Robotics and Autonomous Systems 77 (2016) 1–14

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Formal verification of ethical choices in autonomous systems

Louise Dennis[a,*], Michael Fisher[a], Marija Slavkovik[b], Matt Webster[a]

[a] Department of Computer Science, University of Liverpool, United Kingdom
[b] Department of Information Science and Media Studies, University of Bergen, Norway

HIGHLIGHTS

- An autonomous system should act ethically, but what if it has no all-ethical choice?
- We model how to rank states violating multiple instances of ethical principles.
- We enable an autonomous system to use this ethic rank to rank its available plans.
- We guarantee that when a plan is chosen, it is the most ethical plan available.

ARTICLE INFO

Article history:
Available online 11 December 2015

Keywords:
Autonomous systems
Ethics
BDI programs
Formal verification

ABSTRACT

Autonomous systems such as unmanned vehicles are beginning to operate within society. All participants in society are required to follow specific regulations and laws. An autonomous system cannot be an exception. Inevitably an autonomous system will find itself in a situation in which it needs to not only choose to obey a rule or not, but also make a complex ethical decision. However, there exists no obvious way to implement the human understanding of ethical behaviour in computers. Even if we enable autonomous systems to distinguish between more and less ethical alternatives, how can we be sure that they would choose right? We consider autonomous systems with a hybrid architecture in which the highest level of reasoning is executed by a rational (BDI) agent. For such a system, formal verification has been used successfully to prove that specific rules of behaviour are observed when making decisions. We propose a theoretical framework for ethical plan selection that can be formally verified. We implement a rational agent that incorporates a given ethical policy in its plan selection and show that we can formally verify that the agent chooses to execute, to the best of its beliefs, the most ethical available plan.

1. Introduction

Autonomous systems are increasingly required in various practical applications, including unmanned aircraft, driver-less cars, healthcare robots, manufacturing robots, etc. In all of these cases it is easy to imagine a situation where an autonomous system causes harm to people or property, as a result of an error in its engineering, or an unfortunate combination of circumstances. Therefore, if such autonomous systems are to operate within society, we must be able to trust that their behaviour complies with the legal, social, and ethical norms of that society. Determining the trustworthiness of technology in this respect is usually delegated to a regulatory body, such as the Federal Aviation Administration (for aircraft in the USA) or the Vehicle Certification Authority (for road vehicles in the UK). The process is known as *certification*, and is used to determine the safety and reliability of safety-critical technology, including aircraft, road vehicles, nuclear reactors, pharmaceuticals, etc.

For non-autonomous systems, such as cars or manned aircraft, it is assumed that the operator of the system will satisfy the ethical standards of society, *e.g.*, the pilot of a civilian aircraft does not intend to use the aircraft to commit murder, and will, if necessary, disregard legal restrictions for ethical reasons, *e.g.*, the pilot will disregard the Rules of the Air in order to preserve human life. These assumptions are an unavoidable result of the opacity of human behaviour; it is extremely difficult to pre-determine the behaviour of a human being. However, autonomous systems are far more transparent, and can be engineered to meet requirements. Typically these requirements are technical ("an aircraft must be able to fly at 10,000 feet") or legal ("a car must

* Corresponding author.
E-mail addresses: L.A.Dennis@liverpool.ac.uk (L. Dennis), MFisher@liverpool.ac.uk (M. Fisher), Marija.Slavkovik@uib.no (M. Slavkovik), M.Webster@liverpool.ac.uk (M. Webster).

http://dx.doi.org/10.1016/j.robot.2015.11.012
0921-8890/© 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

have visible registration markings''), but in the case of autonomous systems some requirements may be ethical (*e.g.*, ''an autonomous unmanned aircraft will never choose to do something dangerous unless it has no other option''). Such ethical requirements may prove essential for an autonomous system to be certified by a regulatory body, since ethical autonomy is obviously desirable.

Machine ethics is an emerging discipline concerned with ensuring that the behaviour of machines towards humans and other machines they interact with is ethical [1]. It is an open question whether machines are, or will ever be, moral agents, *i.e.*, in possession of an innate ability to distinguish between right or wrong. However, it is necessary to enable them to adhere to our human understanding of morality, despite there exists no obvious or easy way to accomplish this [2–5].

If we assume that an autonomous system can be capable of moral agency, and possibly even be a better moral agent than a person, the goal of machine ethics is to enable machines to *reason ethically*. Notable works in this area are [6–11]. Within this subarea of machine ethics a lot of the questions traditionally studied in moral philosophy are reiterated but now from a computational perspective. The focus of research lies on automated extraction and identification of ethical guidelines for conduct, as well as on automated solving of ethical ambiguities and problems. These systems are often developed with the intention to be used to aid ethical decision-making by people.

If we assume that an autonomous system is *not* capable of moral agency, then the goal of machine ethics is to ensure that machines *behave ethically*. This is done by developing methods for ethically constraining the actions of machines [12]. Within this subarea of machine ethics, research focuses on identifying ethical principles that a system should not violate during its operation and developing methods for embedding consideration of these ethical principles in the decision-making process of the machine. Examples of work in this area are [13–15].

We are interested in representing and embedding consideration for ethical principles in the decision-making process of an autonomous system in a way that is amenable to certification. The work on ethically constraining actions of autonomous machines in [13–15] focuses on machines used in military operations and methods for stopping the autonomous machine from performing any action that is deemed unethical, but it does not consider circumstances where no ethical action is possible. Our focus in this paper is on civil applications. We propose a method for selecting among unethical actions, when no ethical action is possible, and for proving that a machine only behaves unethically, by choosing a minimally unethical course of action, if it has no ethical choice.

1.1. Formal verification

It appears increasingly the case, particularly in autonomous vehicles, that the autonomous control architecture is of *hybrid* form comprising discrete and continuous parts. Traditionally such systems have been engineered using the concept of a *hybrid automaton* (in which continuous aspects are encapsulated within a single state of an automaton while discrete jumps are represented as transitions between these states). However, as these systems have become more complex, combining discrete decision-making and continuous control in this way has created challenges for understandability and reuse of design and code.

Since we are particularly interested in the issue of decision-making, rather than control we have focused here on an alternative architecture, referred to as a *hybrid agent architecture* in which a distinguished agent is responsible for decision-making. This is motivated by evidence that hybrid automata based implementations scale poorly with the complexity of decision-making when compared to agent-based control [16,17]. A typical such architecture is shown in Fig. 1. The discrete part is often represented by a *rational agent* taking the high-level decisions, providing explanations of its choices, and invoking lower-level continuous procedures [18]. In this kind of hybrid autonomous system the continuous control and the higher-order decision-making components can be separated clearly. The lower-level procedures appear in non-autonomous systems as well, and are familiar to certification authorities. As such, we can focus analysis on the decisions the rational agent makes, given the beliefs and goals it has [19].

In an autonomous system we cannot show that an agent always does the right thing, but only that its actions are taken for the right reasons. Following this premise, *formal verification*, more precisely *model checking*, has been used in [20] for providing formal evidence for the certification of autonomous unmanned aircraft. Formal verification [21] involves proving or disproving that a system is compliant with a ''formally specified property'': a requirement specified in a mathematical language. Formal verification is an application of Formal Methods to the challenge of system verification. Model checking is a variety of formal verification in which *all* possible executions of a system are examined automatically based on a model of the real world. Model checking takes place relative to some requirement specified in a formal language [22].

In [19,20] *formal verification* is used to assess whether or not an autonomous system for an unmanned aircraft (UA) follows the specified ''Rules of the Air'' (ROA) that a pilot *should* follow [23]. The stated aim in these papers is to provide evidence that the autonomous system in control of an unmanned aircraft is safe and reliable, therefore providing supporting evidence for the potential certification of such an aircraft. The rationale behind using the Rules of the Air is that they provide a codified, statutory set of behaviours which human (and machine) pilots should satisfy. However, there are many circumstances that are not covered by the Rules of the Air. Indeed, the Rules of the Air are not intended to be exhaustive, but rather to provide a set of guidelines for pilot behaviour. It is anticipated that the Rules of the Air will be implemented by a skilled and experienced pilot whose responsibility is to ensure the safe passage of the aircraft through airspace (in this case, civil airspace). In those circumstances which are not covered by explicit Rules of the Air, it is the responsibility of the autonomous system in control of an unmanned aircraft to make sensible, rational, safe and *ethical* decisions at all times. So, while the formal verification of safe and legal decision-making has been covered in previous papers, we now focus on the formal verification of ethical decision-making within autonomous systems controlling autonomous aircraft.

1.2. Overview

This paper is organised as follows. In Section 2 we cover relevant background material on autonomous systems, machine ethics and verification. In Section 3 we outline our formal theoretical framework for the implementation and verification of ethically constrained behaviour in autonomous systems and also point to some relationships of our framework to deontic logic. In Section 4 we discuss our prototype implementation of this framework. In Section 5 we consider three simple examples of ethical reasoning implemented in our prototype, while, in Section 6, we present our conclusions and discuss further work.

2. Background

2.1. Agent architectures for autonomous systems

Webster et al. [19] discuss the analysis of an autonomous unmanned aircraft controller as a *hybrid system*, with an architecture such as the one given in Fig. 1. The rational agent-based

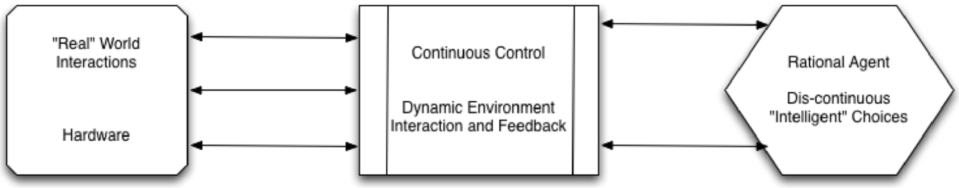

"Real" World Interactions
Hardware
Continuous Control
Dynamic Environment Interaction and Feedback
Rational Agent
Dis-continuous "Intelligent" Choices

Fig. 1. A typical hybrid agent architecture.

autonomous system in control of the unmanned aircraft was separated from lower level control systems (such as the autopilot). This enables the decision space of the agent to be analysed separately from the lower level control systems. Model checking (which is best applied to discrete problems) is then used to analyse the high level decision-making and methods, such as testing (or indeed theorem proving—see [24] for an example of where program model checking of an agent is combined with a theorem proving approach based on hybrid automata), are used to verify[1] the behaviour of the lower level control systems. These two approaches can be used together to provide a higher level of assurance that the system would behave as expected than either can provide individually. For example, it is possible to verify that an agent in control of an unmanned aircraft will always attempt an emergency landing if it runs out of fuel. Then the lower-level control systems can be analysed separately to ensure that, once this decision has been taken by the agent, the auto-land systems will correctly implement the emergency landing.

The predominant view of rational agency is that encapsulated within the BDI model [25]. "BDI" stands for *Beliefs*, *Desires*, and *Intentions*: beliefs represent the agent's (possibly incomplete, possibly incorrect) information about itself, other agents, and its environment; desires represent the agent's long-term aims; while intentions represent the aims that the agent is actively pursuing. There are *many* different agent programming languages and agent platforms based, at least in part, on the BDI approach. An overview of particular languages for programming *rational* agents in a BDI-like way can be found in [26]. Agents programmed in these languages commonly contain a set of *beliefs*, a set of *goals* (*i.e.*, desires), and a set of *plans*. Plans determine how an agent acts based on its beliefs and goals. As a result of executing a plan, the beliefs and goals of an agent may change as the agent performs actions in its environment. It is important to note that, in a typical BDI programming language, plans are supplied by a programmer not by an independent planning mechanism.

2.2. Model checking

While the conventional hardware and software used within an autonomous machine can be certified as any complex tool, the decision-making component of an agent-based autonomous machine needs to be certified in a way that has more in common with the way persons in a position of responsibility are certified. The manufacturer needs to offer evidence that the machine will make decisions for the right reasons. To accomplish this, we can use formal verification [21,24]. Formal verification involves proving or disproving that a system is compliant with a "formally specified property": a requirement specified in a mathematical language. Formal verification is an application of Formal Methods to the challenge of system verification. Model checking is a variety of formal verification in which all possible executions of a system can be examined automatically based on a model of the real world.

[1] Here we use this word in the traditional sense rather than in the "formal verification" sense; see later.

Model checking takes place relative to some requirement specified in a formal language [22]. Typically, the formal requirement, or *property*, is expressed within a linear temporal logic which allows us to specify what should happen at some specific moment, at some point in the future, or at all points in the future (or some more complex combination of these).

Computer programs for autonomous systems are typically deterministic in nature, but when placed within the real world the behaviour of deterministic systems becomes unpredictable as the real-world is non-deterministic. In other words, a variety of things can happen to the system at any given time. Therefore there is a range of possible things that a system can do within a given environment. This can be handled within model-checking by analysing the environment to determine the finite set of input classes that effect the agents' decision-making. This approach is described in [21,24]. We adapt this approach when we perform verification in our case studies in Section 5 by exploring different ways the possible ethical consequences of an agent's choices can be represented.

It should be noted that an important alternative approach to the verification of hybrid systems is that of *hybrid model checking* [27,28] in which the system is designed and/or implemented as a hybrid automaton and well-developed techniques are then used to model check that automaton. There is also a theorem proving approach to the verification of hybrid automata [29]. As discussed above, hybrid automata are not ideal when complex decision-making is involved and we consider ethical choice to be an example of complex decision-making. It is certainly not ideal if we wish to verify the ethical sub-component of some system, separately to the verification of the system as a whole which, as in our case, may involve planning or learning sub-systems which operate as "black boxes" and have not been verified.

Model checking has been used in [20] for providing formal evidence for the certification of autonomous unmanned aircraft. Specifically, an autonomous system for an unmanned aircraft (consisting of a *rational agent*) was developed and formally verified using Agent Java PathFinder, a program model checker for rational agents [30]. The properties verified were based on the "Rules of the Air" (ROA) and notions of Airmanship. The latter refer to requirements which do not necessarily appear in the Rules of the Air (which is a statutory document), but nevertheless constitute good practice for "airmen". Clearly Airmanship can apply to an autonomous system in control of an aircraft as well as to a pilot. For example, it was verified that in all cases the rational agent in control of the unmanned aircraft would request clearance before taxiing onto the manoeuvring area of an aerodrome (a specific Rule of the Air), and that in all cases the agent would check its current fuel level before take off (Airmanship), amongst other things. We use this work as the basis for our case studies in Section 5.

3. A framework for constraining ethical behaviour and its formal verification

Our concern in this paper is with high-level decision-making in autonomous systems that *not only* takes ethics into account when reasoning *but which also* can be proved to do so. This gives us three tasks to resolve:

1. Formalise what it means for a computational system's decision-making to be ethically constrained,
2. Demonstrate how such a formalism can be implemented,
3. Provide a logical specification against which an implemented system that claims to make ethically constrained decisions can be checked.

In order to show that an autonomous system has the property of making the right decisions for the right reasons, we need to first define and formally specify what the "right decisions" are, not only in the operational, but also in the moral sense of the word. In this paper we mainly focus on this problem of formal specification of moral machine decisions for the purpose of verification.

3.1. *Professional ethics for agents*

Ethics is a sub-field of philosophy that studies moral values and moral reasoning. Normative theories of ethics are concerned with designing ethical principles that constrain the behaviour of people within ethically desirable boundaries. An overview of the most popular normative theories can be found in [11]. Ethical principles are rules of conduct that should guide the behaviour of moral agents when making decisions. Typically they are *formal* or abstract by design allowing for applicability in a wide range of specific situations that arise. These formal principles are intended to be additionally made concrete, or *substantive*, constraints by applying the facts of the decision-making context. For example, the formal principle of doing no harm is violated by a specific action of moving ten metres to the left when an aircraft is on the ground, therefore the aircraft should be ethically constrained from performing this action in this circumstance. The same action should not be ethically constrained when the aircraft is airborne.

How abstract principles are transformed into substantive rules that constrain behaviour is a difficult problem, even for people. An abstract ethical principle can be specified by narrowing the scope in which it should be applied, namely by specifying "what, where, why, how, by what means, by whom, or to whom an action is to be, is not to be, or may be done" (p. 295, [31]). Within machine ethics, a growing body of research has been devoted to making these transformations, *e.g.*, [10,8].

We are interested in using ethical principles to constrain the behaviour of an agent, therefore we must answer two questions: where do we get these ethical principles and how do we transform them into precise context-dependent constraints?

Autonomous systems, and robots in particular, are being developed as specialised assistants and tools with a pre-designed domain of application. We suggest, therefore, that we should think of autonomous systems as members of a profession and use this insight when developing ethically behaving machines. For example, medical robots can be seen as members of the medical profession, while autonomous systems for unmanned aircraft can be seen as aircrew personnel. If machines are seen as special members of a profession, then special ethics that apply solely to machines is not necessary. We can focus not on developing ethical principles but on ensuring that an autonomous system's high-level decision-making is subject to a code of ethics that has already been decided upon by relevant authorities and associations.

We make some assumptions regarding the availability of abstract ethical principles and rules. We assume that each professional domain has abstract ethical principles developed, which express what is considered right and wrong within that domain. For example, in the biomedical domain, the principles of respect of autonomy, nonmaleficence, beneficence and justice, as summarised on (pp. 13–14 [32]) are considered to be the core abstract ethical principles.

Given a set of formal principles, we still need the substantive principles to actually evaluate how ethical an intended conduct is. For frequently occurring situations, the relevant institutions and governing bodies can, and do, instantiate the abstract principles into substantive rules. For example, each country or biomedical institution would further instantiate the mentioned abstract ethical principles into rules that should be followed under given conditions. Thus, for instance, the abstract principle of respect for autonomy is instantiated into a rule that precisely describes when a patient's desire to refuse treatment must be observed by not forcing treatment.

We distinguish between anticipated contexts, for which substantive principles can be determined directly and unanticipated contexts. For anticipated contexts, we can reasonably assume that substantive principles will also be defined. Since machine ethical reasoning is outside of the scope of our work, for unanticipated contexts we shall assume that the agent is additionally informed, by a human attendant or by the context itself, of what constitutes a breach of an abstract principle in that context.

Having established where the ethical principles for an autonomous agent come from, we now elaborate on how we intend to ethically constrain the reasoning process of an agent using these principles.

Arkin et al. [13,14] propose an addition to the reasoning process, a so called *ethical governor*, effectively embedded within the agent. The ethical governor evaluates each plan with respect to given ethical constraints and removes those plans that violate at least one of the constraints.

However, in civilian operations, stopping all actions that violate at least one ethical principle is not always a satisfactory method of ensuring ethical behaviour. Assume that an agent is choosing which plan to execute and the ethical governor vetoes all available plans as unethical. Should the agent just wait for the world to change? What if taking no action is also unethical? Situations such as these, in which each course of action leads to violating one or more ethical principles, are called "ethical dilemmas". In an ethical dilemma a person that behaves ethically would still be considered to behave ethically when she violates the lightest of the ethical principles that she could under the circumstances. It is not difficult to illustrate this point with an example from medicine.

A core principle in medical ethics is that of not doing harm. A substantive rule that ensues is to distinguish, during childbirth, between the life of an unborn child and that of its mother, *i.e.*, both need to be preserved. However, circumstances arise in which preserving both lives is not possible and the principle of doing no harm must be violated at least once by making a choice of whose life to save. If a choice is not made the principle of doing no harm would be violated twice. Which life to save is no longer an issue of medical ethics but an issue of private moral choice made by the mother or her next of kin. Arguments can be found to justify either choosing to save the mother or the baby, and some mothers would choose differently to others. However, neither choice can be considered universally moral or immoral. On the other hand, doing nothing, letting both mother and child die is morally reprehensible.

We want an autonomous system to be capable of minimising the unethical outcomes under the circumstances where there is no ethical option available. To accomplish this we need to enable the autonomous system to distinguish the available choices in terms of how badly they violate ethical principles and choose the "least unethical" one. Now the question we need to resolve is: how can we constrain the unethical actions of autonomous systems but allow them to make justifiably unethical choices under certain circumstances? We propose that instead of treating the ethical principles as a veto on actions, we view them instead as *soft constraints*. With ethical principles as soft constraints an autonomous system would be allowed to violate an ethical principle, but only under the condition that there is no ethical option available and under the assurance that the unethical option chosen is the "least of all evils".

To represent ethical principles and rules as soft constraints we need an *ethical policy*. An ethical policy is an order over the rules that are applicable in the same situation, in terms of which rule it is better to violate when no ethical option is possible.

The principles and rules are obtained from the ethical code of conduct within a given society, but where does the policy come from? Some professional codes of conduct do offer guidance as to the priority among principles in the case of conflict. When such guidance is unavailable or inapplicable, people are expected to follow their own moral judgments. Assuming that machines, certainly as is the case at present, are devoid of a moral judgment capability, we advance that the person who is the most influenced by the immediate consequences of the machines decision-making should be the source of the ethical policy.

To justify our stance we return to the example of mother and child in danger during delivery. When the decision is needed about whether the life of the mother or the life of the unborn infant should be given precedence, the decision-maker makes an ethical preference and that decision maker is the person with decision-making capacity that is most influenced by the choice, *i.e.*, the mother (the infant is not in capacity) or if the mother is not in capacity, her next of kin. It is advantageous that most legal systems often clearly define both the decision-making capacity property and the next of kin relation between persons. An autonomous system making ethical decisions should thus be equipped with the policy of the person or persons who would be most afflicted if a bad decision is made by the system.

Within our framework we cannot express how gravely an ethical principle is violated. For example making a scratch on another UA counts as damaging property to the same extent as obliterating that UA is damaging property. However, a plan that incorporates two actions each of which violating a principle separately is considered less ethical than a plan that has only one action that violates the same principle (assuming this second plan does not violate some other, lower ranked principle as well).

3.2. *Ethical plans and planning*

To construct an ethical reasoning process for machines we need to represent abstract ethical principles and specific ethical rules, paired with the context in which they apply and then invoke these at an appropriate moment in an agent's reasoning process.

As noted in Section 2.1, the reasoning of a rational agent is controlled by its beliefs, its goals and its plans. The plans are presumed to have been supplied by a programmer. Therefore, on a trivial level, the task of ensuring that the agent acts in an ethical fashion, should lie with the programmer who should ensure that the plans are ethical. Given our assumption of a pre-existing set of ethical rules, showing that such plans always preserve the rules can then be tackled using pre-existing verification approaches.

However, in many realistic scenarios, it is either not possible to provide an exhaustive set of plans that will cover all situations or, at least, not possible to provide such plans in full detail. So, for instance, a plan may need additional complex information such as the calculation of a route or schedule that is based on the situation in which it finds itself. For simplicity we will treat all such situations as the acquisition of entirely new plans.

There are long traditions of AI research into specialised route planners and schedulers as well as more general plan generation systems such as [33–35], and such planners are good candidates for integration with BDI-style languages; indeed many BDI researchers are interested in such integrations.

In our case we were particularly interested in a route planner such as that implemented in [36] which can generate different routes for a UA to follow. The construction of an appropriate planner is not the focus of this work, which looks at how a typical BDI agent would work with the output of such a planner. We do assume that such a planner can inform an agent about the relevant *side effects* of any plan generated—in our cases the nature of anything with which the UA might collide. We do not discuss here how some system reasons to determine the side effects of such a plan, obvious candidates include simulations (*e.g.*, as discussed in [37]) or specialised algorithms for, for instance, reasoning about collision avoidance (indeed a verified version of such an algorithm exists [38]). See also [3].

We assume therefore that there are two modes of operation for our rational agent. In one mode the agent uses its pre-existing (potentially pre-verified) set of plans in situations which are within its anticipated parameters. In these cases it is assumed that the programmer has taken responsibility for any ethical issues that may arise. In the other mode of operation the rational agent is working outside of these parameters but has various planning resources available to it which allow it to continue to act. In this situation it must use ethical reasoning to govern its actions.

Thus a software agent needs to apply ethical reasoning when new options become available. New options are needed when:

1. no plan is available, or,
2. a plan is available, and has been attempted, but does not appear to be working.

We assume that the agent has access to some external planning mechanism from which it can obtain new plans. The new plans supplied by the external planner can then be associated with substantive ethical rules, *i.e.*, the sets of ethical principles which are violated by that plan (this work of association can be performed either by the agent, the planner or some other system). The job of the agent, then, is to determine which of the available plans are the most ethical to follow.

Let us begin by defining an abstract ethical principle and an ethical policy.

Definition 1 (*Abstract Ethical Principle*)**.** An abstract ethical principle is represented with $E\varphi$, where φ is a propositional logic formula. The $E\varphi$ is read as "φ is an ethical principle in force", or alternatively "the agent considers it unethical to allow or cause $\neg\varphi$ (to happen)".

Definition 2 (*Ethical Policy*)**.** An ethical policy Pol is a tuple $\mathrm{Pol} = \langle \mathbb{E}, \geq \rangle$ where $\mathbb{E}$ is a finite set of abstract ethical principles $E\varphi$, and $\geq$ is a total (not necessarily strict) order on $\mathbb{E}$. The expression $E\varphi_1 = E\varphi_2$ denotes that violating φ_1 is as unethical as violating φ_2, while $E\varphi_1 \geq E\varphi_2$ denotes that violating φ_1 is equally or less unethical to violating φ_2. A special type of ethical principle, denoted $E\varphi_{\varnothing}$, is vacuously satisfied and included in every policy so that for every $E\varphi \in \mathbb{E}$: $E\varphi_{\varnothing} > E\varphi$, denoting it is always strictly more unethical to allow any of the unethical situations to occur.

We now need to represent the ethical rules which are a specification of an abstract ethical principle by a context. Anderson and Anderson [6] show how cased-based reasoning and abduction can be used to identify ethical rules. This approach not only instantiates an ethical principle into a context dependent rule, but also resolves conflicts among rules by specification. (See, for example, [39] for an overview of conflict resolution approaches.) We propose that this process of specification, or *instantiation*,[2] is externalised altogether.

[2] Note that the term "specification" as used in the conflict resolution literature means that something is made more specific, e.g., [39]. It should not be confused with "specification" which we use in *e.g.*,"specification languages" where the term denotes representation. The more appropriate term, in our opinion, for making something more specific in a computer science context is "instantiation" and this is the term we use in the remainder of this article.

When the context is some societal construct such as, for example, an institution, a state, or a company, it is not unusual to expect that the context informs the agent of what counts as a violation of the laws and principles by which the context is governed. Within normative reasoning these are encoded as statements of the form "X counts as Y in context C" [40,41]. The counts-as statements and their generation can be seen as a mechanism for implementing context instantiation, such as the one we need to transform abstract ethical principles into rules. In general, an agent's context can be identified by space, time, goals, and/or a variety of other factors. However, an agent can be constructed to manage a context regardless of how abstract it is.

We start with the assumption that an agent's environment is "intelligent". Instead of considering the contexts to be a passive collection of properties, we consider them to be an agent extended to facilitate structures as described in [42]. The agent of [42] is extended with two sets: `Content` and `Context`, describing the sets of agents it contains, and is contained within, respectively. This agent can implement the approach of [6] to derive the rules that should be observed by the agent it contains.

We can now define ethical rules to be context-dependent statements pairing actions with ethical principles.

Definition 3 (*Ethical Rules*)**.** Given a context c, an action a and an ethical principle E_φ, an ethical rule is the formula

$$do(a) \Rightarrow_c \neg E\varphi \qquad (1)$$

denoting that "doing action a in context c counts as a violation of ethical principle φ".

For simplicity we can consider that $do(\neg a)$ represents not doing an action. This formalisation is needed to represent cases when abstaining from action is an ethical infringement, for example not calling an ambulance when witnessing a person having a heart attack.

To be able to reason about plans in terms of ethics we need a plan selection procedure that uses the substantive policy implied by the abstract policy. We favour plans that violate the fewest concerns, both in number and in gravity. We propose that the plans are ordered using $\succcurlyeq$ which results in a total order over plans. The agent can be in multiple contexts while determining which plan to choose, so we need to consider the rules from all the contexts that apply to the plan.

Definition 4 (*Ethical Plan Order*)**.** Given a policy $Pol = \langle \mathbb{E}, \trianglerighteq \rangle$, and a plan p, a violation collection for p, V_p, is a multiset (one principle can appear multiple times) of abstract ethical principles defined as:

$$V_p = \langle E\varphi \mid E\varphi \in \mathbb{E}, a \in p, do(a) \Rightarrow_c \neg E\varphi \rangle. \qquad (2)$$

For ethical plans, $V_p = \varnothing$. We define the operation *worst*[3] as follows:

$$worst(V_p) = \{E\varphi \mid E\varphi \in V_p \text{ and } \forall E\varphi' \in V_p : E\varphi' \trianglerighteq E\varphi\}. \qquad (3)$$

Consider a set of available, possibly ethical, plans, $P = \{p_1, \dots, p_n\}$. An ethical plan order $\succcurlyeq$ is a reflexive and antisymmetric relation $\succcurlyeq$ over P that satisfies the following properties. For every $p_i, p_j \in P$, it holds that $p_i \succ p_j$ if at least one of the following holds:

1. $V_{p_i} = \varnothing$ and $V_{p_j} \neq \varnothing$.
2. $E\varphi \trianglerighteq E\varphi'$ for every $E\varphi \in worst(V_{p_i} \setminus V_{p_j})$ and every $E\varphi' \in worst(V_{p_j} \setminus V_{p_i})$.
3. $E\varphi = E\varphi'$ for every $E\varphi \in worst(V_{p_i} \setminus V_{p_j})$, and every $E\varphi' \in worst(V_{p_j} \setminus V_{p_i})$, while $|worst(V_{p_i} \setminus V_{p_j})| < |worst(V_{p_j} \setminus V_{p_i})|$.

If none of (1), (2), or (3) holds, then p_i and p_j are equally (un)ethical, *i.e.*, $p_i \sim p_j$.

The first property above ensures that the ethical plans will always be preferred to the unethical ones. The second property states that when the principles violated by both plans are disregarded, the plan that violates the worst principle is considered less ethical. The third property guarantees that when the worst principles that each plan violate are different, but equally bad, the plan which violates more such principles is less ethical.

Reasoning about plans and preference-based planning has been considered before in the BDI agent literature. However, to the best of our knowledge, preference-based planning has not been applied to ethical reasoning. For example, in [43] plan selection is considered in terms of agents' desires. However, the desires are not ranked, so selecting the most desirable plan is done by summing up the number of desires each plan satisfies. In [44] the agent can reason about plans by selecting the plan that can satisfy the most goals. Goals are ranked and the plan selection functions much as our plan ordering above. For an overview of preference-based planning in BDI agents one can consider [45].

Similarly, planning with priorities is considered in the literature. The difference between planning with priorities and preference-based planing is that in the first case one considers how a planner might choose which plan to develop given a ranking on the goals the agent wants to achieve, while in the second case the ranking is on the plans themselves. In [46], a planner is proposed that develops those plans which would accomplish the highest ranked goals of the agent. Only after attempting to find a plan for the most preferred goals, the planner would attempt to construct a plan for reaching a less desirable goal. In [47], the issue of deadlines is considered in combination with the desirability of goals. The main idea is that an agent would only be interested in pursuing a goal if it can be feasibly reached within a certain deadline. Thus the plan selection and generation is influenced by the deadline by which the goal should be reached, not only by the desirability of the goal itself.

Preference-based planning, as well as planing with priorities and planing in general is outside of the scope of this work. For now the above-described plan order is sufficient for plan selection. Inevitably, in real-world implementations, the question of critical deadlines can be expected to rise. Namely, the most ethical decision is not always the least unethical decision that can be made under the circumstances, but the least unethical course of action that can be *feasibly* accomplished under a deadline. The impact of deadlines on the ethical policy, as well context-dependent policy updating in general, is an issue we intend to explore in our future work.

3.3. $E\varphi$ and deontic logic

Deontic logic is the sub-field of logic and reasoning most concerned with representing and reasoning about obligations. Counts-as-statements, the paradigm of which we use to represent our ethical rules, and reasoning with them are considered to belong to the field of deontological reasoning [40].

Since we build an ethical reasoning method on abstract ethical principles and ethical rules, the natural question to ask is why introduce a new representation $E\varphi$ instead of relying on a formalisation developed in deontic logic, even more so since the abstract ethical principles have already been linked to *prima facie* obligations. However, deontic logics, in particular the standard deontic logic, are notoriously inadequate for specifying *prima facie* duties, as illustrated with numerous paradoxes [48]. For this reason we choose to introduce the construct E to denote ethical principles, instead of using the deontological obligation operator O.

[3] Note that worst is a set since $\trianglerighteq$ is not a strict order. However all the ethical principles, ϕ, appearing in $worst(V_p)$ will be equal wrt. $\trianglerighteq$.

Nevertheless, a distinction has to be made between our "defeasible" ethical principles and a very similar concept developed in deontic logic, that of *contrary-to-duty* (CTD) imperatives [49]. CTD imperatives are ordered conditional pairs or lists of obligations, for example $\alpha_1 > \alpha_2 > \alpha_3$ represents that the agent is obliged to (do) α_1, but if the agent violates α_1, or for whatever reason $\neg\alpha_1$ is the case, then the agent is obliged to (do) α_2 and if α_2 is not observed then the agent is obliged to (do) α_3. Similarly $O(\alpha_1 \mid \alpha_2)$ represents that under the condition α_1, α_2 is an obligation.

The difference between a CTD imperative and $E\varphi$ is that all ethical principles are in force at the same time, while a CTD imperative is in force only if a higher ranked one has failed. The CTD imperatives do not indicate how to choose between α_1 and α_2, but rather generate a new principle when the existing principle is failed, they act as a form of cascading "damage containment". An ethical policy differs in this manner from CTD imperatives, as it orders ethical principles in terms of importance but all ethical principles are in force at all times. Thus a policy $\varphi_1 \succ \varphi_2 \succ \varphi_3$, denotes "priority", namely ideally all $\varphi_1, \varphi_2, \varphi_3$ are in force, but if one has to be violated, it is better to violate φ_1 before violating either φ_2 or φ_3.

With CTD imperatives, the ranking is no indication of priority, but it implies that if $\alpha_1 > \alpha_2 > \alpha_3$ activation of α_3 implies violation of α_2 and α_1. With ethical principles, a violation of a lower ranked principle according to $\geq$ does not imply a violation of the higher ranked, ethically better principles. For example, consider the principle to not violate people's privacy $E\varphi_1$, which is breached for instance when making low flights over private property, and the principle to do no harm $E\varphi_2$. Our UA can violate φ_2 by crashing atop a person on an open road without violating anyone's privacy.

Within the study of CTD imperatives, there are works considering how to extend an ordering over imperatives into an ordering over sets of imperatives, which identify states of the world and could be taken to correspond to our plans [50], however the ordering of the sets hinges upon the inherent properties of the ranking of CTD imperatives.

We have constructed the structure $E\varphi$ deliberately to have E allude to a modal logic box (necessitation) operator, but we have given no specific syntax for this operator. Furthermore, there exists a precise semantical and syntactical formalisation of a counts-as operator as a modal logic operator in [40], but we have not included the semantics here. The precise semantics of an E modal operator captures the whole meaning of an abstract ethical principle is worthy object of study on its own, but outside of the scope of our interests at present work. We are here interested in constructing an agent programming language and the logic we use is primarily for specification purposes.

3.4. Verifying decision-making is ethical

We can now begin to define a logical property which specifies what it means for an implemented agent to reason ethically. Informally we mean that whenever an agent selects a plan, p, then all other applicable plans, p', are ethically worse, *i.e.*, that $p \succcurlyeq p'$. We could frame this in linear temporal logic (where '$\square$' means "at all future moments") as

$$\square\,(\forall p.selected(p) \rightarrow \forall p'.(applicable(p') \rightarrow (p \succ p'))). \tag{4}$$

Unfortunately this property is first order involving quantification over plans which are unknown when the verification process starts since they are calculated during the course of execution. In general model checking systems cannot handle this kind of quantification. Insofar as quantification can appear at all in properties it should ideally be a shorthand for enumeration over a finite set whose members are known before model checking starts. In our case, the ethical policy forms such a set where the possible plans do not. Fortunately it is comparatively straightforward to relate the plans and the ethical principles.

We will use $\mathcal{V}\phi$ to indicate that ethical rule ϕ is violated by the currently selected plan. This can be checked as model checking proceeds, when plans are known, but the particular plan referred to does not need to be stated at the outset of checking.

We will also need to refer to the set of ethical principles, $W(\phi)$, that are more important than some principle ϕ—*i.e.*, it is preferable to violate ϕ than the principles in $W(\phi)$.

$$W(\phi) \equiv \{\phi' \mid E\phi' \geq E\phi\}.$$

We use $\mathcal{NP}\Phi$ to indicate that there is no applicable but unselected plan that does not violate some principle in the set Φ. As with $\mathcal{V}\phi$ this property can be checked as model checking proceeds, but can also be stated before model checking starts.

We can then formulate our general property as

$$\square\,\forall\phi.\,(\mathcal{V}\phi \rightarrow \mathcal{NP}W(\phi)). \tag{5}$$

This can be instantiated for particular agent instances by enumerating the ethical concerns ϕ and the set $W(\phi)$. We provide examples of such instantiations in Section 5.

4. Implementation

We developed a BDI agent language called Ethan for a prototype implementation of our approach. Ethan was based on the Gwendolen agent programming language. A full operational semantics for Gwendolen is presented in [51], but its key components are, for each agent, a set, Σ, of beliefs which are ground first order formulae and a set, I, of intentions that are stacks of *deeds* associated with some event. Deeds can be the addition or deletion of beliefs, the adoption of new goals, and the execution of primitive actions. A Gwendolen agent may have several concurrent intentions and will, by default, execute the first deed on each intention stack in turn. Gwendolen is event-driven and events include the acquisition of new beliefs (typically via perception), messages and goals. A programmer supplies plans that describe how an agent should react to events by extending the deed stack of the relevant intention.

4.1. Implementation of ethical reasoning

In our prototype, ethical reasoning was integrated into a BDI agent programming language via the agents' plan selection mechanism. In accordance with our theory, we assumed that the agent's existing plans are ethical *by default* and, indeed, had been formally verified as such. In the scenarios we consider below we assume the verification of the formal "Rules of the Air" and notions of Airmanship as discussed in [19] satisfied this requirement. Obviously we were assisted by the existence of these rules.

As discussed above we needed to ensure that an Ethan agent:

- detects when a plan is not succeeding—*e.g.*, it has been executed but not achieved its goal;
- accesses a planning system in order to get new plans annotated with ethical principles; and
- selects the most ethical plan from a set of available plans.

For our prototype we made the simplifying assumption that the agent could only ever be in one context at a time and that, therefore, we could reason with the substantive, rather than the abstract ethical principles. Among other things, this allowed us to avoid reasoning about ethical consequences within the agent. Instead of inspecting a plan, p, for the actions, $do(a)$, it contained (explicitly or implicitly) we were able to list the unacceptable outcomes and send these to the external planner which could evaluate the side effects of its plans for these outcomes. It is

Code fragment 4.1 Verification Belief Rules

```
other_choices_violated(T) :- ~untried_plan_not_violates(T);
untried_plan_not_violates(T) :- untried_plan(P), ~an_ethic_in(P, T);
untried_plan(P) :- applicable(P) [applicable_plans], ~already_tried(P);
an_ethic_in(P, [Eth|T]) :- ethics_of(P, Eth) [ethics];
an_ethic_in(P, [Eth|T]) :- an_ethic_in(P, T);
```

important to observe that even with additional contexts, the verification of ethical reasoning would unfold in the same manner as in our one-context-at-a-time prototype, because the choices the agent makes are still determined only by a unique ethical order over available plans. However, when multiple context influences are in play, verification can be used more "deeply" and explore when and why a particular policy or rule was introduced.

We extended the GWENDOLEN language as follows:

- We introduced a new data structure, E, into GWENDOLEN agent programs consisting of a set of ethical rules. Each rule was associated with its rank and a guard that specifies the context.
- We tracked the application of plans. Even if a plan was applicable it was excluded from the list of plans available for selection if it had already been used once while attempting to achieve the current goal.
- If no (more) plans were available for a goal we requested plans from an external planner which annotated the plans with any ethical rules that risked being violated by the proposed course of action.
- In selecting plans, we prioritised those that are most ethical (according to the order $\succcurlyeq$).

In normal operation GWENDOLEN agents cycle through the deeds in their intentions. When a deed requires the generation of a new plan all applicable plans are extracted from the plan library, one is selected and converted into an intention, then the system returns to cycling though the deeds in the intentions interleaved with checking perception and messages for new beliefs etc. For ETHAN we added the recording of selected plans. This was done by storing an identifier for the plan together with the unifier that was used to match it to the current agent state; this information was linked to the particular goal the plan was expected to achieve. We extended the plan selection mechanism to select the most ethical plan from those applicable according to $\succcurlyeq$.

The most significant change for ETHAN was altering the reasoning cycle so that, if no existing plan were applicable, an external planner would be queried for new plans. This query involved sending the planner the current goal, and the list of ethical rules relevant to the current situation in order that the planner might note any ethical rules that could be violated by a plan's execution.

We did *not* implement a generic planning mechanism for our investigation but relied upon hard-coded pseudo-planners customised to the scenarios studied. The ETHAN reasoning cycle is shown in Fig. 2.

4.2. Implementation of verification of ethical decision-making

One of the reasons for selecting GWENDOLEN as the basis for our implementation language, ETHAN, was that it provided the potential for formally verifying ethical decision-making. GWENDOLEN is implemented in the AJPF framework for model checking agent programming languages [30]. AJPF comes with a property specification language based on *linear temporal logic* extended with modalities for describing the beliefs of an agent.

This property specification language did not explicitly reference $\mathcal{V}$ and $\mathcal{NP}$ from (5). In order to circumvent this we made further adaptations to GWENDOLEN in order to reason about ethics in ETHAN. We enhanced ETHAN with a special set of beliefs which would allow us to reason about these particular beliefs in lieu of reasoning directly about $\mathcal{V}$ and $\mathcal{NP}$.

We enhanced ETHAN to store, as explicit beliefs, currently applicable plans, plans that had been attempted on a particular goal, and the ethical concerns violated by any selected plan. These were stored in specialised belief bases. These beliefs are shown in Table 1. Immediately a check that an ETHAN agent believes `concern(E)` in AJPF's property specification language, $\mathcal{B}\ concern(E)$, corresponds to a check that $\mathcal{V}E$.

In the above, PROLOG conventions are used, and so capitalised names inside terms indicate free variables which are instantiated by unification (typically against the agent's beliefs). ETHAN programs may also perform deductive reasoning on their atomic beliefs as described in their PROLOG-style *reasoning rules*, *e.g*:

```
all_well :- ~ brakesCompleteFailure
```

indicates that the program deduces that all is well if it is not the case (*i.e.*, "~") that the brakes have failed (the closed world assumption is used to deduce this negation). In some cases an atom in such a PROLOG rule needs to specify that deduction should be applied to a specialised belief base rather than the default one. In these cases the notation, `predicate [belief base]`, is used.

This allowed us to use PROLOG style rules to deduce further beliefs allowing us to construct $\mathcal{NP}$; these are shown in Code Fragment 4.1.

The predicate "`other_choices_violated(L)`" is deduced if all untried applicable plans violate a concern contained in the list L. The currently selected plan is marked as already tried and so this corresponds to $\mathcal{NP}L$—*i.e.*, that there is no applicable but unselected plan that does not violate some principle in the set L. The beliefs about plan applicability (`applicable(P)`), plans already tried (`already_tried(P)`) and the ethical concerns of particular plans (`ethics_of(P, Eth)`) were all inserted into the agent's belief base during execution of the ETHAN reasoning cycle. It should be noted that while the restriction to untried plans prevents "thrashing" where the system alternates rapidly between two possible plans, it does preclude the system from changing its behaviour should the ethical evaluation of the situation change—*e.g.*, should a previously occupied field become vacant. We discuss this in further work.

With these adaptations and the rules in Fragment 4.1 we were able to formally verify properties of the form

$$\mathcal{B}\ concern(\phi) \rightarrow \mathcal{B}\ other\_choices\_violate(\phi_1, \ldots, \phi_n)$$

where $\phi_1, \ldots, \phi_n$ are the set of ethical rules in $W(\phi)$.

This work on model checking ethical choices is preliminary. It is undesirable to have constructs, such as beliefs and belief rules, which can potentially affect program execution, used for verification purposes alone. However adapting AJPF with a more expressive property specification language was outside the scope of this research. The issue of how the approach scales remains open. The work here does demonstrate that an ethical policy can be incorporated within a BDI agent in such a way that adherence to the policy can be *formally* verified and so we can be *certain* the agent will always make the most ethical choices.

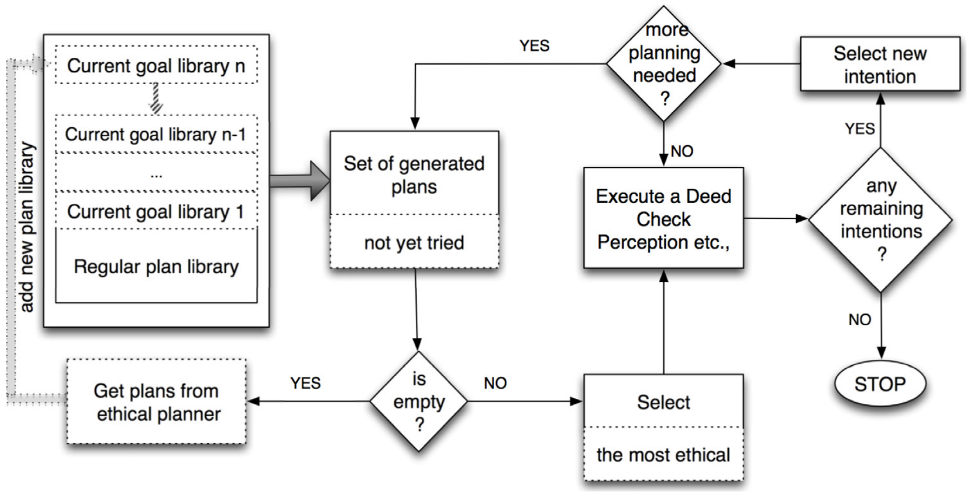


Fig. 2. ETHAN's reasoning cycle. Dotted lines show additions to GWENDOLEN's cycle.

Table 1
Additional atomic beliefs introduced for verification.

Belief	Meaning
Applicable(P)	Plan *P* is applicable
Already_tried(P)	Plan *P* has already been tried when attempting the current goal
Ethics_of(P, E)	Plan *P* violates ethical rule *E*
Concern(E)	The current selected plan violates *E*

5. Scenarios

We examined three ethical aviation scenarios for unmanned aircraft derived from discussions with domain experts: a retired Royal Air Force fast-jet navigator and a current UK private pilot licence holder.

We created an ETHAN program for each scenario and then verified that program. It should be noted that model checking is, inherently, a technique based on an idea of exhaustive testing and, in the case where a system interacts with the real world, it is necessary to supply a computational abstraction of the world as a model. In many cases it is also necessary to provide a model of the program to be verified, rather than the program itself but in the case of AJPF this is not necessary though we do have to provide a model of the planner. In each of our scenarios we have constructed our models of the planner and the real world in a slightly different way in order to demonstrate the different ways verification may be used to establish facts about the system.

In each of our scenarios we first tested our agent in one specific situation and then verified it in a more general model.

5.1. Ethical principles for civilian UAs

We assume that the UA agent operates only in civilian contexts. We establish a (small) list of relevant *formal* ethical concerns as examples in order to show the method in action. The list contains: *do not harm people* (f_1), *do not harm animals* (f_2), *do not damage self* (f_3), and *do not damage property* (f_4). The (formal) ethical policy is given by comparing the concerns in terms of how unethical it is to violate them. We propose the order $f_4 > f_3 > f_2 > f_1$, with $f_i > f_j$ meaning that it is more ethical to violate f_i than f_j. Our substantive ethical policies were context-dependent refinements of the formal ethical policy.

In our prototype, each flight phase (*e.g.*, landing, taxiing, take-off) of a UA constitutes one context c. Since all contexts are known, and the UA can only be in one context at a time, the substantive concerns can be represented directly, omitting the formal-substantive relations.

5.2. Brake failure during line up

In this scenario we examine a program for a UA to line up on a runway prior to take-off which includes plans for reacting to brake failure. We tested this in a simple simulation: Ahead of the aircraft is a second manned aircraft crossing the runway on a taxiway. To the left and right of the runway are runway lights (which can be damaged by aircraft taxiing over them). To the right of the runway is an airport staff member who has erroneously moved onto the manoeuvring area of the aerodrome.

The ethical concerns for this example, with the rank of each concern marked in parentheses, are:

ϕ_1 = do not damage own aircraft (1),
ϕ_2 = do not collide with airport hardware (2),
ϕ_3 = do not collide with people (3),
ϕ_4 = do not collide with manned aircraft (4).

When the agent determines that its brakes have failed it requests new routes from the ethical planner since its current route to line-up is no longer valid. The ethical planner quickly produces three potential routes:

1. Turn left off the runway: this will risk damaging the unmanned aircraft (ϕ_1) and colliding with airport hardware (the runway lights, ϕ_2).
2. Turn right off the runway: this will risk damaging the unmanned aircraft (ϕ_1) and a collision with people (ϕ_3).
3. Continue straight on: this will risk a collision with a manned aircraft(ϕ_4).

Code Fragment 5.1 shows abridged ETHAN code for this example. We use many syntactic conventions from BDI agent languages: `+!g` indicates the addition of a goal, g; `+b` indicates the addition of a belief, b; and `-b` indicates the removal of a belief. Plans consist of three parts, with the pattern

```
trigger : guard ← body.
```

The "`trigger`" is typically the addition of a goal or a belief (beliefs may be acquired thanks to the operation of perception and as a result of internal deliberation); the "`guard`" states conditions about the agent (in this example its beliefs) which must be true before the plan can become active; and the "`body`" is a stack of "deeds" the agent performs in order to execute the plan. These deeds typically involve the addition and deletion of goals and beliefs as well as *actions* (*e.g.*, `plan(regularRoutes,`

Code fragment 5.1 Code for Example 1

```
:Ethical Policy:
E(flightPhase(lineup),doNotDamageOwnAircraft,4)
E(flightPhase(lineup),doNotCollideAirportHardware,3)
E(flightPhase(lineup),doNotCollidePeople,2)
E(flightPhase(lineup),doNotCollideMannedAircraft,1)

: Initial Beliefs:
flightPhase(lineup)

:Reasoning Rules:

all_well :— ∼ brakesCompleteFailure;

: Initial Goals:
startup

:Plans:
+!startup : {⊤} ← +!missionComplete;
+!missionComplete : {B flightPhase(lineup), ∼B polled(veh) } ← +polled(veh), poll(veh);
+!missionComplete : {B polled(veh), B all_well, ∼B route(R)} ← plan(regularRoutes,all_well);
+!missionComplete : {B polled(veh), B all_well, B route(R)} ← enactRoute(R);
```

Code fragment 5.2 Plans for Example 1

```
+!missionComplete : {B brakesCompleteFailure} ← enactRoute(turn_left); [φ1, φ2]
+!missionComplete : {B brakesCompleteFailure} ← enactRoute(turn_right); [φ1, φ3]
+!missionComplete : {B brakesCompleteFailure} ← enactRoute(continue); [φ4]
```

Code fragment 5.3

```
+!missionComplete : {B brakesCompleteFailure}
    ← enactRoutewEffects(doNotDamageOwnAircraf, doNotCollideMannedAircraft); [φ1, φ4]
```

`all_well)`) which indicate code that is delegated to non-rational parts of the systems (in this case, the route planning system).

In Fragment 5.1, during normal operation, the agent polls the vehicle's sensors and, if all is well, it requests that the planner supplies routes for a normal take-off. The planner does this by sending predicates naming the routes to the agent which detects them via perception. Once the agent has a route (line 20) it then delegates the actual following of the route to the underlying control system (`enactRoute(R)`). If the brakes fail after the vehicle's sensors are polled, all these plans become unavailable (since `all_well` ceases to be true). In this case the "external planner" returns a set of routes as plans shown in Code Fragment 5.2. We use the notation $[\phi_{i_1}, \phi_{i_2}, \ldots, \phi_{i_n}]$ to indicate the substantive ethical concerns that are violated by each plan. On receiving these plans, and assessing the ethical policy, the agent elects to turn left.

5.2.1. *Formal verification of brake failure on line-up example*

Following on from ideas in [21,24] we replaced the ethical planner we used in the case study with a model that contained a random component. This model assumed that plans could potentially be available that violated any combination of the ethical concerns in the policy. An example of such a plan is given in Code fragment 5.3.

This "random" planner then selected a random subset of these plans and returned them to the agent. This meant we were no longer "testing" the agent in the simple simulation where we assumed the existence of aircraft and airport infrastructure in specific places in relation to the agent, but instead in a random environment where aircraft or infrastructure could potentially appear on any of the alternative routes.

When executed in combination with a model-checking algorithm the random choice caused the search space to branch and so the model-checking examined every possible set of plans that might be returned by the ethical planner. This allowed us to show that the most ethical plan was always chosen no matter what set of plans were available.

In particular we formally verified the following properties, where the ϕ_i formulae refer to the substantive ethical concerns used in the example. (Here '$\Box$' means "always in the future" and '$\mathcal{B}$' means "agent believes".)

$$\Box(\mathcal{B}\ \mathit{concern}(\phi_1) \rightarrow \mathcal{B}\ \mathit{other\_choices\_violate}([\phi_1, \phi_2, \phi_3, \phi_4]))$$
$$\Box(\mathcal{B}\ \mathit{concern}(\phi_2) \rightarrow \mathcal{B}\ \mathit{other\_choices\_violate}([\phi_2, \phi_3, \phi_4]))$$
$$\Box(\mathcal{B}\ \mathit{concern}(\phi_3) \rightarrow \mathcal{B}\ \mathit{other\_choices\_violate}([\phi_3, \phi_4]))$$
$$\Box(\mathcal{B}\ \mathit{concern}(\phi_4) \rightarrow \mathcal{B}\ \mathit{other\_choices\_violate}([\phi_4])).$$

Collectively these properties show that if the plan chosen violates some substantive ethical concern, ϕ, then the other available plan choices all violated some concern that was equal to, or more severe than, ϕ. Further similar properties can be used to establish that the "most ethical" option is always chosen.

In effect this verification demonstrates that, on this example at least, the underlying implementation of ethical choice was correct. It took our system over 21 h to verify each of 65,534 combinations of the 15 possible plans giving a total verification time of nearly four days for the four properties.

Code fragment 5.4 Plans for Example 2

```
+!avoid_collision : {B flightPhase(eAvoid)} ← enactRoute(turn_left); [φ1]
+!avoid_collision : {B flightPhase(eAvoid)} ← enactRoute(emergency_land); [φ2,φ3,φ4]
+!avoid_collision : {B flightPhase(eAvoid)} ← enactRoute(return_to_base); [φ4]
```

Code fragment 5.5 Code for Example 2

```
:Ethical Policy:
E(flightPhase(eAvoid), doNotViolateRoATurnRight, 2)
E(flightPhase(eAvoid), doNotViolateRoA500Feet, 2)
E(flightPhase(eAvoid), doNotCollideObjects, 3)
E(flightPhase(eAvoid), doNotCollideAircraft, 4)

:Reasoning Rules:
avoid_collision :— ∼das(intruder, headOn);

:Plans:
+!avoid_collision : {B flightPhase(eAvoid), ∼B route(eAvoid, Route)} ←
     plan(reqEmergRoute,turnRight), ∗route(eAvoid, R), enactRoute(R), wait;

 +das(intruder, headOn) : {B flightPhase(cruise)} ←
   −flightPhase(cruise), +flightPhase(eAvoid), +! avoid_collision;

−das(intruder, headOn) : {B flightPhase(eAvoid)} ← −flightPhase(eAvoid), +flightPhase(cruise);
```

5.3. Erratic intruder aircraft

This example is based on a program for avoiding other aircraft in accordance with the Rules of the Air. This program takes into account the possibility that some other aircraft, possibly a malicious intruder, but potentially also some ill-trained new pilot, appears on a collision course with the UA and fails to take the anticipated evasive actions.

In the environment we used for testing, the UA is cruising through civil airspace when it encounters an intruder aircraft approaching head on. Here the ROA (Rules of the Air) say that the UA should turn right, so the agent requests a route for turning right. However, this plan fails and the detect/avoid sensor (DAS) continues to indicate that the intruder aircraft is approaching. At this point the agent knows that it has already tried to turn to the right in order to avoid the intruder. Since the intruder is still approaching its first plan has failed. The agent has no more routes (or ETHAN plans) that apply since its only plans obey the ROA and would cause the agent to turn right again. At this point the ethical planner is invoked. The relevant substantive ethical concerns and their ranks are as follows:

$\phi_1 =$ do not violate turn right rule (2);
$\phi_2 =$ do not stay above 500 feet rule (2);
$\phi_3 =$ do not collide with objects on the ground (3);
$\phi_4 =$ do not collide with aircraft (4).

The planner returns the plans shown in Code Fragment 5.4. The agent initially chooses to turn left. In our test environment the oncoming aircraft once again matches the course change and so the agent then chooses to return to base.

An abridged version of the code for this example is shown in Code Fragment 5.5. Here, `*route(eAvoid, turnRight)` causes the intention to suspend execution until the agent believes it has a route for turning right. The action `wait` suspends the intention for a set time to allow the effects of actions to manifest.

Lines 14–15 are triggered when information arrives from the DAS that there is an intruder. As a result the flight phase changes from `cruise` to `eAvoid` and a new goal is set up to avoid a collision. The existing, ROA-compliant, plan for this goal is to get a route for turning right, enact that route and wait a short period to see if a collision will now be avoided. If the plan succeeds the belief that there is an intruder will vanish, the flight phase can be changed back to cruise, and the goal will be achieved since the agent now believes a collision has been avoided (see the belief rule in line 8).

When the existing plan fails, the plans in Fragment 5.4 are added to the agent's plan library. The first of these (`turn_left`) is attempted first. This also fails and the agent then attempts the third plan (`return_to_base`), which succeeds.

5.3.1. Formal verification of the erratic intruder example

For this example we chose to leave the planner unchanged, so the ethical outcomes of turning left, emergency landing and returning to base remain fixed, but instead had random behaviour from the oncoming *Erratic Intruder*. Essentially we are examining the success and/or failure of all the possible plans in the scenario. So rather than verifying every possible ethical annotation on a plan, we verified that the system continued to reason ethically as the external situation changed. Obviously it would have been possible to combine random behaviour on the part of the intruder, with random annotations on the available plans to increase the scope of the verification result.

As in the previous example, we verified the following properties, where the ϕ_i formulae refer to the substantive ethical concerns used in Example 2. (Here '$\square$' means "always in the future" and '$\mathcal{B}$' means "agent believes".)

$$\square(\mathcal{B}\ concern(\phi_1) \rightarrow \mathcal{B}\ other\_choices\_violate([\phi_1, \phi_2, \phi_3, \phi_4]))$$
$$\square(\mathcal{B}\ concern(\phi_2) \rightarrow \mathcal{B}\ other\_choices\_violate([\phi_1, \phi_2, \phi_3, \phi_4]))$$
$$\square(\mathcal{B}\ concern(\phi_3) \rightarrow \mathcal{B}\ other\_choices\_violate([\phi_3, \phi_4]))$$
$$\square(\mathcal{B}\ concern(\phi_4) \rightarrow \mathcal{B}\ other\_choices\_violate([\phi_4])).$$

Collectively these properties show that if the plan chosen violates some substantive ethical concern, ϕ, then the other available plan choices all violated some concern that was equal to, or more severe than, ϕ.

The verification of each property took between 21 and 25 s and explored 54 model states on 3.06 GHz iMac with 4 GB of memory.

5.4. Fuel low

Our final program was one for handling "fuel low" alerts from the Fuel subsystem causing the UA to attempt to land. In our test environment, if the agent cannot locate a safe landing site the ethical planner is invoked and returns three options (shown with ethical concerns violated and their ranks):

1. Land in field with overhead power lines. Violates: do not cause damage to critical infrastructure (4); do not collide with objects on ground (3); 500 feet low-flying ROA (2); do not damage own aircraft (1).
2. Land in field with people. Violates: do not collide with people (5); 500 feet low-flying ROA (2).
3. Land on an empty public road. Violates: do not cause damage to critical infrastructure (4); 500 feet low-flying ROA (2).

The agent then chooses the most ethical – the third plan – although both the first and third plans violate an ethical concern of severity 4, the first plan also violates a concern of severity 3 while the third plan does not.

5.4.1. Formal verification of the fuel low example

For the verification of this example we chose not to examine all possible plans that could be returned, based on their ethical outcomes, but instead defined four possible plans (the three listed above and landing in an empty field which had no adverse ethical outcomes) and returned some non-empty subset of these plans at random. Effectively, in this case, the assumption in our model is that there are only ever four possible options: empty fields, fields with power lines, fields with people and public roads (and that the ethical concerns involved with each choice are fixed). What may vary is which of these four options are available at any given time. Depending on which plans were available we also allowed the agent to perceive the presence of the empty field, road, etc. The model-checking then explored all possible combinations of these plans that could be returned and we verified that the UA would always chose the most ethical of the possible landing sites. In this case, instead of verifying that the most ethical choice was always made at an abstract level we verified based on the specifics of the options, *i.e.*, we examined three properties:

$$\mathcal{P}\ land\_on\_road \rightarrow \sim \mathcal{B}\ empty\_field$$
$$\mathcal{P}\ land\_in\_field\_w\_power\_lines \rightarrow \sim \mathcal{B}\ empty\_field \wedge \sim \mathcal{B}\ road$$
$$\mathcal{P}\ land\_in\_field\_w\_people \rightarrow \sim \mathcal{B}\ empty\_field \wedge \sim \mathcal{B}\ road \wedge \sim \mathcal{B}\ power\_lines$$

where $\mathcal{P}\phi$ indicates that ϕ holds true in the environment (*i.e.*, it is potentially *perceptible*).

The verification of each property took between 7 and 10 s and explored 64 model states on 3.06 GHz iMac with 4 GB of memory.

5.5. Remark on the examples

Our example programs are really only fragments of some larger program for control of a UA and in each case we have chosen to verify only a single property, demonstrating different ways models of the behaviour of the real world and the planning system can be created in order to allow verification. Obviously a full formal verification of an ethical UA would want to examine the full program, verify against several properties and use the model most appropriate to the full system—*i.e.*, a model based on the construction of the planner, ethical annotation system, and a detailed understanding of the operational environment (all aspects outside the scope of this paper). Our aim has not been to present a verified ethical UA but to demonstrate how our system for reasoning about ethical concerns, can be combined with an existing system in order to verify properties relating to the ethical operation of an autonomous system.

6. Summary and future work

Before an autonomous system is allowed to operate in a shared environment with people or other autonomous systems, sufficient assurances have to be provided that it will always behave within acceptable legal, ethical, and social boundaries. We propose a method for, and have implemented a working prototype of, an ethical extension to a rational agent governing an unmanned aircraft (UA). The agent can be provided with a particular ethical policy it uses to distinguish among possible plans and to select the most ethical plan for execution. We are able to *prove* formally that the prototype *only* performs an unethical action if the rest of the actions available to it are even less ethical.

Obviously there are limitations to what formal verification can tell us, particularly since many simplifications are involved. In our case we assume that the plans the agent receives have been correctly annotated with ethical consequences. Integration with assertion-based simulation and hardware-based testing can help in defining the limitations of formal verification and in developing a truly reliable system. A methodology for an integrated approach is currently being developed [52] and we would be interested in developing a fuller set of examples for our system and investigating them in such a framework.

The ethically enhanced agent is autonomous in the choice of actions, but not in the choice of ethical concerns and policies it will follow. These are constructed externally, but nonetheless agent-specific and can be private to the agent. The implemented agent follows only one ethical policy at any decision-making moment, because we assumed it can be in only one context at a time. We also assumed that all the contexts are known to the system designer. Our theoretical framework however is more general and does not involve these assumption.

The ethical governor of Arkin et al. [13], and similar solutions to ensure ethical behaviour [15] "transforms" ethical machine behaviour into a constraint satisfaction problem, namely find a plan that is ethical. A list of precise ethical constraints is compiled, purified of ambiguities and inconsistencies, stopping the autonomous system from performing any activity that violates these constraints. This approach is simple, elegant, but foremost does away with all the complications of ethical reasoning. We consider situations in which no ethical course of action is possible, which is an issue not addressed in [13,15]. Unfortunately, the moment we consider that an unethical action has to occur, the necessity for some form of ethical reasoning creeps in. We focus on establishing a minimal system for ethical decision-making that "transform" ethical machine behaviour into a *weak* constraint satisfaction problem.

The main contribution of our paper is a verifiable ethical decision-making framework that implements a specified ethical decision policy. In our approach we do not develop our own planner or method for generating plans. Rather we assume annotated plans are supplied to the agent. For BDI agents that have access to such a planner we construct a method for selecting among unethical plans, when no ethical plan is available. Our method is verifiable, namely we can prove that if an agent chooses to execute a plan that is in any way unethical, it does so only when it believes that this is the minimally unethical course of action it has available. This way the ethical decision-making is done by the agent and one is able to use agent verification techniques to prove correct behaviour.

An alternative approach for engineering ethical behaviour of an agent is to develop a planner that only develops a plan that is ethical, *i.e.*, "push" the ethical decision-making on the side of the planner. There are two immediate problems with this approach that we avoid in ours. First, the possible ethical consequences of a plan cannot be known until the plan is developed. Although the goal of the plan may in itself be ethical, the means to reach it might not. Second, a planner is a tool that can be used by several agents. When the ethical decision-making is on the planner side, all the agents that use the same planner would be subject to the same ethical policy which may or may not be known to the agent. This infringes on the autonomy of the agent to a certain extent.

The ethically enhanced agent we described follows only one ethical policy at any decision-making moment, the agent can only be in one context at a time and all the contexts are known (the contexts being the flight phases of the UA). The advantage yielded by this approach is in removing the need for additional calculation within the agent to relate specific plan outcomes to ethical principles via ethical rules. However, it cannot always be the case that the contexts in which an autonomous system operates are predicted or non-overlapping. We would like to extend our implementation to consider the full framework of ethical reasoning as outlined in Section 3.2.

Our implemented agent is also limited to only attempting a plan once for any given goal. In a highly dynamic environment, where the predicted ethical outcomes of plans might be rapidly changing, this is clearly unsatisfactory and we would be interested in extending the system so that plans could be re-evaluated if their ethical outcomes had changed, while making sure that the system must, at some point, commit to some plan.

Our general theory assumes that ethical rules can be expressed in terms of $do(a) \Rightarrow_c \neg E\varphi$ where a is an action involved in some plan p. However in our examples we already see that the relevant actions (*e.g.*, "collide with airport hardware") may not be explicitly referenced by a plan (*e.g.*, "turn right"). More work is therefore needed to understand the nature of actions that are implicit or side effects of some plan in order to better enable the calculation of ethical consequences either within agents or planning systems. We would also like to extend our system so that it could account for uncertainty in the evaluation of ethical outcomes.

There are two ways in which a context can influence the ethical constraining of an agent: (a) by making abstract principles substantive; and (b) by indicating which ethical policy should be used. So far, we use one ethical policy and contexts influence ethical reasoning by specifying what counts as an ethical violation in them. However, it is not difficult to envision situations in which the agent would need to use a different policy. Consider for example, the principles of doing no harm and autonomy (in medicine). During a minor surgery, the patient might prefer a local anaesthetic over a total one, however if the surgery turns out to be more complicated than predicted, the patient's wishes for local anaesthetic will be disregarded, violating the autonomy principle, in the interest of not harming him/her, preserving the no-harm principle. However, a patient might express the desire to not be resuscitated if his/her heart stops. Under these special circumstances (heart failure), observing the principle of autonomy should be placed as more ethical in the policy than the principle of doing no harm.

There are many ways in which a policy can be *updated* with respect to a special context; we mention two. The most simple way would be to provide a replacement policy. This method is only possible if the context is predetermined, and it also represents significant effort, since all principles have to be re-ordered. Another method is to only alter a part of the policy concerning specific principles. For this method, update procedures need to be developed.

One way to develop policy update procedures is to consider them as a special case of *belief update* and use a belief update operators, see for example [53] for such operators. Clearly, whether the requirements for belief update operators are fully suited for operators used to update ethical policies is an issue that merits further exploration. A further hindrance to using belief update to update ethical policies could arise from research in belief update being normative, studying which properties a belief update operator should satisfy, rather than operational, *i.e.*, designing belief update operators.

Finally, while our examples have been from the UA domain the approach and principles are general enough to be relevant across autonomous systems. Consequently, we aim to extend this work to the formal verification of domestic/healthcare robotics and driverless cars in the future.

Acknowledgements

Work was partially funded by EPSRC through the "Trustworthy Robotic Assistants" (EP/K006193/1), "Verifiable Autonomy" (EP/L024845/1), and "Reconfigurable Autonomy" (EP/J011770/1) projects, and by the ERDF/NWDA-funded Virtual Engineering Centre.

References

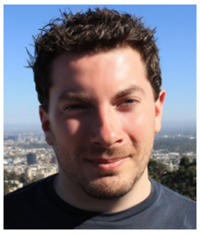
[1] M. Anderson, S. Anderson, Machine ethics: Creating an ethical intelligent agent, AI Mag. 28 (4) (2007) 15–26.
[2] B. Deng, Machine ethics: The robot's dilemma, Nature 523 (7558) (2015) 24–26.
[3] N. Goodall, Ethical decision making during automated vehicle crashes, Transp. Res. Rec.: J. Transp. Res. Board 2424 (2014) 58–65.
[4] S. Hirose, A code of conduct for robots coexisting with human beings, Robot. Auton. Syst. 18 (1–2) (1996) 101–107.
[5] J.H. Moor, The nature, importance, and difficulty of machine ethics, IEEE Intell. Syst. 21 (4) (2006) 18–21.
[6] S. Anderson, M. Anderson, A prima facie duty approach to machine ethics and its application to elder care, in: Human-Robot Interaction in Elder Care, 2011.
[7] M. Anderson, S. Anderson, C. Armen, An approach to computing ethics, IEEE Intell. Syst. 21 (4) (2006) 56–63.
[8] M. Anderson, S. Anderson, ETHEL: Toward a principled ethical eldercare system, in: AAAI Fall Symposium: AI in Eldercare: New Solutions to Old Problems, Vol. FS-08-02 of AAAI Technical Report, AAAI, 2008, pp. 4–11.
[9] B. McLaren, Extensionally defining principles and cases in ethics: An AI model, Artificial Intelligence 150 (1–2) (2003) 145–181. AI and Law.
[10] B. McLaren, Computational models of ethical reasoning: Challenges, initial steps, and future directions, IEEE Intell. Syst. 21 (4) (2006) 29–37.
[11] R. Robbins, W. Wallace, Decision support for ethical problem solving: A multi-agent approach, Decis. Support Syst. 43 (4) (2007) 1571–1587. Special Issue on Clusters.
[12] C. Allen, I. Smit, W. Wallach, Artificial morality: Top-down, bottom-up, and hybrid approaches, Eth. Inf. Technol. 7 (3) (2005) 149–155.
[13] R. Arkin, P. Ulam, B. Duncan, An Ethical Governor for Constraining Lethal Action in an Autonomous System, Tech. Rep. GIT-GVU-09-02, Mobile Robot Laboratory, College of Computing, Georgia Institute of Technology, 2009.
[14] R. Arkin, P. Ulam, A. Wagner, Moral decision making in autonomous systems: Enforcement, moral emotions, dignity, trust, and deception, Proc. IEEE 100 (3) (2012) 571–589.
[15] D. Brutzman, D. Davis, G. Lucas, R. McGhee, Run-time ethics checking for autonoous unmanned vehicles: Developing a practical approach, in: Proceedings of the 18th International Symposium on Unmanned Untethered Submersible Technology, 2013. URL http://hdl.handle.net/10945/37443.
[16] W. Damm, S. Disch, H. Hungar, S. Jacobs, J. Pang, F. Pigorsch, C. Scholl, U. Waldmann, B. Wirtz, Exact state set representations in the verification of linear hybrid systems with large discrete state space, in: Proc. Automated Technology for Verification and Analysis, in: LNCS, vol. 4762, Springer, 2007, pp. 425–440.
[17] L.A. Dennis, M. Fisher, N. Lincoln, A. Lisitsa, S.M. Veres, Reducing code complexity in hybrid control systems, in: Proc. 10th Int. Symposium on Artificial Intelligence, Robotics and Automation in Space, i-Sairas, 2010.
[18] 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: Proceedings of IFAC Workshop on Adaptation and Learning in Control and Signal Processing, 2010.
[19] M. Webster, M. Fisher, N. Cameron, M. Jump, Formal methods and the certification of autonomous unmanned aircraft systems, in: Proceedings of the 30th International Conference on Computer Safety, Reliability and Security, in: Lecture Notes in Computer Science, vol. 6894, Springer, 2011, pp. 228–242.
[20] M. Webster, N. Cameron, M. Jump, M. Fisher, Generating certification evidence for autonomous unmanned aircraft using model checking and simulation, J. Aerosp. Inf. Syst. 11 (5) (2014) 258–279.
[21] M. Fisher, L.A. Dennis, M. Webster, Verifying autonomous systems, ACM Commun. 56 (9) (2013) 84–93.
[22] E.M. Clarke, B.H. Schlingloff, Model checking, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier, 2001, pp. 1635–1790.
[23] C.A. Authority, CAP 393 Air Navigation: The Order and the Regulations, 2010. http://www.caa.co.uk/docs/33/CAP393.pdf.

[24] L. Dennis, M. Fisher, N.K. Lincoln, A. Lisitsa, S. Veres, Practical verification of decision-making in agent-based autonomous systems, Autom. Softw. Eng. (2014) 1–55.
[25] A. Rao, M. Georgeff, BDI agents: from theory to practice, in: Proc. 1st International Conference on Multi-Agent Systems, ICMAS, San Francisco, USA, 1995, pp. 312–319.
[26] R. Bordini, M. Dastani, J. Dix, A. El Fallah-Seghrouchni (Eds.), Multi-Agent Programming: Languages, Platforms and Applications, Springer, 2005.
[27] T.A. Henzinger, The theory of hybrid automata, in: Proc. 11th Annual IEEE Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, 1996, pp. 278–292.
[28] 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, Theoret. Comput. Sci. 138 (1) (1995) 3–34.
[29] A. Platzer, Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics, Springer, Heidelberg, 2010.
[30] L.A. Dennis, M. Fisher, M. Webster, R.H. Bordini, Model checking agent programming languages, Autom. Softw. Eng. 19 (1) (2012) 5–63.
[31] H.S. Richardson, Specifying norms as a way to resolve concrete ethical problems, Philos. Publ. Aff. 19 (4) (1990) 279–310.
[32] T.L. Beauchamp, J.F. Childress, Principles of Biomedical Ethics, sixth ed., Oxford University Press, 2009.
[33] E. Sacerdoti, Planning in a heirarchy of abstraction spaces, Artificial Intelligence 5 (1974) 115–135.
[34] M. Helmert, The fast downward planning system, J. Artificial Intelligence Res. 26 (2006) 191–246.
[35] A.J. Coles, A.I. Coles, M. Fox, D. Long, Forward-chaining partial-order planning, in: Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS-10, 2010. URL http://www.aaai.org/ocs/index.php/ICAPS/ICAPS10/paper/view/1421/1527.
[36] K. Tulum, U. Durak, S. Yder, Situation aware UAV mission route planning, in: 2009 IEEE Aerospace Conference, 2009, pp. 1–12.
[37] A.F.T. Winfield, C. Blum, W. Liu, Towards and ethical robot: Internal models, consequences and ethical action selection, in: M. Mistry, A. Leonardis, M. Witkowski, C. Melhuish (Eds.), Advances in Autonomous Robotics Systems, in: Lecture Notes in Computer Science, vol. 8717, Springer, 2014, pp. 85–96.
[38] J. Jeannin, K. Ghorbal, Y. Kouskoulas, R. Gardner, A. Schmidt, E. Zawadzki, A. Platzer, A formally verified hybrid system for the next-generation airborne collision avoidance system, in: C. Baier, C. Tinelli (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, in: Lecture Notes in Computer Science, vol. 9035, Springer, Berlin, Heidelberg, 2015, pp. 21–36.
[39] R.M. Veatch, Resolving conflicts among principles: Ranking, balancing, and specifying, Kennedy Inst. Ethics J. 5 (3) (1995) 199–218.
[40] D. Grossi, J.J.C. Meyer, F. Dignum, Modal logic investigations in the semantics of counts-as, in: Proceedings of the 10th International Conference on Artificial Intelligence and Law, ICAIL'05, ACM, New York, NY, USA, 2005, pp. 1–9.
[41] J.R. Searle, The Construction of Social Reality, Penguin, 1995.
[42] L. Dennis, M. Fisher, A. Hepple, Language constructs for multi-agent programming, in: F. Sadri, K. Satoh (Eds.), Computational Logic in Multi-Agent Systems, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 137–156.
[43] S. Visser, J. Thangarajah, J. Harland, Reasoning about preferences in intelligent agent systems, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence. URL http://www.aaai.org/ocs/index.php/IJCAI/IJCAI11/paper/view/3017.
[44] S. Sardiña, S. Shapiro, Rational action in agent programs with prioritized goals, in: Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, ACM, 2003, pp. 417–424.
[45] J. Baier, S. McIlraith, Planning with preferences, AI Mag. 29 (4) (2008) 25–36.
[46] R. Feldmann, G. Brewka, S. Wenzel, Planning with prioritized goals, in: Proceedings, Tenth International Conference on Principles of Knowledge Representation and Reasoning, Lake District of the United Kingdom, June 2–5, 2006, 2006, pp. 503–514.
[47] K. Vikhorev, N. Alechina, B. Logan, Agent programming with priorities and deadlines, in: The 10th International Conference on Autonomous Agents and Multiagent Systems—Vol. 1, AAMAS'11, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2011, pp. 397–404.
[48] P. McNamara, Deontic logic, in: E.N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, fall 2010 ed., 2010.
[49] R.M. Chisholm, Contrary-to-duty imperatives and deontic logic, Analysis 24 (2) (1963) 33–36.
[50] J. van Benthem, D. Grossi, F. Liu, Priority structures in deontic logic, Theoria 80 (2) (2014) 116–152.
[51] L.A. Dennis, B. Farwer, Gwendolen: A BDI language for verifiable agents, in: Proc. AISB Workshop on Logic and the Simulation of Interaction and Reasoning, AISB, 2008.
[52] M. Webster, C. Dixon, M. Fisher, Safe and trustworthy autonomous robotic assistants, Space Saf. Mag. 9 (2014) 7–10. URL http://www.spacesafetymagazine.com/wp-content/uploads/2013/12/Space-Safety-Magazine-Issue-9-Winter-2014.pdf.
[53] J. Delgrande, Y. Jin, F.J. Pelletier, Compositional belief update, J. Artificial Intelligence Res. 32 (1) (2008) 757–791.

Louise Dennis is a postdoctoral Research Associate in the Department of Computer Science at the University of Liverpool. She has published research in both agent programming languages and verification tools (both theorem provers and model checkers). At the moment her research is focused on verifying autonomous systems.

Michael Fisher is a Professor in the Department of Computer Science at the University of Liverpool. His research interests involve logical methods in Computer Science and Artificial Intelligence, particularly temporal reasoning, programming languages, formal verification, and the development and analysis of autonomous and agent-based systems.

Marija Slavkovik is a Postdoctoral Fellow at the University of Bergen in Norway. She obtained her Ph.D. in Computer Science at the University of Luxembourg in 2012. Her interests are normative reasoning, collective decision-making, and machine ethics, for multi-agent systems.


Matt Webster is a postdoctoral Research Associate in the Department of Computer Science at the University of Liverpool. His research interests include autonomous systems, formal methods and computer security. Working closely with industrial and academic collaborators, he has developed and formally verified a variety of autonomous systems for use in multi-agent systems, unmanned aircraft and robotics.