On Proactive, Transparent and Verifiable Ethical Reasoning for Robots

Paul Bremner, Louise A. Dennis, Michael Fisher and Alan F. Winfield

***Abstract*—Previous work on ethical machine reasoning has largely been theoretical, and where such systems have been implemented it has in general been only initial proofs of principle. Here we address the question of desirable attributes for such systems to improve their real world utility, and how controllers with these attributes might be implemented. We propose that ethically-critical machine reasoning should be proactive, transparent and verifiable. We describe an architecture where the ethical reasoning is handled by a separate layer, augmenting a typical layered control architecture, ethically moderating the robot actions. It makes use of a simulation-based internal model, and supports proactive, transparent and verifiable ethical reasoning. To do so the reasoning component of the ethical layer uses our Python based Beliefs, Desires, Intentions (BDI) implementation. The declarative logic structure of BDI facilitates both transparency, through logging of the reasoning cycle, and formal verification methods. To prove the principles of our approach we use a case study implementation to experimentally demonstrate its operation. Importantly, it is the first such robot controller where the ethical machine reasoning has been formally verified.**

I. Introduction

Robots are increasingly autonomous: semi-autonomous flying robots are commercially available, and driverless cars are undergoing real-world tests [1]. This trend is expected to continue [2]. Such systems have expanding abilities for making unsupervised decisions. This makes it imperative both that robotic systems are capable of taking human ethical values[1] into account when they make decisions, and that mechanisms are in place to guarantee that the behaviour executed by the robot respects those values. A particularly important value, and one at the forefront of many people's minds when considering robotic systems that will interact closely with humans, is the question "is it safe?" – highlighting the importance of mechanisms for guaranteeing safety [3], [4]. However, many other ethical considerations, such as human free will, privacy and dignity also come into play when considering autonomous robotics, particularly robots that may be operating in domestic or healthcare situations.

Safe robot behaviour is clearly essential, but not sufficient. Smart autonomous robots should be more than safe; they should also be *transparent*, at least where ethical reasoning is concerned –– able to both choose and justify [2] actions that relate to human values. Following Moor [5] our focus in this paper is on explicit ethical agents in which ethics are represented explicitly, and actions chosen on the basis of those ethics. As the cognitive, perceptual and motor capabilities of robots expand, they will be expected to have an improved capacity for moral judgment[2]. By transparent we mean that ethical machine reasoning should be accountable and accessible, such that human scrutiny of such decision processes is possible [6]. Making ethical machine reasoning scrutible in this way enables exposition of the reasoning behind actions taken, and facilitates trust in such systems. It is clear, through international efforts such as the developing IEEE P7001 standard on Transparency in Autonomous Systems, that this view is becoming mainstream.

We are used to the idea of *safety-critical* systems and that such systems require high standards of validation – preferably formal verification. Clearly, since safety is one of the ethical principles[3] relevant to robotic systems these will also require high standards of validation. More generally, since ethical principles are fundamental to the way a society judges both itself and others we would argue that *ethically-critical* systems (systems that have the capacity to impact adversely any ethical value considered important by the community it is created for) should in general be held to high standards of validation, just as safety-critical systems are. Therefore ethical machine reasoning should be *verifiable* [7].

Initial approaches to the implementation of ethical machine reasoning have focused on the idea of an ethical layer that can veto actions suggested by the underlying system [8]. However, as was notably observed in Asimov's construction of his three

The work described in this paper was funded by the EPSRC "Verifiable Autonomy" project (EP/L024845/1 (Liverpool) and EP/L024861/1 (UWE))

[1]We define human values here as broad preferences concerning appropriate courses of actions or outcomes, and ethical values as generally agreed right actions which, for instance, maintain safety and dignity.

[2]Note we are not claiming that a robot must be a full moral agent – capable of not only reasoning about ethics but also justifying its ethical choices [5] – only that its reasoning needs to take ethical considerations into account.

[3]We regard ethical principles as simply expressions of ethical values. So the principle 'do no harm' expresses the human value of respect for others' safety.

Laws of Robotics [9], ethical reasoning cannot simply be *reactive* (it is insufficient to state simply that a robot may not harm a human) but must also be *proactive* ('a robot should not, through inaction, allow a human to come to harm'). We suggest that in order to make proper ethical judgements any ethical reasoning component should be capable of generating and evaluating ethically proactive actions, not just evaluating those proposed by some underlying goal-directed system.

We therefore obtain three key desirable properties for ethical machine reasoning for robotic systems:

- It should be *proactive* and able to initiate action where inaction risks violating ethical principles.
- It should be *transparent* in order to allow direct inspection of its reasoning by stakeholders, and potentially also allow human-understandable explanations of its reasoning.
- It should be possible to *verify* that it respects the values it reasons about, particularly (though not exclusively) where safety is concerned.

Clearly ethical machine reasoning is possible without adhering to these properties. However, we argue that adherence improves the utility for an ethical machine operating in the real world. Moreover, while transparency and verifiability are of benefit to most computational systems (including robot controllers in general), we argue that they are of particular importance in ethically-critical systems where the dual importance of both respecting human ethical values and being seen to do so is key to acceptance.

To demonstrate the practicality and utility of our proposed key properties we have devised an extension of the ethical layer in the architecture proposed in [10]. The extension we have proposed consists of two main components. The first is the addition of a pro-active Planner Module to produce ethically pro-active plans as required. Second is the addition of an 'ethical black box' recorder module, that logs the operation of the ethical layer and allows for post-hoc scrutiny of ethical decisions. As the ethically-critical component of our proposed architecture, it is the ethical layer in which our desirable properties are implemented.

To better illustrate the architecture, we detail an implementation following the proposed model. To do so we have developed a Python library for *Beliefs, Desires, Intentions*-style agent reasoning called *BDIPython*. *BDIPython* allows us to explicitly express the ethical layer's ethical decision module using logical rules. Doing so facilitates transparency of the reasoning in the ethical layer. In this implementation we have used Asimov's Laws of Robotics as our code of ethics, chosen not because they represent a viable code of machine ethics, but because they are a well-known and straightforward set of ethical rules that can be used to illustrate our approach.

To demonstrate that our implementation fulfils the desired verifiable property, we detail a means of formal verification of the ethical decision module. We have linked *BDIPython* directly to the Agent JavaPathfinder (AJPF) model-checker by implementing a parser for the *BDIPython* code into a Java data structure that is used to generate a model. We can then verify that the *BDIPython* ethical decision module respects some given ethical system using an established methodology for the formal verification of autonomous systems supported by AJPF [11].

Finally we have validated our approach experimentally using a simple case study with real robots. Through a series of experiments we demonstrate that our implementation is capable of making demonstrably correct ethical decisions, and the reasoning process is transparent to human scrutiny. We then verify that this system obeys Asimov's three laws of robotics.

II. Background

In this section we elaborate upon the background motivation and design principles of the components comprising the proposed architecture and implementation thereof. Each subsection that follows relates to a particular element of the work presented in this paper. First, we consider the motivation behind ethical reasoning in robots; second, the simulation-based approach to robot anticipation, and how this can provide a robot with the capability to reason about ethical consequences; third, we detail the case for an 'ethical black box' (EBB) recorder; fourth, we describe the *Beliefs, Desires, Intentions* (BDI) paradigm, and why it is suitable for implementation of transparent ethical reasoning; finally, we discuss formal verification, and how we can apply such a methodology to our BDI ethical reasoning.

A. Ethical Robots

As robots are increasingly expected to operate in environments within which their actions have moral or ethical implications, especially those with other agents, they require the capability to reason about these implications. Indeed, the need for robots equipped with ethical reasoning capabilities has been recognised in much recent work (e.g., [5], [8], [12]–[14]). Studies have shown that people expect ethical decision making from robots, holding them to moral standards according to their appearance [15].

In order for a robot to reason about ethics it requires *inter-alia* a set of ethical rules. How

these rules should be selected is, in general, an open problem, and various authors have suggested multiple approaches (see [16] for a review). Two paradigms of human ethical reasoning underlie the main approaches to machine ethics thus far suggested: consequentialist and deontological [16]. The central premise of consequentialist ethics is that actions should be evaluated on the basis of their consequences. Asimov's laws of robotics [9] are a well known set of rules for governing machine ethics. However, their suitability has been justifiably questioned for a range of reasons, including the lack of any mechanism for resolving ethical dilemmas caused by intra-law conflicts, e.g., if two humans are in danger and only one can be saved [2]. Deontological ethics on the other hand, is concerned with evaluating the motivation behind actions, i.e., does the intended purpose of the selected action abide by a given code of ethics. However, such an ethical paradigm has difficulties for artificial morality since it requires the correct attribution of mind states to artificial agents [16].

Allen et al [16] defined two clear approaches to implementing machine ethics for either ethical paradigm: top-down, whereby ethical decisions are incorporated into a robot's control architecture; or bottom-up, where a learning system is used to emulate ethically acceptable behaviour without necessarily understanding the underlying ethics adhered to. Here we consider the top-down approach because of its suitability both for verification and human scrutability.

B. Anticipation in Robotics (for Proactive Ethics)

In order for a robot to reason about ethics to be based on consequentialism it is necessary for it to be able to anticipate the outcomes of its actions. More generally, for a robot designed to operate in the highly dynamic environment of the real world, the ability to 'anticipate' future events is a major advantage. Such predictions can allow a system to make decisions in a way that combines past, present and future events, so it is better equipped to react appropriately to unconstrained environments [17]. We follow Rosen's definition of an anticipatory system as "[...] a system containing a predictive model of itself and/or of its environment, which allows it to change state at an instant in accord with the model's predictions pertaining to a later instant." [18].

The conventional use of internal models, in which a system is mathematically modelled, is encompassed within Rosen's definition. However, even though such approaches have been extended to cover well-defined uncertainties and non-linear plant [19], their anticipatory capabilities are limited. Typcally the external environment is not explicitly modelled beyond *a-priori* defined exogenous disturbances to the system model.

A newer and more powerful way in which a robot can be equipped with the cognitive machinery to enable anticipation is through an embedded simulation of the robot, its environment and agents therein [20]. A robot so equipped, has the potential to generate and test *what-if* hypotheses: what if I carry out action x?; of several possible actions x_i which should I choose? One aim of such hypothesis testing is to enable assessment of the consequences of proposed actions without needing to commit to carrying out those actions [21]. That is to say, alternative sequences of motor actions are evaluated to find the sequence that best achieves the goal, before executing the *best* sequence. Identification of the *best* actions typically requires sufficient environment simulation to establish action context.

Arguably these simulation-based internal models of others are a step in the direction of an artificial theory of mind [22], i.e., the ability to form a predictive model of others [23]; indeed one of the several theories of mind is the simulation theory of mind [24].

Here we utilise simulation-based hypothesis testing as a fundamental component of our architecture. We do so to enable the ethical layer in our robot controller to assess the ethical consequences of potential behaviours; thus, it can select the most ethically appropriate. Central to the simulation we use are assumptions of how the human being modelled (in the robot's environment) will act – as a crude form of theory of mind.

C. Ethical Black-box Recorder (for Transparency)

In line with expectations of ethical decision making, there is also a requirement that such an ethical robot's control processes are transparent and understandable. There are a number of reasons for this: it enables trust if stakeholders can understand a robot's reasons and validate for themselves that those reasons align with their ethics; in accident investigation scenarios, transparency helps in diagnosing the causes of error, and in their subsequent correction [25].

Winfield and Jirotka argue for a robot equivalent of the flight data recorder or *black-box* installed in aircraft to facilitate this [25]. By recording sensor data, and the control processes that acted on this data, *post-hoc* scrutiny of robot behaviour becomes possible. While some modern vehicles with partial autonomy have a data recording device, companies have thus far constrained access to this data to their own experts. We suggest that to be of real value this data needs to be open to outside scrutiny, outweighing any liability fears. To this end we incorporate a data logging module into our proposed ethical layer. Following the proposal in [25], it records

the information needed for human scrutiny of the ethical decision process followed.

D. Beliefs-Desires-Intentions Programming (for Transparent Ethics)

However, we argue that records of sensor data and control processes are insufficient by themselves. For the key parts of ethical reasoning it must be possible for a human to understand *how* a deduction was generated from sensor data. More importantly this process of deduction needs to be explainable in a way that is similar to the way humans justify their own actions. At present it is often challenging to extract such explanations from utility functions in general, and virtually impossible with opaque control techniques such as artificial neural networks. We argue therefore that ethical reasoning should be represented in a declarative fashion as typified, for example, by the logic programming paradigm – for this we turn to ideas from the field of *rational agents*.

At its most general, an agent is an abstract concept that represents an autonomous computational entity that makes its own decisions [26]. A general agent is thus simply the encapsulation of some distributed computational component within a larger system. However, in settings such as ours, it is increasingly important for the agent to have explicit reasons (that it could explain, if necessary) for making one choice over another.

Beliefs, *Desires*, and *Intentions* (BDI) agent programming languages provide this capability. They are based on the concept of *rational agency* [27]–[30] and draw heavily from the logic programming paradigm. Crucially BDI agents make decisions based on intuitive concepts of how an agent's beliefs and desires lead to particular choices. These intuitive concepts were first elaborated by Bratman [27] and gave rise to a selection of *BDI logics* which were subsequently operationalised as BDI programming languages. In the BDI programming paradigm, beliefs represent the agent's (possibly incorrect) information about its environment, desires represent the agent's long-term goals, while intentions represent the goals 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 (e.g., AgentSpeak [31], *Jason* [32], 3APL [33], Jadex [34], *Brahms* [35], GOAL [36], and GWENDOLEN [37]). Agents programmed in these languages commonly contain a set of *beliefs*, a set of *goals* (representing desires and usually forming part of intentions), and a set of *rules*. Rules determine how an agent acts based on its beliefs and goals.

In general a rule is selected based on the beliefs and goals of the agent, and transforms the goal (desire) into an intention which is some executable structure that is supposed to achieve the goal. As a result of executing a rule/intention, the agent may perform actions and its beliefs and goals may change, often as a result of the actions. Crucially the reasoning that is performed over beliefs and goals is based on logic programming, and is often explicitly expressed in Prolog. This makes it possible to understand the choice of rules (and by extension actions), as deduction in first-order predicate logic, and to explain such choices in these terms.

At the core of most BDI programming languages lies a *reasoning cycle*. The details vary from language to language but key aspects are: polling an external environment for new perceptions represented as beliefs; reasoning over beliefs and goals to select appropriate rules; executing rules to perform actions or manipulate beliefs and goals. In general the execution of actions causes some effect in the external environment. This external environment may be a software artefact, or software that mediates between the agent and the real world. The explicit assumption of interaction with an external environment makes BDI programs particularly suitable for embedding as reasoning components in larger systems. Thus, as we desire the reasoning executed by our ethical layer to be human comprehensible, we implement the reasoning part of the ethical layer using BDI programming. To facilitate the integration of a BDI component within the ethical layer as a whole, we have developed a Python BDI implementation.

E. Verification of Agent-based Autonomous Systems (for Verifiable Ethics)

In addition to human scrutability, we also desire that the ethical reasoning be verifiable, i.e., can be proven to abide by a given code of ethics. A property to which a BDI implementation lends itself well. The approach used for formal verification is dependent on the control architecture. *Hybrid* control architectures such as ours, comprising discrete and continuous parts are becoming increasingly popular in the construction of autonomous systems. A typical hybrid system architecture is shown in Figure 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 [38].

Formal verification is essentially the process of assessing whether a specification given in formal logic is satisfied on a particular formal description of the system in question. For a specific logical property, φ, there are many different approaches to this [39]–[41], ranging from deductive verification against a logical description of the system ψ_S (i.e., $\vdash \psi_S \Rightarrow \varphi$) to the algorithmic verification of the

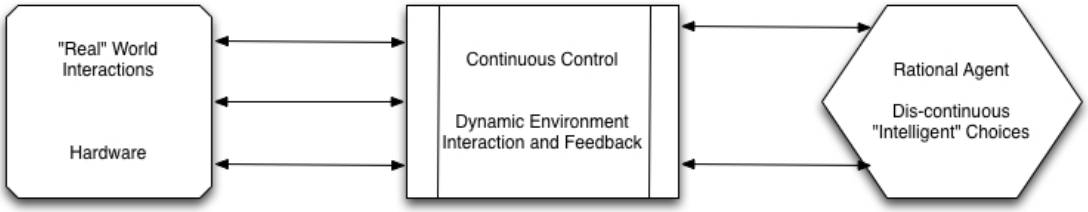

Figure 1: A Typical Hybrid Agent Architecture.

property against a model of the system, M (i.e., $M \models \varphi$). The latter has been extremely successful in Computer Science and Artificial Intelligence, primarily through the *model checking* approach [42]. This takes a model of the system in question, defining all the model's possible executions, and then checks a logical property against this model (and, hence, against all possible executions).

In a hybrid autonomous system the continuous control and the higher-order decision-making components can (ideally) be cleanly separated. The lower-level procedures generally also appear in non-autonomous systems, and well understood techniques for their validation exist. One approach, therefore, is to focus formal verification efforts on the decisions the rational agent makes, given the beliefs and goals it has [7], [43]. In any system that interacts with a highly unpredictable real world, we cannot show that an agent always does the right thing, but we can show *its actions are taken for the right reasons*.

This approach has been adopted as as methodology by a range of work considering applications in autonomous aircraft, spacecraft and road vehicles [11], [43]–[45], in which a rational agent is verified using model-checking to assess all its potential decisions. This methodology clearly adapts well if we can implement the Ethical Decision Module component of our supervisory ethical layer as a BDI agent, allowing us to verify that such a system always chooses options that align with a given code of ethics based on the information that it has.

III. A Simulation Based Ethical Reasoning Layer for Robot Control Architectures

Though robot controller architectures are many and varied (see [46] for a review), most can be considered to fit a three-layered model paradigm [46], given the hierarchical organisation of behaviours being controlled [47]. The layers in such a hierarchy are characterised by the degree of abstraction and temporal granularity at which they operate. The highest level control layer generates overall goals to be achieved (e.g., retrieve the beer from the fridge). At the next control layer the goals are broken down into a series of tasks that make up the goal (e.g., move to fridge, open door). At the lowest level tasks are broken down into motor actions that the robot must execute (e.g., move forward, raise arm)[4].

Although this generalization of robot controllers ignores defining features of individual architectures, it serves as a useful framing mechanism for the addition of ethical decisions via a specialised fourth layer. Arkin proposed that the purpose of an *ethical layer* is to moderate the outputs of the other layers by evaluating them against a set of ethical criteria [8], [48]. Ensuring ethical behaviour through a separate layer has a number of advantages. One key advantage is that the functionality can be verified independently of the rest of the robot controller [45].

However, an ethical layer that acts only as a moderator can only evaluate behaviours that are proposed as part of the normal controller's operation. As discussed previously, acting ethically might also involve being *proactive*, i.e. choosing a new action specifically to adhere to ethical principles. An example of such proactive behaviour is acting to prevent a human from coming to harm even if it might put the robot in danger (as per Asimov's laws).

A previous approach to addressing this issue was to have the controller suggest all possible tasks as behavioural alternatives, and not just those relating to system goals [10]. However, the possibility of modifying a controller to suggest alternatives in the whole behaviour space relies on there being a practicable number of behavioural alternatives. In [10] this approach was possible due to a highly constrained experimental setting where the environment could be discretised into a grid, and each grid square was considered as a possible behavioural choice. It seems reasonable to assume that in most settings this will not be possible (due to challenges in defining suitable discretisation, and computational cost of evaluating a large number of behaviours). Moreover, modifying the core robot

[4]Where a rational agent is used, as described in section II-E, then it generally combines the processes of goal selection and breaking the goal down into tasks.

controller in this way eschews the benefit of a separate layer as an independently testable system.

By having the ethical layer generate behaviours it can do so in a more constrained manner, given the ethical principles being utilised. Further, it only needs to do so when the behaviours suggested by the other control layers do not satisfy those ethical principles.

To this end we describe here an ethical layer that not only evaluates options suggested by the other control layers, but also generates and evaluates behavioural alternatives. These behavioural alternatives are generated as a consequence of ethical issues it anticipates will occur if only the behaviours suggested by the other layers are considered.

Where the ethical layer sits in the layer structure (and hence what we mean by behavioural alternatives) is dependent on the level of abstraction used at each layer, and hence whether the outputs from a given layer can be considered for their adherence to the adopted code of ethics. Here we suggest that it sits between the task and action layers as this is the level of abstraction that best fits our adopted code of ethics. Thus, tasks proposed by the task layer need to be tested before they are used to generate action sequences. The ethical layer takes as input the tasks the task layer suggests, and outputs the most ethically appropriate task to the action layer.

The ethical layer detailed here consists of four modules: a Simulation Module to predict the outcome of tasks; a Planner Module to generate alternative tasks if those proposed by the task layer have ethical issues; an Ethical Decision Module to compare the simulated actions and select the most ethically appropriate using declarative BDI reasoning, and an EBB data logging Module to record the situations encountered and decisions made. Data flow and module integration is shown in Figure 2, and detailed in its caption. This is an extended version of the architecture proposed in [49], but with a more sophisticated, and transparently ethical decision process.

In order to prove the principles of operation for the control architecture and verification process, and to provide a clearer system description, we consider here a simple demonstration use case. In this case a second robot is used as a proxy human (H-robot), operating with the ethical robot (E-robot) in a sparsely featured environment containing a few objects which can be defined as safe or dangerous (dependent on experimental settings, see Figure 4). This demonstration use case will be elaborated upon in the module descriptions, and in Section V. In this context if the proxy human is predicted to move towards a dangerous location, the Planner Module will suggest points at which the robot can intercept the human path as potential tasks to be evaluated.

In the demonstration use case we have treated Asimov's laws of robotics [9] as a test code of ethics, despite their obvious shortcomings, noted above and in [2], [50]. There is currently no agreement on the code of ethics a robot should adhere to, even in simple scenarios [2]. However, we require ethical rules against which a robot's behaviour can be evaluated, and hence demonstrate our architecture. In the context of the current paper, therefore, we choose to make use of Asimov's laws to demonstrate the efficacy of our approach to proactive, transparent and verifiably ethical robots without assigning any particular status to these laws.

Asimov's Laws of robotics [9] are the earliest and best known set of ethical rules proposed for governing robot behaviour. Despite originating in a work of fiction, Asimov's Laws explicitly govern the behaviour of robots and their interaction with humans. This contrasts with more traditional consequentialist ethical frameworks which would need adapting to this purpose. The laws are simply described as follows.

1 A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2 A robot must obey the orders given it by human beings, except where such orders would conflict with the First Law.
3 A robot must protect its existence as long as such protection does not conflict with the First or Second Laws.

We have supplemented Asimov's laws with additional rules that relate to the likely success of an intercept plan. These are given a lower priority than the 3rd law, so will only influence behaviour if no other law is violated.

4a If the human is far from danger prioritize waiting time at the intercept point.
4b If the human is close to danger prioritize shorter robot walking distance.

The rationale behind 4a is that the longer the wait time for the robot after arriving at the intercept location before the human arrives, the more robust the system is to errors in predicted paths and travel times. The rationale behind 4b is that walking incurs some risk of falling over. This risk increases the further the robot walks, so a shorter walking distance increases the chance of achieving the robot's goal.

A. *Simulation Module*

The Simulation Module predicts the outcomes of behavioural alternatives. In order to do so it needs to be equipped with:

1) a model of the robot controller;
2) a domain specific model of the human;

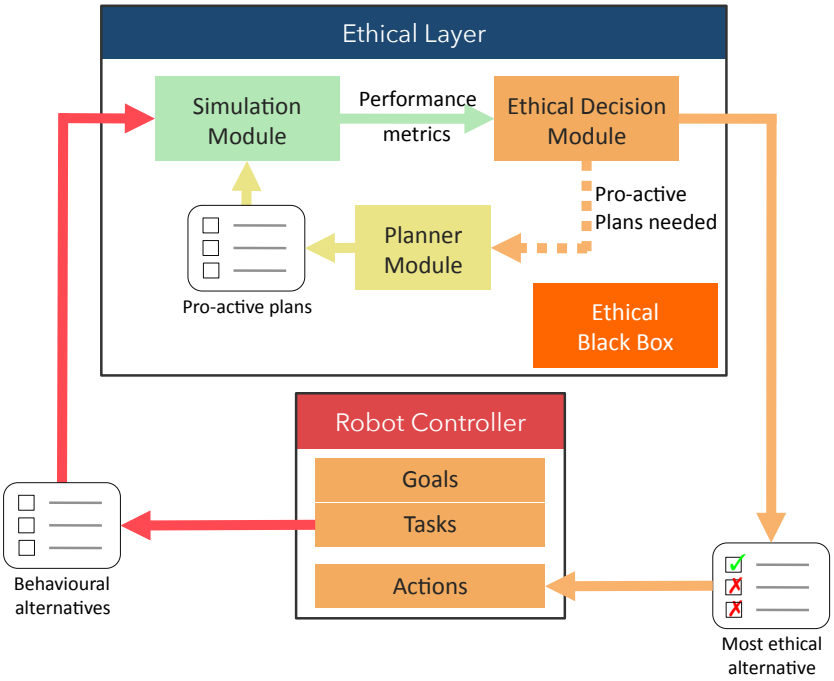

Figure 2: The architecture of our proposed Ethical Layer alongside the Robot Controller. The task layer in the Robot Controller generates a set of tasks that might fulfil the current goal proposed by the goal layer. Before generating the actions needed for task execution, the set of proposed tasks is sent to the Ethical Layer, along with sensor data for the current situation and the goal to be achieved. The Simulation Module simulates each of the tasks (behavioural alternatives), producing a set of evaluation metrics for each, that are sent to the Ethical Decision Module. The Ethical Decision Module checks its ethical criteria to see if ethically proactive tasks are required, and triggers the Planner Module if so. When triggered the Planner Module generates a set of pro-active tasks to send to the Simulation Module, and as with the controller generated tasks they are simulated. If the Planner Module was triggered the Ethical Decision Module adds the pro-active task simulation results to its current set. The Ethical Decision Module evaluates the set of simulated task alternatives to determine the most ethically appropriate, and sends this selection to the action layer for execution. A solid arrow is a flow of data, a dashed arrow is a control signal. The Ethical Black Box (EBB) Module logs data from each of the other modules (for clarity these data flows are not shown).

3) a model of the world; and
4) a set of metrics against which the simulated plans are measured.

As stated in [49] the models need only be of sufficient fidelity to handle the domain in which the robot is to operate. In the demonstration case presented here, a low fidelity simulation, modelling the motions of agents as ballistic trajectories, is adequate to prove the principles of operation.

In order to simulate its own behaviour the ethical robot uses three basic assumptions: (1) the robot uses a path planning algorithm that navigates the shortest path to its goal; (2) the robot has collision avoidance, so it plans a path to avoid known obstacles, and stops when 0.5m or closer to the human or a dangerous obstacle, and (3) valid goal locations are objects in the environment. It is also important to note that behavioural alternatives consist of target locations in the arena to be moved to.

In order to simulate the behaviour of the human the robot's model also incorporates four basic assumptions: (1) The human walks in a straight line to its goal. (2) The human goal is always an object in the environment. From these two assumptions the human goal can be inferred by projecting a line in its direction of travel, and the closest object to that line is assumed to be the goal. (3) The human is unaware of which objects are dangerous. And (4) the human has collision avoidance, so it plans a path to avoid known obstacles, and stops when 0.5m or closer to the robot.

The world model in this demonstration case consists only of static objects, designated as safe or dangerous (see Figure 4).

Using the goal estimated for the human robot, and the ethical robot's goal as suggested for a particular behavioural alternative, paths are simu-

lated for both agents. We also simulate the obstacle avoidance process running on both agents. Hence, if the paths of the two agents would come within 0.5m of one another they will stop.

The simulated paths are scored on a set of performance metrics which will be used in the Ethical Decision Module to evaluate the plans. The set of metrics chosen reflects the characteristics of a given simulation – some metrics will reflect goal utility and others ethical implications. These metrics are not combined into an overall utility function but passed to the ethical decision module where they are reasoned about declaratively (see section III-C). The precise set of metrics chosen is both domain and ethical criteria specific, but the set used here demonstrates the basis upon which they should be selected:

- *H-robot distance to danger* – danger to the human can be considered to increase the closer it gets to a dangerous object;
- *E-robot distance to danger* – danger to the robot can be considered to increase the closer it gets to a dangerous object;
- *E-robot distance to objective* – here the robot objective is to get to a designated object in the environment, the closer it gets to this object the better its goal can be considered achieved;
- *E-robot wait time at intercept point* – how long before the human the robot arrives at the intercept point (used for law 4a);
- *E-robot walking distance* – how far the robot walks (used for law 4b).

While the set described here is relatively simple, more complex metrics could be calculated if the context demands it. One caveat being that any metric added must produce a numerical output (however it is calculated), such that plans can be compared using the declarative logic structures of the Ethical Decision Module, resulting in a definitively best plan.

B. Planner Module

The Planner Module generates additional proactive behavioural alternatives, and does so in such a way that they are likely to satisfy the ethical criteria. It is triggered if, after simulation and evaluation (by the other modules) of the behavioural alternatives suggested by the robot controller, the outcomes are deemed not to satisfy the ethical criteria by the ethical decision module.

In the current implementation the plans are generated using a heuristic based on assumptions of the ethical issues that will be encountered, and features specific to the use case. That is, plans that are likely to keep the human and robot from danger and give a range of values for the other metrics being evaluated. We have chosen this approach to allow us to demonstrate the utility of the architecture as a whole. Clearly it does not generalize well, requiring the existence of an appropriate heuristic. In Section VII we suggest future work in which the Planner Module could utilise probabilistic methods to intelligently sample the behaviour space.

In the demonstration case presented here, if the H-robot is predicted to be heading toward danger (as evaluated by the Ethical Decision module), proactive tasks are generated and tested. Three task alternatives are generated as points on the projected path of the H-robot. They are selected as the earliest point along the path the E-robot is able to reach in time to intercept, and then two equally spaced points further along the path. This simple heuristic is used as it is likely to produce behaviours in which the E-robot intercepts the H-robot, preventing it from reaching danger. We choose a limited number of alternatives to improve the responsiveness of the E-robot and prevent the Ethical Decision module from introducing delays. Further, considering a small number of alternatives matches with models of human cognition for similar processes [51], from which our architecture draws inspiration [49].

C. Ethical Decision Module

In order to select a behavioural alternative, the Ethical Decision Module utilises the simulation metrics (reported by the simulation module) to assess each task against a set of ethical rules. In our previous work describing a related architecture [49] simulation metrics were collated using a mathematical function to give a single numerical value for each alternative to allow comparison. Although this has advantages in terms of computational efficiency and performance tuning (through parameters in the function), it presents difficulties in terms of both human understanding and formal verification of the decision process. This is particularly problematic if model-checking is the chosen verification method, since this requires a finite search space and arbitrary numerical values tend to introduce infinite (or at least very large) search spaces. To overcome these two issues, our ethical decision module makes use of declarative logic based reasoning as typified by BDI agents.

An obstacle here is that current BDI programming languages are not widely known and represent a style of programming unfamiliar to those in robotics. Furthermore the need to use one programming language for decision making and a different one for the underlying control and then integrate these programs together on a robotic platform increases, rather than decreases the potential for errors. We have opted therefore to integrate BDI style reasoning into Python as a library, *BDIPython*. (Python is widely used for robotics coding.) *BDIPython* has been designed as a generic module that could be used in many settings and

can be considered as simply the tool by which we enabled verifiable BDI reasoning in our ethical decision module. However, since it has not been described elsewhere, we will digress briefly into an outline of its key features.

In *BDIPython* a BDI agent is a Python object and its reasoning cycle can be started and stopped as the program so desires. This agent object then interacts with the rest of the Python program which can be viewed as its environment.

BDIPython maps BDI concepts to Python constructs:

- **Beliefs**. We implemented a data structure for an agent's beliefs, referred to as the *belief base* in BDI programming, using a Python dictionary. The *belief base* represents a set of ground first order predicates for the purposes of logical reasoning[5] and will be treated as such in what follows. We briefly outline some implementation details in the supplementary materials Section II.
- **Goals** are also stored in a Python dictionary. We did not use goals in this case study[6] and so will not discuss them in any further detail.
- **Rules** consist of of two Python functions one of which represents a logical *guard* which determines whether the rule is applicable by inspection of the agent's belief and goal bases. This guard function may return values, representing the instantiation of variables in queries, that can be passed as parameters to the second function, the *rule body*. The *rule body* is executed if the rule is applicable.
  When more than one rule is applicable the system simply selects the first in the list of rules – as is standard in many logic programming languages.

Rule guards and bodies may contain arbitrary Python code, but *BDIPython* supplies several support functions to assist their construction, in particular functions for inspecting the belief and goal dictionaries and composing the results using propositional logic connectives such as AND and OR. It also provides functions for adding and removing beliefs and goals from within rule bodies.

The reasoning cycle for a *BDIPython* agent first updates its belief dictionary (in an application specific fashion) typically as a response to sensor inputs, then it manages its goals (checking and removing any that have been achieved), lastly it selects a rule and executes it. This reasoning cycle is shown in Figure 3.

BDIPython is best suited for rules whose guards can be expressed as propositional logic formulae over the agent's beliefs and goals and thus as functions which simply return true or false. However limited support is provided for richer reasoning in guards. In particular it is possible to construct a rule which selects *the best* option from among several beliefs, based upon a comparison function defined by the programmer, and passes this option as a parameter to the rule body. If the comparison function does not define an antisymmetric transitive relation (for instance, the relation it defines represents a cyclic graph) so no 'best' option can be identified, then the rule is considered inapplicable (i.e., its guard is considered to be false) and so not selected for execution. This requirement for antisymmetry and transitivity will turn out to be a key consideration when we discuss the verification.

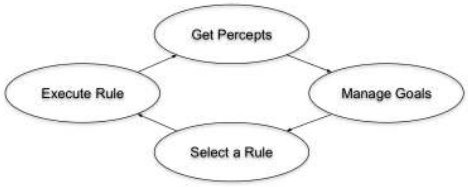

Figure 3: The Python BDI Reasoning Cycle

We use *BDIPython* to implement two behaviours within the Ethical Decision Module. Firstly the agent controls the triggering of the Planner Module. In the *get percepts* part of the reasoning cycle the human distance to danger is checked for a set of robot controller task alternative simulations. If it is below a threshold for all of them, a belief that pro-active tasks are needed is added to the belief base. This belief in turn is used as a guard for a rule that when executed signals the Planner Module. This mechanism is used as the Planner Module is a separate process, not part of the BDI agent; importantly it allows logging of said process through the belief base and rule activation.

Secondly we use the ability to select a best option to allow our ethical layer to select the most ethical of the tasks available to it and then transform the rule body into an intention to add the belief that this should be the current task. The Action layer then executes the current task.

At the start of the process for selecting the most ethical task in accordance with Asimov's laws, the agent's belief base contains a number of beliefs about the existence of candidate tasks suggested either by the underlying robot control system or by the Planner Module. The tasks are labelled, $t_1, \ldots, t_n$ (one label for each task), and the belief $task(t_i)$ is in the agent's belief base for each of these labels.

We implement two comparison functions for comparing two tasks t_i and t_j. The underlying

[5]Note that the belief base represents actual beliefs, hence it is ground. Queries over the belief base may contain variables which are instantiated by logical reasoning.

[6]since our ethical decision module is not motivated by the desire to achieve some particular outcome.

BDIPython implementation for selecting a best option ensures that all matching beliefs are compared to find the best (by iterating twice over the belief base), so we can use it to compare all tasks, t_i, t_j, such that $task(t_i)$ and $task(t_j)$ appear in the belief base.

These comparison functions used to implement Asimov's Laws, are designed to respect the order of precedence of said Laws. This is accomplished using a series of if-then-else statements which explicitly state this order. Two rules, one for each different comparison function, are used, the first for when the agent believes the human is close to danger and the second for when the agent believes the human is not close to danger (to represent the distinction between rules 4a and 4b discussed in Section III). This core set of rules and comparison functions form the primary component of the Ethical Decision Module. As the Ethical Decision Module implements the ethical rules, it is this component which needs to be verified.

We show mathematically the algorithm used by the Ethical Decision Module in Box 1 and discuss it here. In Box 1 we use the following notation:

- $G \leftarrow RB$ is our syntax for a *BDIPython* rule where G is the guard (evaluated against the belief and goal bases) and RB is the rule body which is executed if the guard is true. We use quantification with this notation – e.g. $\exists x.(G(x) \leftarrow RB(x))$ – to indicate when a parameter, x, is returned by the guard and passed to the rule body.[7]
- $\mathcal{B}(p)$ means that p is a predicate in the belief base of the agent.
- $\mathsf{add\_belief}(p)$ adds the predicate p to the belief base of the agent.

The two rules are shown in equations 1 and 2. These both use *BDIPython*'s support for picking a best option. In equation 1 the best task is chosen according to the comparison relation $\lhd_{wd}$ (where $t_1 \lhd_{wd} t_2$ means t_1 is preferable to t_2 as t_1 has a lower walking distance) in the situation where the agent believes $danger\_close$. In equation 2 the best task is chosen according to the comparison function $\lhd_{wt}$ (the preferable task will have longer wait time between task completion and human interception) in the situation where the agent does not believe $danger\_close$. Both rule bodies add a belief that the selected task is the 'current task' (this belief is then consulted by the Action Layer when deciding which task to execute).

The two comparison functions are very similar and are defined as relations in definitions 1 and 2 (and implemented as nested if-then-else statements). These relations represent our modified version of Asimov's Laws and use three other specially designed relations $\prec_{hd}$, $\prec_{ro}$, $\prec_{rd}$ which compare the two tasks with respect to how close the human is to danger ($\prec_{hd}$), how close the robot is to its objective ($\prec_{ro}$) and how close the robot is to danger ($\prec_{rd}$). These are used in order of precedence – i.e., the two tasks are only compared with respect to $\prec_{ro}$ if they are indistinguishable in terms of how close the human is to danger. Lastly – if there is no difference between the two tasks in terms of these three orders then $\lhd_{wd}$ compares them using the valuations of the walking distance associated with each task $v_{wd}(t_1) < v_{wd}(t_2)$, and $\lhd_{wt}$ compares them according to the waiting time associated with each task.

Our three relations pertaining to each individual Law are implemented in the same way using $\prec_m$ (the relation between two tasks according to metric m) where m is one of hd, ro or rd and $t_1 \prec_m t_2$ means t_1 is preferable to t_2 according to metric m In order to prevent small changes in sensor information having too great an effect on robot behaviour we use a threshold, th_m, for each metric so that if two tasks are both 'good enough' – e.g., in the outcome of neither task the human ends up particularly close to the danger – then they are considered comparable, $\approx_m$. Therefore in the case of a metric to be minimised (e.g., the closeness of the robot to its goal), t_1 is better than t_2 if the value calculated for t_1 on that metric, $v_m(t_1)$, is below the threshold for the metric, and $v_m(t_2)$ is above the threshold; t_1 is also better than t_2 if both $v_m(t_1)$ and $v_m(t_2)$ exceed the threshold and $v_m(t_1) < v_m(t_2)$. This is formalised in definitions 3 and 4.

We assume, in what follows, that tasks t_1 and t_2 have been generated in such a way that $v_m(t_1) \neq v_m(t_2)$ for any metric. This means that $\prec_m$ is antisymmetric.

The ethical behaviour of the robot is hence embedded not only in the BDI agent rules, but also in the set of thresholds used to govern if, and when, a law might be violated. For example, a high threshold for robot danger distance could make the robot more prone to self preservation, only considering tasks incomparable on this metric if both placed the robot a long way from the dangerous area. It is important to note that how these values are labelled and used makes their impact transparent to human observation and interpretation.

Clearly this approach to ethical decision making assumes that the chosen code of ethics used can be expressed in a similar way, i.e., with scorable metrics that can be used to compare the desirability of behavioural alternatives in a declarative fashion.

[7]Note that a use of an arrow symbol here is inherited from logic programming notation (and is commonly used in this context in BDI languages) but does not imply a logical relationship between guard and rule body.

$$(\exists x.\, \mathcal{B}(danger\_close) \wedge \mathcal{B}(task(x)) \wedge \forall y \neq x.\, x \lhd_{wd} y) \leftarrow \mathsf{add\_belief}(current\_task(x)) \quad (1)$$

$$(\exists x.\, \neg\mathcal{B}(danger\_close) \wedge \mathcal{B}(task(x)) \wedge \forall y \neq x.\, x \lhd_{wt} y) \leftarrow \mathsf{add\_belief}(current\_task(x)) \quad (2)$$

Note: In what follows we use the notation $v_m(t)$ to indicate the valuation of task, t, according to metric m. So $v_{hd}(t)$ indicates the distance task t leaves the human from danger and $v_{wd}(t)$ indicates the walking distance associated with task t.

Definition 1 ($\lhd_{wd}$): Task, t_1, is preferable to task t_2 if t_2 places the human closer to danger ($t_1 \prec_{hd} t_2$), else it is preferable if t_1 places the robot closer to its objective ($t_1 \prec_{ro} t_2$), else t_2 places the robot closer to danger ($t_1 \prec_{rd} t_2$) else t_1 has a shorter walking distance $t_1 <_{wd} t_2$. Formally $t_1 \lhd_{wd} t_2$ iff

1) $t_1 \prec_{hd} t_2$ or,
2) $t_1 \approx_{hd} t_2$ and $t_1 \prec_{ro} t_2$ or,
3) $t_1 \approx_{hd} t_2$ and $t_1 \approx_{ro} t_2$ and $t_1 \prec_{rd} t_2$ or,
4) $t_1 \approx_{hd} t_2$ and $t_1 \approx_{ro} t_2$ and $t_1 \approx_{rd} t_2$ and $v_{wd}(t_1) < v_{wd}(t_2)$.

Definition 2 ($\lhd_{wt}$): Task, t_1, is preferable to task t_2 if t_2 places the human closer to danger ($t_1 \prec_{hd} t_2$), else it is preferable if t_1 places the robot closer to its objective ($t_1 \prec_{ro} t_2$), else t_2 places the robot closer to danger ($t_1 \prec_{rd} t_2$) else t_1 has a shorter waiting distance $t_1 <_{wt} t_2$. Formally $t_1 \lhd_{wt} t_2$ iff

1) $t_1 \prec_{hd} t_2$ or,
2) $t_1 \approx_{hd} t_2$ and $t_1 \prec_{ro} t_2$ or,
3) $t_1 \approx_{hd} t_2$ and $t_1 \approx_{ro} t_2$ and $t_1 \prec_{rd} t_2$ or,
4) $t_1 \approx_{hd} t_2$ and $t_1 \approx_{ro} t_2$ and $t_1 \approx_{rd} t_2$ and $v_{wt}(t_1) > v_{wt}(t_2)$.

Definition 3 ($\prec_m$): Given a metric, m and a threshold th_m then $\prec_m$ defines a relation on tasks such that $t_1 \prec_m t_2$ iff $v_m(t_1) < th_m \wedge th_m < v_m(t_2)$ or $v(t_1) < v_m(t_2) < th_m$ (where the metric is to be maximised – $\prec_{hd}$ and $\prec_{rd}$) and $t_1 \prec_m t_2$ iff $th_m < v_m(t_1) \wedge v_m(t_2) < th_m$ or $th_m < v_m(t_2) < v_m(t_1)$ (where the metric is to be minimised – $\prec_{ro}$).

We use the notation $t_1 \approx_m t_2$ if t_1 and t_2 are considered incomparable on $\prec_m$ (i.e., both are below the threshold, th_m, introduced in definition 3).
Definition 4 ($\approx_m$): $t_1 \approx_m t_2$ iff $t_1 \not\prec_m t_2$ and $t_2 \not\prec_m t_1$.

Box 1: Mathematical description of the BDI Code for the Ethical Decision Module

D. Ethical Black-box Module

In line with the suggestion by Winfield and Jirotka [25] we have a module that logs the outputs of the other modules. Specifically, it logs:

- location data;
- behavioural alternative parameters;
- metrics for each alternative as scored by the evaluation module; and
- the reasoning process used in the ethical decision module: the belief base at each iteration, rule invocations, task selection decisions and reasoning, the plan output to the robot controller

Each item of data is logged with a time stamp to allow recreation and analysis of a given moment. Further details of the data logged, particularly of the reasoning process, are given in Section V as they relate to the BDI implementation explained in Section III-C.

IV. Verification of the Ethical Decision Module

We adopt the verification methodology from [11] which describes the formal verification of rational agent components in autonomous systems. This uses model checking to demonstrate that the rational agent always tries to act in line with requirements and never *deliberately* chooses options that lead to bad states (e.g. ones the agent believes are unsafe). The agent's program is assessed to determine logical predicates that represent information coming in from the outside world. All possible combinations of these inputs are then explored via the model checker, allowing the verification to be agnostic about how the real world might *actually* behave; it simply verifies how the agent behaves *no matter what* information it receives.

In this context, model checking can be viewed as a kind of exhaustive testing. The *BDIPython* agent executes its reasoning cycle – at given moments in this cycle the agent receives some information from the external program (that certain tasks are available, for instance, and the human is close to

danger and that the walking distance is less for one task than for another) and then continues execution. When execution completes, or the agent reaches a state that has already been examined, the model-checker backtracks to the last perception of interest and supplies a different set of information to the agent until the result of all possible sets have been explored. During this process, the model checker checks that certain properties hold (e.g., that the agent never selects a task that places the human in danger).

In order to apply model-checking to our Ethical Decision Module we need to translate our *BDIPython* program, plus the semantics of the *BDIPython* reasoning cycle into the input language of a model checker.

The approach from [11] is implemented in the MCAPL framework [52] which provides access to model checking facilities to programs written in a wide range of BDI-style agent programming languages so long as those languages have a Java-based program interpreter. The MCAPL framework has two main sub-components: the *AIL-toolkit* [53] for implementing interpreters for rational agent programming languages and the Agent JPF (AJPF) model checker which is an extension of the Java Pathfinder (JPF) model-checker for Java programs [54].

AJPF is a customisation of JPF that is optimised for AIL-based language interpreters. Agents implemented using the AIL-toolkit can thus be model checked in AJPF. It provides a property specification language to support reasoning about temporal properties of BDI programs and also provides support for implementing models of the agent's external environment in Java so that these models return combinations of the possible inputs to the agent. The AIL provides data structures for *beliefs*, *intentions*, *goals*, etc., which are subsequently accessed by the model checker and on which the modalities of the property specification language are defined.

Thus, we must first implement an interpreter for the *BDIPython* reasoning cycle in the AIL. Second, a parser must be implemented from *BDIPython* programs into suitable Java data structures in order for the program to execute in this interpreter. Third, we must create an appropriate environment to model the rest of the Python program, and finally our properties of interest must be expressed in the AJPF property specification language. Some of this work was independent of the specific application described here and can be re-used in future applications – for instance once the parser was implemented it could be used for any *BDIPython* program not just ones expressing ethical decision making. Nevertheless, as when we introduced *BDIPython*, we briefly outline this work here since it is not described elsewhere.

A. Verification of Python-based Ethical Decision Module

We use the AIL-toolkit not to create an interpreter for a full programming language but to build a model of a *BDIPython* agent. Because *BDIPython* has an explicit reasoning cycle we can model this easily within AIL and then AJPF can be used to simultaneously build and verify a model of the *BDIPython* agent. Much of the supporting and surrounding Python code is then treated as part of the agent's environment in a black-box fashion.

The use of the MCAPL framework is not fundamental to our approach. We have adopted it because of its dual convenience as a framework for building verified models of BDI agents and for its support for the formal verification of autonomous systems controlled by rational agents. For full assurance of Ethical Decision Modules of this kind, we would recommend the development of a *program model-checker* for Python (that is a model-checker that operates directly on Python code rather than on a model of some sub-system), or possibly a customisation of AJPF to Jython [55] (the JVM Python interpreter).

In order to use the MCAPL framework to verify our Ethical Decision Module we need to use the AIL to build a Java data structure that represents the *BDIPython* agent and then execute this in our implementation of the *BDIPython* reasoning cycle in order to create a model of the engine's reasoning. This is a different challenge to the normal implementation of a BDI-language in the AIL. Python lacks the formal operational semantics that underpin most BDI languages (though semantics have been reverse engineered for large parts of the language [56], [57]) which in turns means *BDIPython* lacks a formal semantics. Hence, the system, as a whole, lacks the clear separation between BDI concepts and the 'environment' (which is how we will need to treat the rest of the Python program) that is present in many BDI languages.

From the *BDIPython* code for an agent, we automatically construct a representation of the Python 'agent' object using data structures from the AIL-toolkit and then impose an operational semantics upon the Ethical Decision Module based on the reasoning cycle shown in Figure 3. The AIL provides a data structure for agents which contain a *belief base*, a *goal base* and a *rule base*. The belief and goal bases are sets of ground first order predicates. We translate the Python dictionaries representing the Python agent's belief and goal bases into a set of such predicates according to semantics outlined in supplementary material Section II.

AIL rule structures consist of a guard (evaluated against the agent's belief and goal base, and poten-

tially other structures if relevant) and a sequence of *deeds* which represent atomic actions (again represented as first order predicates) in the environment or the addition or removal of beliefs and goals. We impose restrictions on the Python code that may appear in the functions defining guard and rule bodies in order to more easily construct these rule descriptions in Java.

We assume that a rule body contains only a sequence of atomic Python expressions (i.e., no control structures such as 'if' statements or 'for' loops). Each of these atomic expressions is treated as an action in the environment unless it is add_belief or drop_belief from *BDIPython*, in which case it is treated as belief addition or removal. Where this function definition contains extra parameters these are converted to variables wherever they occur in the body of the rule. So, for instance, if param1, is a parameter of the rule definition and do_something_with(param1) appears in the rule body then this will be converted into the predicate $do\_something\_with(X)$ where X will be bound to the input variable at run time.

We require rule guards to be expressions built up from the library functions B (the agent believes), G (the agent has a goal) and the support provided for propositional logic: AND, OR, and NOT plus comparison functions (as used when the rule is to select the best option). With the exception of comparison functions these can be converted to an AIL guard expression with the obvious semantics.

We restrict comparison functions, $x \lhd y$, to a sequence of nested **if** statements, each **if** with condition $condition_i$ (for $1 \leq i \leq n$) such that the function returns true (i.e., $x \lhd y$) if the condition is satisfied. Thus $x \lhd y$ if $condition_i(x, y)$ returns true for some i and $x \not\lhd y$ otherwise). Each condition is a conjunction of (possibly negated) Python expressions (treated as functions on x and y that return true or false).

We add an additional phase into the reasoning cycle of agents in our AIL-model in which *Python calculations* can take place and expect the verification environment to return the results of these calculations which the agent then stores in a special *calculation base*. This calculation base can be consulted when evaluating the guards on rules in the same way that the belief base is consulted[8]. The Python expressions that make up the conditions in our comparison functions are treated as such Python calculations and stored in the calculation base.

From the rule expression and the conditions and Python expressions that make up the comparison function, it is straightforward, if fiddly, to construct a logical expression representing the rule guard. Consider a rule of the form shown in equation 3.

$$\exists x.\ (\mathcal{B}(something(x)) \wedge \forall y \neq x. \mathcal{B}(something(y) \rightarrow x \lhd y) \leftarrow rb(x) \quad (3)$$

The guard of this rule is essentially the expression on the left hand side of the $\leftarrow$ though it is transformed slightly for use in logic programming style reasoning [9]. $x \lhd y$ is expanded out to the expression $condition_1(x, y) \vee \ldots \vee condition_n(x, y)$. Each expression $condition_i(x, y)$ is expanded out in turn to a conjunction of (possibly negated) Python expressions. So, for instance if $condition_i(x, y)$ were $x \not\prec_{hd} y \wedge y \not\prec_{hd} x \wedge x \prec_{ro} y$ then each of $x \prec_{hd} y$, $y \prec_{hd}$, and $x \prec_{ro} y$ is treated as a Python expression and the calculation base will be consulted for their value. If some x is found that satisfies the expression then it will be passed to the rule body as a parameter[10].

Therefore our functions from Box 1 for comparing two tasks according to specific metrics $\prec_{hd}$, $\prec_{ro}$ and $\prec_{rd}$ as well as simple comparisons of walking distance and time using $<$ and $>$ become Python calculations under this process.

Once the translation of the Python code into an AIL agent data structure has been carried out it can be executed using a reasoning cycle created for it, using support for this provided by the AIL.

The agent data structure and executable reasoning cycle can be combined with an application specific implementation of an environment (i.e., one that can return all possible combinations of beliefs and calculations as required by the methodology we are using). This enables AJPF to construct a model of all possible executions of the agent.

Once we have such a model we can formally verify properties expressed in the AJPF property specification language. In general, constructing such properties is straightforward (the language has constructs that refer to the agent's beliefs and goals etc.). Python calculations, however, represent a non-standard part of our model and so we treat these as *percepts* in the property specification language. Percepts are facts that are *perceptible* to some observer of the environment but not necessary believed by the agent.

Note: Beliefs vs. Python Calculations: The decision about what information should be stored as beliefs and what should be calculated on the fly by Python is in the hands of the programmer using *BDIPython*. Input from sensors should be stored as beliefs, but other information can often plausibly be

[8]The AIL toolkit contains considerable support for this kind of customised base construction.

[9]The details are unimportant here but the mechanism is entirely standard.

[10]via a process of unification in the AIL implementation which uses many concepts from logic programming.

treated either way. In the current system, the distinction makes little difference to the verification.

V. Experimental Validation and Formal Verification

We have conducted a series of experiments in order to validate that our approach results in robot behaviour that adheres to the ethical rules described in Section III, and allows the system's reasoning to be transparently ethical, verifiable and proactive.

To first establish the system obeys Asimov's Laws we reconstructed the experiments previously carried out in [49], with our supplemental rules (laws 4a and 4b) disabled. We then duplicated the experiment that demonstrates human safety with those supplemental rules enabled to observe how they change the robot behaviour, while demonstrating that the robot still adheres to Asimov's Laws. A summary of the aims of each experiment is shown in Table I.

A secondary purpose of our experiments is to demonstrate the feasibility of an ethical black-box recorder as proposed in [25]. In particular, we aim to demonstrate the suitability of our BDI based approach for producing human scrutable logs through a highly transparent decision process. At each update iteration we record the belief base, the rule executed, and – if a compare rule is executed – the comparison results used for behaviour selection. Also logged is the data from the tracking system, the behavioural alternatives evaluated, and the scores in each evaluation metric. Each item of data is time-stamped to enable later reconstruction of all decisions made. This log data is then used to produce a description of the process that has been followed

In addition to the validation experiments, we have also followed the previously described formal verification procedure for the ethical decision module. The BDI rules and comparison functions were extracted from the Ethical Decision Module and parsed into data structures in the AIL. Properties expressing adherence to Asimov's laws were then verified using AJPF.

A. Experimental Setup

In these experiments we use two NAO robots as the H-robot and E-robot, as defined in Section III, i.e., a robot acting as a proxy for the human, and a robot controlled using our ethical architecture. All experiments were carried out in a 3m $\times$ 2.5m arena, with 2 objects designated A and B on the opposite side of the arena from the start locations of the robots; the setup is shown in Fig 4. Object A was designated 'dangerous', and B was designated as 'safe'. The robots are considered to have encountered danger if they get within 1m of the dangerous object. To enable the robots to navigate, and the E-robot's controller to simulate the environment and both robots, a 4 camera tracking system was used that, via reflective markers, tracks positions and orientations of the robots and objects (though the objects were static) at 30Hz.

At the start of each trial both H and E robots move to their designated start locations (see Figure 4), and when both have arrived the trial itself begins. Both H and E robots have pre-determined objective positions (to move to), which are varied by experiment. Whether the E-robot's objective is attributed to its own controller or via an instruction from the H-robot is determined in the experimental script and varied by experiment.

Figure 4: The experimental setup, the red NAO is H-robot and the blue NAO is E-robot.

Both H and E robots use the tracking information to simulate low-level collision detection that would cause them to stop if within 0.5m of each other or an object. The E-robot assumes that the H-robot does not know if a location is dangerous. The E-robot controller can re-start its motion if the H-robot is observed to be stationary – it treats a stationary human as an obstacle to be avoided.

As the H-robot has no ethical layer it simply moves to its predetermined objective unless proximity to the E-robot causes it to stop. The ethical layer in the E-robot operates at about 1Hz[11], re-evaluating a set of plans and selecting the most ethical course of action, determining the goal of the E-robot. The E-robot's control layer suggests A and B as possible goals for evaluation, supplemented if required by the Planner Module in its ethical layer.

The walking speed of the H-robot robot is set to be lower than the speed of the E-robot. The difference in speed gives the E-robot a larger range for intercepting the H-robot. The speeds of the H-robot and E-robot were approximately $0.02ms^{-1}$ and $0.04ms^{-1}$ respectively. These speeds were not varied during the experiment.

B. Experiment Settings and Results

All results reported below are obtained with no changes to the controllers of the H and E-

[11] empirically determined to provide sufficiently reactive planning, without overburdening the NAO controller with control requests.

Experiment	Law(s) Tested
Self Preservation	Law 3
Obedience	Law 2 overrides Law 3
Save Human	Law 1 overrides Law 3
Save Human and Obey	Law 1 overrides Law 2 and 3
Save Human with Supplementary Rules	Law 4a

Table I: Summary of experiments

robots. The *E-robot danger distance* threshold and *H-robot danger distance* thresholds are set to 1.5m; this allows for some errors in E-robot simulations while still keeping the H-robot safe. The *E-robot objective distance* threshold is set to 0.5m, the distance needed to consider the objective reached. The different experimental conditions are set by the goals of the robots, and whether or not the E-robot goal is commanded by the H-robot. Each experiment is a single demonstration trial (representative of multiple trials conducted with the same settings) to facilitate clear discussion of the data logs recorded in the EBB. For each experiment the decision processes are obtained directly from those logs.

1) Self Preservation: The aim of this experiment is to demonstrate adherence to Law 3, i.e., that the E-robot acts to keep itself safe, if (and only if) this does not conflict with obedience (Law 2) or human safety (Law 1). For this experiment the H-robot remains stationary at its start location, the E-robot objective is A and it is not H-robot commanded.

There is no inferred goal as the H-robot is stationary, so the H-robot is believed to be safe. There is no H-robot command, and of the two behavioural alternatives, moving to location A or to location B, A is considered dangerous to the E-robot (i.e., danger distance is below the threshold) while B is not. Hence, the E-robot moves to location B (Figure 5).

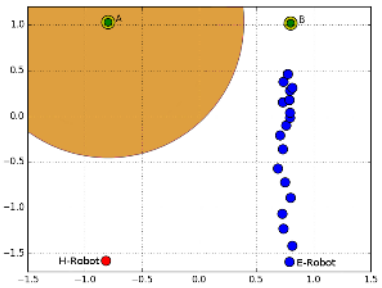

Figure 5: The tested behavioural alternatives for the initial plan (yellow dots), are the A and B objects. The E-robot path (blue dots) moves to the non-dangerous object B and stops 0.5m from it (due to collision avoidance). The orange circle shows the E-robot and H-robot danger thresholds.

2) Obedience: The aim of this experiment is to demonstrate that adherence to Law 2 overrides Law 3, i.e., the E-robot will walk toward danger if commanded by the H-robot. This experiment only differs from the previous one in that the robot's objective is now H-robot commanded.

There is no inferred goal as the H-robot is stationary, so the H-robot is believed to be safe. There is belief in an H-robot command, so for the two behavioural alternatives, locations A and B, B is above the *distance to E-robot objective* threshold, while A is below the threshold. The values for *robot distance to danger* are not compared due to Law 2 having priority. Hence, we observe that the E-robot moves to location A even though it is dangerous to the robot (Figure 6).

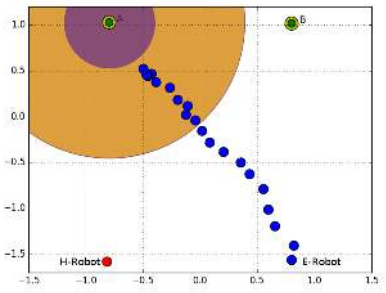

Figure 6: The tested behavioural alternatives for the initial plan (yellow dots), are the A and B objects. The E-robot path (blue dots) moves to the dangerous object A and stops 0.5m from it (due to collision avoidance). The orange circle shows the E-robot and H-robot danger thresholds. The purple circle is the robot objective distance threshold.

3) Save Human: The aim of this experiment is to demonstrate that adherence to Law 1 overrides Law 3, i.e., the E-robot will act to prevent the H-robot from coming to harm even if doing so is dangerous for itself. For this experiment the H-robot objective is A, the E-robot objective is B and it is not H-robot commanded.

The inferred goal for the H-robot is A, so the E-robot believes the H-robot is in imminent danger. Of the 5 behavioural alternatives the objects A and B and intercept point i3 (see Figure 7) allow the H-robot to come closer to danger than the allowed threshold. Hence, points i1 and i2 are considered more desirable than any of the others, and though it puts the E-robot closer to danger than the threshold, i1 is selected as it puts the robot further from danger than i2. The robot objective distance is not

checked due to there being no belief in any H-robot command. Hence, we observe the E-robot moves to intercept the H-robot. Due to inaccuracies in simulated movement times the intercept occurs with the E-robot approaching from the side. The precise target point shifts over time due to replanning compensating for inaccuracies in simulated potential intercept points and movement times. It is important to note that despite this reality gap the intercept still occurs. After interception, the H-robot is no longer believed to be in danger and the E-robot moves to its remaining goal B (Figure 7).

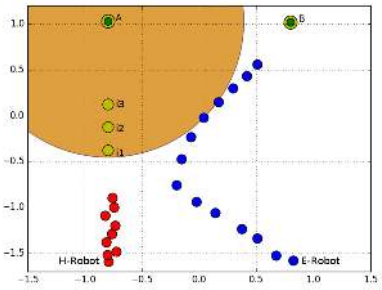

Figure 7: The tested behavioural alternatives for the initial plan (yellow dots), are the A and B objects, and the intercept points i1-i3. The E-robot path (blue dots) moves to intercept the H-robot (path in red dots). The H-robot stops when the E-robot approaches within 0.5m. The E-robot then continues to the safe object B, and stops 0.5m from it (due to collision avoidance). The orange circle shows the E-robot and H-robot danger thresholds.

4) Save Human and Obey: The aim of this experiment is to demonstrate that adherence to Law 1 overrides Law 2 and 3, i.e., the E-robot will act to prevent the H-robot from endangering itself even if doing so is dangerous for itself and also ignores a direct command. For this experiment the H-robot objective is A, and the E-robot objective is also A, as commanded by the H-robot.

Initial behaviour and beliefs are the same as in the previous experiment, the notable difference being that points i0-i2 are seen as being too far from the human-commanded objective, but this is overridden in the same way as the perceived danger to the E-robot. After interception, the H-robot is no longer believed to be in danger and the E-robot obeys the human command in the same way as in the *Obedience* experiment above (Figure 8).

5) Save Human with Supplementary Rules: The aim of this experiment is to demonstrate the effect of one of our context-specific extension to Asimov's Laws, Law 4a, and hence the flexibility of our approach while still maintaining verifiable and scrutable ethical decisions. This mainly differs from the *Save Human* experiment above by the inclusion of the additional ethical rules. To do so

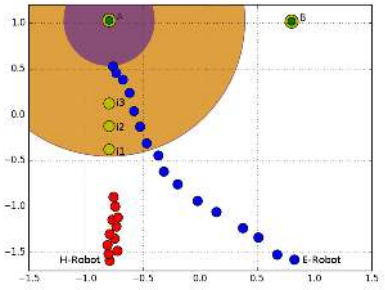

Figure 8: The tested behavioural alternatives for the initial plan (yellow dots), are the A and B objects, and the intercept points i1-i3. The E-robot path (blue dots) moves to intercept the H-robot (path in red dots). The H-robot stops when the E-robot approaches within 0.5m. The E-robot then continues to object A as it was commanded by the H-robot, and stops 0.5m from it (due to collision avoidance). The orange circle shows the E-robot and H-robot danger thresholds. The purple circle is the robot objective distance threshold.

we extended the conditional rule set in the BDI logic to enable selection of the appropriate plan comparison function. In addition, to allow for the secondary rules to operate properly with our arena size, a robot is considered in danger if it is within 0.5m of a dangerous object; this reduces the E-robot and H-robot danger thresholds to 1m.

We observe that an intercept occurs at a point much closer to the danger than without the supplementary rules. In the initial plans, the H-robot is believed to be far from danger, and only points i1 and i2 do not put the human or robot too close to danger; i2 is selected as it allows for a longer wait time than i1. However, it is important to note that the intercept occurs actually on the H-robot path, rather than from the side as in previous experiments; the waiting time allowed for, combined with frequent replanning, compensates for inaccuracies in simulated travel times, resulting in better interception (Figure 9).

C. Verification of Ethical Reasoning in our Asimov-based Proactive Ethical Decision Module

While we were able to experimentally validate the behaviour of the robot in several specific examples, the link between *BDIPython* and the AJPF verification system allows us to provide deeper analysis of the ethical layer. We consider here the implemented Ethical Decision Module associated with task selection when the human has specified an objective through a direct command. This is the engine shown in Box 1.

This agent should believe that some task t is the current task if it conforms to our modified version

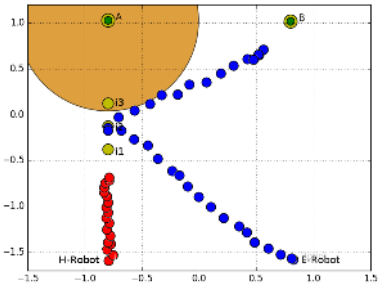

Figure 9: The tested behavioural alternatives for the initial plan (yellow dots), are the A and B objects, and the intercept points i1-i3. The E-robot path (blue dots) moves to intercept the H-robot (path in red dots). The H-robot stops when it approaches within 0.5m of the E-robot. The E-robot then continues to the safe object B, and stops 0.5m from it (due to collision avoidance). The orange circle shows the E-robot and H-robot danger thresholds.

of Asimov's laws – i.e., it is the best of all the tasks available as described in Section III. The Python code for the Ethical Decision Module is automatically converted to a Java data structure as described in Section IV-A.

We also construct a verification environment for this agent. Verification environments are designed to generate random choices for inputs to the agent (in this case, beliefs put by the environment in the belief base and the possible results of Python calculations in comparison functions). The process of model-checking then explores all possible combinations of these choices. Therefore, our verification environment may, at random, return $danger\_close$ as a belief. We considered cases where either two or three tasks are available: $task1$, $task2$ and $task3$. Therefore the environment always returns $task(task1)$ and $task(task2)$ as beliefs in the case of two tasks and $task(task1)$, $task(task2)$ and $task(task3)$ in the case of all three. Note that in our experimental set-up the ethical layer automatically generates three additional tasks for consideration when it deduces that the tasks suggested by the task layer result in the human moving too close to danger; hence, the three task case is sufficient to verify this layer.

At the point where the agent requests the value of Python calculations then the environment considers all possible results for the Python expressions appearing in the comparison functions. For instance, in the case of $\prec_{hd}$ it returns either

1) $task1 \prec_{hd} task2$ or
2) $task2 \prec_{hd} task1$ or
3) neither (indicating that $task1 \approx_{hd} task2$).

It similarly returns a choice for the comparison of $task1$ and $task3$ and for the comparison of $task2$ and $task3$. In the case of the movement and waiting times, we assume that these impose a strict order on tasks so that there are only two possible results. Model-checking allows us to consider all possible combinations of the results of these beliefs and calculations and so verify that our Ethical Decision Module makes the correct (most ethical) choice in each case.

Equations (4), (5) and (6) describe three properties representing Asimov's laws in the AJPF property specification language. The AJPF property specification language uses standard Linear Temporal Logic (LTL) operators: $\Box$ means "it is always the case that" and $\Diamond$ means "it is eventually the case that". LTL is extended with specific operators for BDI concepts, so $\mathcal{B}$ means that something is in the agent's belief base and $\mathcal{P}$ means that something is "perceptible" – in the case of our Python agent we interpret this as meaning it is in the agent's calculation base (a full description of the AJPF property specification language and its semantics can be found in [52]).

$$\Box((\mathcal{B}(current\_plan(task1)) \rightarrow \neg\mathcal{P}(task1 \prec_{hd} task2) \quad (4)$$

$$\Box((\mathcal{B}(current\_plan(task1)) \wedge \mathcal{P}(task2 \prec_{ro} task1) \rightarrow \mathcal{P}(task1 \prec_{hd} task2) \quad (5)$$

$$\Box((\mathcal{B}(current\_plan(task1)) \wedge \mathcal{P}(task2 \prec_{hd} task1) \rightarrow \mathcal{P}(task1 \prec_{ro} task2) \vee \mathcal{P}(task1 \prec_{rd} task2) \quad (6)$$

The three properties state that: it is always the case that if $task1$ is believed to be the current task then Python has calculated that $task1$ either does not place the human in significant danger or, if it does, then $task2$ places the human in greater danger (property (4) – corresponding to Asimov's first law); it is always the case that if $task1$ is believed to be the current task and Python calculates that it places the robot further away from its (human specified) objective than $task2$ then Python has calculated that $task2$ places the human in more danger than $task1$ (property (5) – corresponding to Asimov's second law); and lastly that if $task1$ is believed to be the current task and Python calculates that it places the robot in more danger than $task2$ then either $task2$ places the robot much further from its objective than $task1$ or it results in the human being in much closer to danger than $task1$ (property (6) – corresponding to Asimov's third law). Similar properties can be constructed to compare $task1$ and $task3$ etc.

Table II: Results of Verification in the 2 Plan Case

	Time (seconds)	States in the Model
(4)	43	1192
(5)	48	1192
(6)	52	1192

Table III: Results of Verification in the 3 Plan Case

	Time (hours minutes seconds)	States in the Model
(4)	64h 10m 23s	1,708,076
(5)	72h 33m 22s	1,708,076
(6)	76h 19m 07s	1,708,076

The results of verifying the three properties for the 2 plan case on a 4 core 3.4 GHz iMac with 8 GB memory running MacOS 10.13.1 are shown in table II. We record the time taken for the verification and the number of states in the resulting model of Ethical Decision Module execution created by the system.

We were unable to perform a similar verification in the three task case because the combinatorial explosion involved caused the size of the model to become too large and we terminated the verification process after one week. We will discuss this further in Section VII.

As mentioned previously, *BDIPython*'s support for picking a best option selects no task in the case that the comparison function does not represent an antisymmetric transitive relation and the corresponding guard expression in our AIL implementation naturally behaves the same way. Our verification environment generates many cases where this is not the case – it does not even guarantee that the subsidiary relations represented by $\prec_m$ are transitive. This meant that the attempt to prove property (7) (Eventually either $task1$, $task2$ or $task3$ is believed to be the current task), for instance, failed rapidly with a counter-example.

$$\Diamond B(current\_task(task1)) \vee B(current\_task(task2)) \vee B(current\_task(task3)) \quad (7)$$

This is a known issue with so-called *unconstrained environments*. While they are agnostic about the behaviour of the world and system beyond the BDI agent, and while they capture correct behaviour for all possible inputs, they usually represent over-generalisations of reality and flag up many false negatives. The solution to this is to create a *constrained abstraction* of the environment that embodies certain assumptions about the behaviour of the real world.

We modified our unconstrained environment with the following assumptions:

- All the predicates: $\prec_{hd}$, $\prec_{ro}$, $\prec_{rd}$, $<$ and $>$ represent transitive relations (i.e., if $t_1 \prec_{hd} t_2$ and $t_2 \prec_{hd} t_3$ then it is automatically the case that $t_1 \prec_{hd} t_3$).
- The overall relations specified by $\lhd_{wd}$ and $\lhd_{wt}$ are antisymmetic and transitive.

This allowed us to verify (7) in approximately 3 days for the three task case (thanks to the reduced search space) as well as properties (4), (5) and (6). The times taken for these verifications on a 4 core 3.4 GHz iMac with 8 GB memory running MacOS 10.13.1 are shown in Table III.

We note that there is no guarantee that assumptions made for constrained environments are correct. It may, however, be possible to validate them through other means – for instance it is straightforward to prove that our comparison functions are antisymmetric and transitive given some basic assumptions about the behaviour of objects in space; informal proofs are given in section I of supplementary materials. A methodology has recently been developed to allow such assumptions to be checked using *runtime verification* [58]. In this methodology a specification of the assumptions is created as a *trace expression* [59] which is then automatically converted into both the environment used by the AJPF model-checker and a *runtime monitor* that operates when the system is deployed and can react if it detects that the environment is violating the assumptions used during verification (and so the system is now operating in an "unverified" state).

While the time taken to perform verification may seem slow. It is important to note that the work we did on the translation of *BDIPython* programs into the input for AJPF does not need to be repeated. Therefore a programmer need only write a *BDIPython* program and supply a list of sensor inputs and Python calculations in order to automatically verify properties of their programs. This has advantages over many other verification techniques which require programs to be transformed by hand into some modelling language, and may also require manual intervention to guide the proof process.

VI. Related Work

Here we review three areas of research which relate to different elements of the work described in this paper. Firstly, we discuss anticipation in robots, a core design principle of our architecture. The ability to model and predict ethical consequences is, we believe, a differentiating feature of our work. Secondly, we note the paucity of experimentally tested ethical robots, and relate our contribution to previous work. Finally we argue that we have advanced the field of machine ethics verification significantly beyond that which was undertaken previously.

A. Anticipation in Robots

Providing robots with the ability to anticipate the future through the use of simulation-based internal models integrated into their control architecture has, in recent years, been demonstrated by a number of researchers. For example Vaughan and Zuluaga show that self-simulation of both a robot and its environment can be used to overcome incomplete self-knowledge and enable navigation task planning [60]. Similarly, Bongard *et al.* describe a 4-legged starfish-like robot that self simulates in order to learn its own morphology and how to control it, i.e., compensate for lack of self-knowledge (although without simulating the environment) [61].

In addition to self and environment simulation, the simulation of other agents may also be required. Zagal *et al.* use simulation in soccer robots to facilitate the adaptation of behaviours before they are deployed on the real robots; i.e., using the simulation to test how possible behaviours might actually operate in the environment with other robots [62].

The common methodology underlying the reported works on simulation based anticipation is the principle of hypothesis generation and testing. This has been demonstrated to be a powerful approach to robot control with a variety of applications. However, these differ from the work we have presented here in the purpose of hypothesis evaluation, i.e., previous work has focussed on task performance which we supplement with evaluating ethical consequences. Further, the robot simulates humans in the environment in addition to itself. Previous works have not simulated environments with human actors.

B. Ethical Robots

Machine ethics is a nascent field, consisting of only a few studies implementing ethics on actual robots. To the best of our knowledge, the efforts of Anderson and Anderson on the GENETH system [63], Bringsjord's *Akratic Robot* [64], and our previous work [10], [49] are the only instances of real robots equipped with (limited) moral principles.

GENETH has been developed over a number of years [2], [65], and uses an approach based on evaluating planned actions against a set of ethical constraints, but as there was no simulator only reactive ethical decisions were possible. GENETH uses inductive logic programming as a machine learning process with input from domain ethicists in order to determine its ethical principle. This principle is then represented in a fashion which allows transparent explanations to be provided for decisions (as we have recommended here). Among other things this approach demonstrates that bottom-up approaches to defining an ethical system can be adopted without necessarily sacrificing transparency.

Bringsjord *et al*'s [64] work builds on a program of developing a logic, the *deontic cognitive event calculus,* $\mathcal{DCEC}_{CL}$, in which various ethical theories can be expressed. The Akratic robot, for which a simple example has been implemented on a NAO robot, considers a scenario in which a robot charged with guarding a prisoner of war must choose between retaliating with violence to an attack (and satisfying a self-defence goal) or refraining from retaliation. It is referred to as *Akratic* from the Greek *akrasia* referring to when a person acts in contradiction to their better judgement. Bringsjord *et al.* argue that the underlying robot architecture, into which modules for self-defence and detainee management are embedded, must be capable of ethical reasoning in order to detect when such conflicts may arise and prevent them occuring (either by preventing the installation of conflicting modules or by over-riding goal-based reasoning when it conflicts with deontologically expressed obligations and prohibitions). This reflects our insistence that verification is an important aspect of implementing ethical reasoning though in the case of the Akratic robot the approach is to embody ethical verification as a fundamental part of the robot's operating system.

In our previous work we used a similar approach to that presented here using a simulator to evaluate *what-if* hypotheses. In contrast to the work presented here ethical decisions were made through evaluation of an ethical desirability function. This resulted in significant challenges in modelling the system for verification [66], and decisions were much less transparent with this approach.

In addition to these few implementations there have been some additional studies examining machine ethics from either a theoretical (e.g., [12], [67]) or simulation standpoint (e.g., [8]). All of these studies have used a similar approach i.e., planned actions are assessed against a set of ethical constraints, with their ethical implications assumed from known features of the environment.

C. Verification of Ethical Machine Reasoning

Previous work on the verification of ethical machine reasoning has focused on the use of the AJPF system that we have used here. Work in [66] attempts to verify the system proposed in [10] upon which the work reported here is also based. In this verification a new version of the system was produced in the AIL using two bespoke languages, one for the ethical consequence engine and another to represent action selection in the robot. The behaviour of the system was verified in an environment consisting of a simple 5×5 grid. A number of

discrepancies became apparent during this verification – for instance the verification on the 5×5 grid was converted into a model for the Prism model-checker [68] and gave very different probabilistic results to those derived by experiments. One lesson learned from this experience was that it is difficult to accurately extract models of Ethical Decision Modules written in Python (and by assumption in other common procedural languages used for robotics) which are tightly embedded in larger programs in the same programming language. One of our key aims here was to strengthen the link between the code produced by the programmer of the Ethical Decision Module and the model that was used in verification. This is achieved via the two-fold approach of supplying a BDI library for Python, encouraging a cleaner separation of the ethical reasoning from the rest of the system, and providing an automatic mechanism to extract code written using that library into a model in AJPF.

Work in [45] also uses AJPF to verify an ethical module operating as part of a larger system. It assumes that ethical reasoning is only invoked in special cases – i.e., that normal operation of the system is ethical by default. However, when some unexpected event occurs, AI techniques such as planning or learning are used to generate a new course of action. Transparent ethical reasoning can be used to choose between the options produced by the AI system based on a number of *ethical principles* and a context-dependent priority among those principles. The general approach is similar to that described here. However an entirely hypothetical system written in a bespoke language is verified. We have verified an Ethical Decision Module for an existing system written in a widely used language. Work in [45] does consider a context-dependent way for an Ethical Decision Module to resolve conflicts among competing ethical principles, as opposed to the strict universal ordering we have considered here.

VII. Further Work

Here we have demonstrated that plans, selected according to a set of ethical rules, can be verified. Pro-active plans are generated by a heuristic designed for the simple case study presented here. However, the utility of plans generated in this way are reliant on the design of the heuristic, and even in our simple demonstration case are unlikely to represent the best possible solutions. Indeed, it is relatively easy to imagine scenarios where heuristics of this form are non-trivial to design. Hence, we suggest that a better, more generalisable method for pro-active plan generation would make the system applicable in a wider range of contexts. One possible method to overcome this in future work is to use a machine learning technique such as Baysian optimisation of Gaussian processes [69] to better sample the plan space, balancing different ethical criteria metrics, hence generating plans that are likely to be ethically desirable. One immediately apparent challenge for such an approach is defining how to combine metrics into an overall score (needed for the optimisation process) reflective of how the Ethical Decision Module does its reasoning.

In order to facilitate clear demonstration of the principles of operation of our ethical layer architecture (as well as the verifiable and scrutable reasoning of the BDI based Ethical Decision Module) our case study only required a simple simulation, with limited models of the world and the proxy human. It is easy to envisage scenarios, closer to real world usage of an ethical robot, where a more complex simulation module would be required. One avenue of future work in this direction is to examine better modelling of the human decision process. A key part of such modelling would be some artificial theory of mind, reducing the dependency on overly simple assumptions of likely human actions.

JPF, which underpins AJPF, is specifically created to provide what is known as *program model-checking* capabilities for Java-bytecodes. In program model-checking the actual executable code of a system is verified as opposed to a model of the system. Program model-checking is, in general, more resource intensive than normal model-checking and can not handle large search spaces. By using AJPF to check *BDIPython* models, therefore, we are suffering from its inability to examine a large search space (hence our inability to prove the system obeys Asimov's laws in an unconstrained environment with even three plans) without gaining the advantage of verifying the actual executable code of the system. There are a number of approaches to improving this situation including implementing a custom model-checker for Python, or executing *BDIPython* in Jython [55], a Java-bytecode based interpreter for Python.

Further development of *BDIPython* itself is also useful, in particular to increase support for predicate logic-like representations in guards which can pass instantiations of variables/parameters from guards to rule bodies (as in the case of `add_pick_best_rule`). We anticipate a need for both a selection of specialised functions (like `add_pick_best_rule`), and a more general mechanism to allow arbitrary predicate logic expressions to instantiate parameters for execution.

VIII. Conclusion

We have considered the question of how ethical reasoning should be implemented in a robot, assuming the popular architecture (seen in, for instance [10], [45], [48], [63]) with a dedicated

ethical reasoner. In particular we have argued that such an ethical control layer should aim to be

- proactive,
- transparent,
- and verifiable.

In order to achieve an ethical control layer with these properties we have devised an architecture in which behavioural alternatives proposed by the underlying control system are evaluated using explicit declarative reasoning, in the form of a rational agent. If all options are deemed unsatisfactory, the ethical control layer has the ability to proactively generate new options and submit these to ethical reasoning.

The use of a rational agent to reason about the ethics of options allows the system's decisions to be explainable and transparent, by recording the agent's beliefs, calculations and rule selections in an EBB data logger. The logical nature of the rational agent rules then allows a deductive argument to be reconstructed.

At the same time the rational agent can be extracted into a model and we can formally verify that its choices respect a specified code of ethics by exhaustive search over the choices it makes given particular beliefs and calculations.

We have implemented this technology in an experimental case study and shown that its ethical reasoning can be verified using a translation from the Python code into the AJPF model-checking system. Thus we have developed and demonstrated[12] what is believed to be the first formally verified ethical robot.

Appendix A
Open Data Statement

The program code and experimental data discussed in this paper are available as follows:

- The source code for *BDIPython* is available at https://github.com/VerifiableAutonomy/BDIPython where it is currently under active development. The version discussed here is archived at http://dx.doi.org/10.17638/datacat.liverpool.ac.uk/667.
- The source code for AJPF is available from http://mcapl.sourceforge.net where the work in this paper can be found in the `ethical_engine` branch of the git repository. The version discussed here is archived at http://dx.doi.org/10.17638/datacat.liverpool.ac.uk/667.
- The Python code for the Ethical NAO Robot is available at https://github.com/VerifiableAutonomy/EthicalNao where the work in this paper can be found in the `ethical_engine` branch of the git repository. The version discussed here is archived at http://dx.doi.org/10.17638/datacat.liverpool.ac.uk/667.
- Experimental data generated by the case study is available from the UWE research repository http://researchdata.uwe.ac.uk/375.

[12] In a simple laboratory test scenario.

Acknowledgements

The work of this paper is funded by EPSRC grants reference EP/L024845/1 and EP/L024861/1 within the project 'Verifiable Autonomy'. The authors are very grateful to the anonymous reviewers for their insightful comments – and the many improvements that have followed.

References

[1] M. M. Waldrop *et al.*, "No drivers required," *Nature*, vol. 518, no. 7537, pp. 20–20, 2015.

[2] M. Anderson and S. Anderson, "Machine Ethics: Creating an Ethical Intelligent Agent," *AI Magazine*, vol. 28, no. 4, pp. 15–26, 2007.

[3] L. Royakkers and R. van Est, "A Literature Review on New Robotics: Automation from Love to War," *International Journal of Social Robotics*, vol. 7, no. 5, pp. 549–570, 2015.

[4] A. Winfield, *Robotics: A very short introduction*. OUP Oxford, 2012.

[5] J. H. Moor, "The Nature, Importance, and Difficulty of Machine Ethics," *IEEE Intelligent Systems*, vol. 21, no. 4, pp. 18–21, Jul. 2006. [Online]. Available: http://dx.doi.org/10.1109/MIS.2006.80

[6] T. Arnold and M. Scheutz, "Against the Moral Turing test: Accountable Design and the Moral Reasoning of Autonomous Systems," *Ethics and Information Technology*, vol. 18, no. 2, pp. 103–115, Jun 2016.

[7] M. Fisher, L. A. Dennis, and M. Webster, "Verifying Autonomous Systems," *ACM Communications*, vol. 56, no. 9, pp. 84–93, 2013.

[8] R. Arkin, P. Ulam, and A. Wagner, "Moral Decision Making in Autonomous Systems: Enforcement, Moral Emotions, Dignity, Trust, and Deception," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 571–589, 2012.

[9] I. Asimov, "Runaround," in *Astounding Science Fiction*. Street & Smith, March 1942.

[10] A. F. T. Winfield, C. Blum, and W. Liu, "Towards an Ethical Robot: Internal Models, Consequences and Ethical Action Selection," in *Advances in Autonomous Robotics Systems*, ser. Lecture Notes in Computer Science, M. Mistry, A. Leonardis, M. Witkowski, and C. Melhuish, Eds., vol. 8717. Springer, 2014, pp. 85–96.

[11] L. A. Dennis, M. Fisher, N. K. Lincoln, A. Lisitsa, and S. M. Veres, "Practical verification of decision-making in agent-based autonomous systems," *Automated Software Engineering*, vol. 23, no. 3, pp. 305–359, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10515-014-0168-9

[12] W. Wallach and C. Allen, *Moral machines: Teaching robots right from wrong*. Oxford University Press, 2008.

[13] R. W. Picard and R. Picard, *Affective computing*. MIT press Cambridge, 1997, vol. 252.

[14] B. Deng, "The robot's dilemma," *Nature*, vol. 523, no. 7558, p. 24, 2015.

[15] B. F. Malle, M. Scheutz, T. Arnold, J. Voiklis, and C. Cusimano, "Sacrifice one for the good of many?: People apply different moral norms to human and robot agents," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '15. ACM, 2015, pp. 117–124. [Online]. Available: http://doi.acm.org/10.1145/2696454.2696458

[16] C. Allen, W. Wallach, and I. Smit, "Why machine ethics?" *IEEE Intelligent Systems*, vol. 21, no. 4, pp. 12–17, 2006.

[17] A. F. Winfield and V. V. Hafner, "Anticipation in robotics," in *Handbook of Anticipation: Theoretical and Applied Aspects of the Use of Future in Decision Making*, R. Poli, Ed. Cham: Springer International Publishing, 2018.

[18] R. Rosen, *Anticipatory systems: philosophical, mathematical, and methodological foundations*, ser. IFSR international series on systems science and engineering. Pergamon Press, 1985. [Online]. Available: https://books.google.co.uk/books?id=73VQAAAAMAAJ

[19] A. Isidori, D. L. Marconi, and D. A. Serrani, "Fundamentals of internal-model-based control theory," in *Robust Autonomous Guidance*. Springer, 2003, pp. 1–58.

[20] O. Holland, *Machine consciousness*. Imprint Academic, 2003.

[21] J. H. Holland, "Complex adaptive systems," *Daedalus*, pp. 17–30, 1992.

[22] A. F. Winfield, "Experiments in artificial theory of mind: From safety to story-telling," *Front. Robot. AI*, vol. 5, no. 75, 2018.

[23] P. Carruthers and P. K. Smith, *Theories of theories of mind*. Cambridge University Press, 1996.

[24] V. Gallese and A. Goldman, "Mirror neurons and the simulation theory of mind-reading," *Trends in cognitive sciences*, vol. 2, no. 12, pp. 493–501, 1998.

[25] A. Winfield and M. Jirotka, "The case for an ethical black box," in *Towards Autonomous Robotic Systems*. Springer, 2017.

[26] M. Wooldridge, *An introduction to MultiAgent Systems*. John Wiley and Sons, LTD, 2002.

[27] M. E. Bratman, *Intentions, Plans, and Practical Reason*. Harvard University Press, 1987.

[28] A. S. Rao and M. P. Georgeff, "Modeling Agents within a BDI-Architecture," in *Proceedings 2nd International Conference Principles of Knowledge Representation and Reasoning (KR&R)*. Morgan Kaufmann, 1991, pp. 473–484.

[29] ——, "An Abstract Architecture for Rational Agents," in *Proceedings International Conference Knowledge Representation and Reasoning (KR&R)*. Morgan Kaufmann, 1992, pp. 439–449.

[30] ——, "BDI Agents: From Theory to Practice," in *Proceedings 1st International Conference Multi-Agent Systems (ICMAS)*, San Francisco, USA, 1995, pp. 312–319.

[31] A. Rao, "AgentSpeak(L): BDI Agents Speak Out in a Logical Computable Language," in *Agents Breaking Away: Proceedings 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, ser. LNCS, vol. 1038. Springer, 1996, pp. 42–55.

[32] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-agent Systems in AgentSpeak Using Jason*. Wiley, 2007.

[33] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Meyer, "Agent Programming in 3APL," *Autonomous Agents and Multi-Agent Systems*, vol. 2, no. 4, pp. 357–401, 1999.

[34] A. Pokahr, L. Braubach, and W. Lamersdorf, "Jadex: A BDI Reasoning Engine," R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, Eds. Springer, 2005, pp. 149–174.

[35] M. Sierhuis, "Modeling and Simulating Work Pratice. BRAHMS: a Multiagent Modeling and Simluation Language for Work System Analysis and Design," Ph.D. dissertation, Social Science and Informatics (SW), University of Amsterdam, 2001.

[36] K. V. Hindriks, F. S. de Boer, W. van der Hoek, and J.-J. Meyer, "Agent Programming with Declarative Goals," in *Intelligent Agents VII (Proceedings 6th Workshop on Agent Theories, Architectures, and Languages)*, ser. LNAI, vol. 1986. Springer, 2001, pp. 228–243.

[37] L. A. Dennis, "Gwendolen semantics: 2017," University of Liverpool, Department of Computer Science, Tech. Rep. ULCS-17-001, 2017.

[38] N. Lincoln, S. M. Veres, L. A. Dennis, M. Fisher, and 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.

[39] J. H. Fetzer, "Program Verification: The Very Idea," *ACM Communications*, vol. 31, no. 9, pp. 1048–1063, 1988.

[40] R. A. DeMillo, R. J. Lipton, and A. J. Perlis, "Social Processes and Proofs of Theorems of Programs," *ACM Communications*, vol. 22, no. 5, pp. 271–280, 1979.

[41] R. S. Boyer and J. S. Moore, Eds., *The Correctness Problem in Computer Science*. London: Academic Press, 1981.

[42] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 1999.

[43] M. Webster, M. Fisher, N. Cameron, and 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*, ser. Lecture Notes in Computer Science, vol. 6894. Springer, 2011, pp. 228–242.

[44] M. Kamali, L. A. Dennis, O. McAree, M. Fisher, and S. M. Veres, "Formal verification of autonomous vehicle platooning," *Science of Computer Programming*, pp. –, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167642317301168

[45] L. Dennis, M. Fisher, M. Slavkovik, and M. Webster, "Formal verification of ethical choices in autonomous systems," *Robotics and Autonomous Systems*, pp. –, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889015003000

[46] D. Kortenkamp and R. Simmons, "Robotic systems architectures and programming," in *Springer Handbook of Robotics*. Springer, 2008, pp. 187–206.

[47] M. M. Botvinick, "Hierarchical models of behavior and prefrontal function," *Trends in cognitive sciences*, vol. 12, no. 5, pp. 201–208, 2008.

[48] R. C. Arkin, "Governing lethal behavior: embedding ethics in a hybrid deliberative/reactive robot architecture," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 2008, pp. 121–128.

[49] D. Vanderelst and A. Winfield, "An architecture for ethical robots inspired by the simulation theory of cognition," *Cognitive Systems Research*, vol. 48, pp. 56–66, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389041716302005

[50] R. Murphy and D. D. Woods, "Beyond asimov: the three laws of responsible robotics," *IEEE Intelligent Systems*, vol. 24, no. 4, 2009.

[51] M. Donoso, A. G. Collins, and E. Koechlin, "Foundations of human reasoning in the prefrontal cortex," *Science*, vol. 344, no. 6191, pp. 1481–1486, 2014.

[52] L. A. Dennis, M. Fisher, M. Webster, and R. H. Bordini, "Model Checking Agent Programming Languages," *Automated Software Engineering*, vol. 19, no. 1, pp. 5–63, 2012.

[53] L. A. Dennis, B. Farwer, R. H. Bordini, M. Fisher, and M. Wooldridge, "A Common Semantic Basis for BDI Languages," in *Proceedings 7th International Workshop on Programming Multiagent Systems (ProMAS)*, ser. LNAI. Springer, 2008, vol. 4908, pp. 124–139.

[54] W. Visser, K. Havelund, G. P. Brat, S. Park, and F. Lerda, "Model Checking Programs," *Automated Software Engineering*, vol. 10, no. 2, pp. 203–232, 2003.

[55] J. Juneau, J. Baker, V. Ng, L. Soto, and F. Wierzbicki, *The Definitive Guide to Jython: Python for the Java Platform*. Springer, 2010.

[56] J. G. Politz, A. Martinez, M. Milano, S. Warren, D. Patterson, J. Li, A. Chitipothu, and S. Krishnamurthi, "Python: The full monty: A tested semantics for the python programming language," 2013.

[57] G. J. Smeding, "An executable operational semantics for python." Master's thesis, 2009.

[58] A. Ferrando, L. A. Dennis, D. Ancona, M. Fisher, and V. Mascardi, "Recognising assumption violations in autonomous systems verification," 2017, under Review.

[59] D. Ancona, A. Ferrando, and V. Mascardi, *Theory and Practice of Formal Methods: Essays Dedicated to Frank de Boer on the Occasion of His 60th Birthday*. Cham: Springer International Publishing, 2016, ch. Comparing Trace Expressions and Linear Temporal Logic for

Runtime Verification, pp. 47–64. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-30734-3_6

[60] R. Vaughan and M. Zuluaga, "Use your illusion: Sensorimotor self-simulation allows complex agents to plan with incomplete self-knowledge," in *International Conference on Simulation of Adaptive Behavior*. Springer, 2006, pp. 298–309.

[61] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

[62] J. C. Zagal, J. Delpiano, and J. Ruiz-del Solar, "Self-modeling in humanoid soccer robots," *Robotics and Autonomous Systems*, vol. 57, no. 8, pp. 819–827, 2009.

[63] M. Anderson and S. Anderson, "Robot be good," *Scientific American Magazine*, vol. 303, no. 4, pp. 72–77, 2010.

[64] S. Bringsjord, N. S. G., D. Thero, and M. Si, "Akratic robots and the computational logic thereof," in *Proceedings of the IEEE 2014 International Symposium on Ethics in Engineering, Science, and Technology*, ser. ETHICS '14. Piscataway, NJ, USA: IEEE Press, 2014, pp. 7:1–7:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=2960587.2960596

[65] M. Anderson and S. L. Anderson, "Geneth: A general ethical dilemma analyzer," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, ser. AAAI'14. AAAI Press, 2014, pp. 253–261. [Online]. Available: http://dl.acm.org/citation.cfm?id=2893873.2893915

[66] L. A. Dennis, M. Fisher, and A. F. T. Winfield, "Towards Verifiably Ethical Robot Behaviour," in *AAAI Workshop on AI and Ethics (1st International Conference on AI and Ethics)*, Austin, TX, January 2015.

[67] A. Mackworth, "Architectures and ethics for robots," in *Machine ethics*, M. Anderson and S. L. Anderson, Eds. Cambridge University Press, 2011, pp. 204–221.

[68] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: Probabilistic Symbolic Model Checker," in *Proceedings 12th International Conference Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS)*, ser. LNCS, vol. 2324. Springer, 2002.

[69] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

Paul Bremner received a BSc in Robotic and Electronic Systems Engineering from the University of Salford in 2003. He received an MSc in Advanced Technologies in Electronics and a PhD in Human-Robot interaction from the University of the West of England in 2005 and 2010 respectively.

Since 2010 he has worked at the University of the West of England on a number of projects, first as a research associate then as a research fellow, where he is currently employed on the Verifiable Autonomy project. His research interests include human-robot interaction, multi-modal communication, tele-presence, artificial intelligence and robot ethics.

Louise A. Dennis received a B.A. in mathematics and philosophy from the University of Oxford in 1992, and an MSc in knowledge-based systems and a PhD in artificial intelligence from the University of Edinburgh in 1994 and 2001 respectively.

She has worked as a research associate at the Universities of Glasgow and Edinburgh and a lecturer at the University of Nottingham. Since 2006 she has been a research associate at the University of Liverpool where she is currently employed on the Verifiable Autonomy project. Her research interests are autonomous systems, formal verification, BDI agent programming languages, automated reasoning and ethical machine reasoning.

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

Michael Fisher is Professor of Computer Science and Director of the multi-disciplinary Centre for Autonomous Systems Technology at the University of Liverpool. He is a member of the British Standards Institution AMT/10 committee on "Robotics", authored *An Introduction to Practical Formal Methods using Temporal Logic* (Wiley) in 2011, is on the editorial boards of both Applied Logic and Annals of Mathematics and Artificial Intelligence journals and is a corner editor for the Journal of Logic and Computation. His research interests mainly involve formal verification for the certification, safety, ethics, and reliability of autonomous systems, and he leads the *UK Network on the Verification and Validation of Autonomous Systems* (`vavas.org`).

Prof. Fisher is a member of the Embedding Values into Autonomous Intelligent Systems committee of the IEEE Global Initiative for Ethical Considerations in Artificial Intelligence and Autonomous Systems and a member of the working group for IEEE-P7009 on Failsafe Mechanisms for Autonomous Systems.

Alan Winfield is Professor of Robot Ethics at the University of the West of England (UWE), Bristol, UK, and Visiting Professor at the University of York. He received his PhD in Digital Communications from the University of Hull in 1984, then co-founded and led APD Communications Ltd until taking-up appointment at UWE, Bristol in 1992. Winfield co-founded the Bristol Robotics Laboratory where his research is focussed on cognitive robotics.

Winfield is an advocate for robot ethics; he was a member of the British Standards Institute working group that drafted BS 8611: Guide to the Ethical Design of Robots and Robotic Systems, and he is a member of the executive committee of the IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems.