H. Bordini, M. Fisher, W. Visser, M. Wooldridge, Model checking rational agents. *IEEE Intell. Syst.* **19**(5), 46–52 (2004)
8. R.H. Bordini, M. Fisher, W. Visser, M. Wooldridge, Verifying multi-agent programs by model checking. *J. Auton. Agent. Multi-Agent Syst.* **12**(2), 239–256 (2006)
9. M.E. Bratman, *Intentions, Plans, and Practical Reason* (Harvard University Press, Cambridge, 1987)
10. P. Bremner, L.A. Dennis, M. Fisher, A.F.T. Winfield, On proactive, transparent, and verifiable ethical reasoning for robots. *Proc. IEEE* **107**(3), 541–561 (2019)
11. M.L. Bujorianu, J. Lygeros, Toward a general theory of stochastic hybrid systems, in *Stochastic Hybrid Systems: Theory and Safety Critical Applications*. Lectures Notes in Control and Information Sciences (LNCIS), vol. 337 (Springer, Berlin, 2006), pp. 3–30
12. N. Cameron, M. Webster, M. Jump, M. Fisher, Certification of a civil UAS: a virtual engineering approach, in *AIAA Modeling and Simulation Technologies Conference, AIAA-2011-6664, Portland*, 2011
13. Civil Aviation Authority, CAP 393 air navigation: the order and the regulations (2010). <http://www.caa.co.uk/docs/33/CAP393.pdf>
14. N. Cointe, G. Bonnet, O. Boissier, Ethical judgment of agents' behaviors in multi-agent systems, in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems (AAMAS)* (ACM, New York, 2016), pp. 1106–1114
15. L.A. Dennis, Gwendolen semantics: 2017. Technical Report ULCS-17-001, University of Liverpool, Department of Computer Science, 2017
16. L.A. Dennis, The MCAPL framework including the agent infrastructure layer and agent java pathfinder. *J. Open Source Softw.* **3**(24), 617 (2018)
17. L.A. Dennis, M. Fisher, *Verifiable Autonomous Systems—Using Rational Agents to Provide Assurance about Decisions made by Machines* (2021, in preparation)
18. L.A. Dennis, M. Fisher, Verifiable self-aware agent-based autonomous systems. *Proc. IEEE* **108**(7), 1011–1026 (2020)
19. L.A. Dennis, M. Fisher, N. Lincoln, A. Lisitsa, S.M. Veres, Reducing code complexity in hybrid control systems, in *Proc. 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)* (2010)
20. L.A. Dennis, M. Fisher, M. Webster, R.H. Bordini, Model checking agent programming languages. *Autom. Softw. Eng.* **19**(1), 5–63 (2012)
21. L.A. Dennis, M. Fisher, N.K. Lincoln, A. Lisitsa, S.M. Veres, Practical verification of decision-making in agent-based autonomous systems. *Autom. Softw. Eng.* **23**(3), 305–359 (2016)
22. L.A. Dennis, M. Fisher, M. Slavkovik, M.P. Webster, Formal verification of ethical choices in autonomous systems. *Rob. Auton. Syst.* **77**, 1–14 (2016)
23. L.A. Dennis, A. Ferrando, D. Ancona, M. Fisher, V. Mascardi, Recognising assumption violations in autonomous systems verification, in *Proc. International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (2018)
24. L.A. Dennis, M. Bentzen, F. Lindner, M. Fisher, Verifiable machine ethics in changing contexts, in *Proc. 35th AAAI Conference on Artificial Intelligence (AAAI)* (2021)

25. Explainable AI, <https://www.darpa.mil/program/explainable-artificial-intelligence>
26. R. Fagin, J. Halpern, Y. Moses, M. Vardi, *Reasoning About Knowledge* (MIT Press, Cambridge, 1996)
27. M. Farrell, M. Luckcuck, M. Fisher, Robotics and integrated formal methods: necessity meets opportunity, in *Proc. 14th International Conference on Integrated Formal Methods (iFM)*, ed. by C.A. Furia, K. Winter. Lecture Notes in Computer Science, vol. 11023 (Springer, Berlin, 2018), pp. 161–171
28. A. Ferrando, L.A. Dennis, D. Ancona, M. Fisher, V. Mascardi, Verifying and validating autonomous systems: towards an integrated approach, in *Runtime Verification*, ed. by C. Colombo, M. Leucker (Springer, Berlin, 2018), pp. 263–281
29. M. Fisher, *An Introduction to Practical Formal Methods Using Temporal Logic* (Wiley, Chichester, 2011)
30. M. Fisher, M. Singh, D. Spears, M. Wooldridge, Guest editorial: logic-based agent verification. *J. Appl. Log.* **5**(2), 193–195 (2007)
31. M. Fisher, L.A. Dennis, M. Webster, Verifying autonomous systems. *ACM Commun.* **56**(9), 84–93 (2013)
32. M. Fisher, E. Collins, L.A. Dennis, M. Luckcuck, M. Webster, M. Jump, V. Page, C. Patchett, F. Dinmohammadi, D. Flynn, V. Robu, X. Zhao, Verifiable self-certifying autonomous systems, in *Proc. 8th IEEE International Workshop on Software Certification (WoSoCer), Memphis* (2018)
33. M. Fisher, C. List, M. Slavkovik, A. Weiss, Ethics and trust: principles, verification and validation (Dagstuhl Seminar 19171). *Dagstuhl Rep.* **9**(4), 59–86 (2019)
34. S. Fleury, M. Herrb, R. Chatila, GenoM: a tool for the specification and the implementation of operating modules in a distributed robot architecture, in *International Conference on Intelligent Robots and Systems* (IEEE, Piscataway, 1997), pp. 842–849
35. Food Machinery Company — Automated Ready Meal Production line. www.foodmc.co.uk
36. R. Gerth, D. Peled, M.Y. Vardi, P. Wolper, Simple on-the-fly automatic verification of linear temporal logic, in *Proc. 15th IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification XV, London* (Chapman & Hall, London, 1996), pp. 3–18
37. Institute of Electrical and Electronics Engineers. P7001—transparency of autonomous systems, 2019
38. Java PathFinder, <http://javapathfinder.sourceforge.net>
39. C.B. Jones, Tentative steps toward a development method for interfering programs. *ACM Trans. Program. Lang. Syst.* **5**(4), 596–619 (1983)
40. A.A. Julius, G. Pappas, Approximations of stochastic hybrid systems. *IEEE Trans. Autom. Control* **54**(6), 1193–1203 (2009)
41. V. Koeman, L.A. Dennis, M. Webster, M. Fisher, K. Hindriks, The ‘Why did you do that?’ Button: answering why-questions for end users of robotic systems, in *Proc. of the 7th International Workshop on Engineering Multi-Agent Systems (EMAS)* (2019)
42. N. Lincoln, S.M. Veres, L.A. Dennis, M. Fisher, A. Lisitsa, 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)
43. C. Livadas, N.A. Lynch, Formal verification of safety-critical hybrid systems, in *Hybrid Systems: Computation and Control*. Lecture Notes in Computer Science, vol. 1386 (Springer, Berlin, 1998), pp. 253–272
44. A. Platzer, J.-D. Quesel, KeyMaera: a hybrid theorem prover for hybrid systems, in *Proceedings of 4th International Joint Conference on Automated Reasoning (IJCAR)*, ed. by A. Armando, P. Baumgartner, G. Dowek. Lecture Notes in Computer Science, vol. 5195 (Springer, Berlin, 2008), pp. 171–178
45. M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ROS: an open-source robot operating system, in *Proc. ICRA Workshop on Open Source Software* (2009)
46. F. Raimondi, A. Lomuscio, Automatic verification of multi-agent systems by model checking via ordered binary decision diagrams. *J. Appl. Log.* **5**(2), 235–251 (2007)

47. A.S. Rao, Decision procedures for propositional linear-time belief-desire-intention logics. *J. Log. Comput.* **8**(3), 293–342 (1998)
48. A.S. Rao, M.P. Georgeff, An abstract architecture for rational agents, in *Proc. 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR&R)* (Morgan Kaufmann, San Francisco, 1992), pp. 439–449
49. Vacuum Cleaning Robot, <http://www.irobot.co.uk/home-robots/vacuum-cleaning>
50. D. Vanderelst, A.F.T. Winfield, An architecture for ethical robots inspired by the simulation theory of cognition. *Cogn. Syst. Res.* **48**, 56–66 (2018)
51. W. Visser, P.C. Mehrlitz, Model checking programs with Java PathFinder, in *Proc. 12th International SPIN Workshop*. Lecture Notes in Computer Science, vol. 3639 (Springer, Berlin, 2005), p. 27
52. W. Visser, K. Havelund, G.P. Brat, S. Park, F. Lerda, Model checking programs. *Autom. Softw. Eng.* **10**(2), 203–232 (2003)
53. M. Webster, M. Fisher, N. Cameron, M. Jump, Formal methods and the certification of autonomous unmanned aircraft systems, in *Proc. 30th Int. Conf. Computer Safety, Reliability and Security (SAFECOMP)*. Lecture Notes in Computer Science, vol. 6894 (Springer, Berlin, 2011), pp. 228–242
54. M. Webster, N. Cameron, M. Fisher, M. Jump, Generating certification evidence for autonomous unmanned aircraft using model checking and simulation. *J. Aerosp. Inf. Syst.* **11**(5), 258–279 (2014)
55. M. Webster, D. Western, D. Araiza-Illan, C. Dixon, K. Eder, M. Fisher, A.G. Pipe, A corroborative approach to verification and validation of human-robot teams. *Int. J. Rob. Res.* **39**(1), (2020). <https://doi.org/10.1177/0278364919883338>
56. A.F.T. Winfield, M. Jirotko, The case for an ethical black box, in *Proc. 18th Annual Conference Towards Autonomous Robotic Systems*. Lecture Notes in Computer Science, vol. 10454 (Springer, Berlin, 2017), pp. 262–273
57. M. Winikoff, L.A. Dennis, M. Fisher, Slicing agent programs for more efficient verification, in *Proc. 6th International Workshop on Engineering Multi-Agent Systems (EMAS)*. Lecture Notes in Computer Science, vol. 11375 (Springer, Berlin, 2019), pp. 139–157
58. R. Woodman, A.F.T. Winfield, C.J. Harper, M. Fraser, Building safer robots: safety driven control. *Int. J. Rob. Res.* **31**(13), 1603–1626 (2012)
59. M. Wooldridge, *An Introduction to MultiAgent Systems* (Wiley, New York, 2002)
60. M. Wooldridge, A. Rao (eds.), *Foundations of Rational Agency*. Applied Logic Series (Kluwer Academic, Dordrecht, 1999)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.