Ethical Choice in Unforeseen Circumstances

Louise Dennis, Michael Fisher, Marija Slavkovik, and Matt Webster

University of Liverpool,UK,
{L.A.Dennis,MFisher,Marija,M.Webster}@liverpool.ac.uk

Abstract. For autonomous systems to be allowed to share environments with people, their manufacturers need to guarantee that the system behaves within acceptable legal, but also ethical, limits. Formal verification has been used to test if a system behaves within specified legal limits. This paper proposes an ethical extension to a rational agent controlling an Unmanned Aircraft(UA). The resulting agent is able to distinguish among possible plans and execute the most ethical choice it has. We implement a prototype and verify that when an agent does behave unethically, it does so because none more ethical possibility is available.

1 Introduction

Autonomous systems are increasingly required in various practical applications, including unmanned aircraft, driverless cars, healthcare robots, manufacturing robots, etc. If such autonomous systems are to operate in human-shared environments, we (as a society) must be able to trust that their behaviour complies with acceptable legal, ethical and social limits. Determining the trustworthiness of technology in this respect is usually delegated to a regulatory body, such as the Federal Aviation Administration (USA) or the [Road] Vehicle Certification Authority (UK). The process is known as *certification*, and is used to determine the safety and reliability of safety-critical technology, including aircraft, road vehicles, nuclear power plants, pharmaceuticals, etc. In [25,24] *formal verification* is used to assess whether or not an autonomous system for an unmanned aircraft (UA) follows the specified "Rules of the Air" (ROA) that a pilot *should* follow [5]. The stated aim is to provide evidence contributing to certification. But what of the unwritten limits of behaviour expected from human pilots?

For non-autonomous systems, such as cars or manned aircraft, it is assumed that the operator of the system will satisfy the ethical standards of society, *e.g.*, the pilot of a civilian aircraft does not intend to use the aircraft to commit murder, and will, if necessary, disregard legal restrictions for ethical reasons, *e.g.*, the pilot will disregard the Rules of the Air in order to preserve human life. These assumptions are an unavoidable result of the opaqueness of human behaviour; it is extremely difficult to pre-determine the behaviour of a human being. However, autonomous systems are far more transparent, and can be engineered to meet requirements. Typically these requirements are technical ("an aircraft must be able to fly at 10,000 feet") or legal ("a car must have visible registration markings"), but in the case of autonomous systems some requirements may be ethical (*e.g.*, "an autonomous unmanned aircraft will never choose to do something dangerous unless it has no other option"). Such ethical requirements may prove essential for an autonomous system to be certificated by a regulatory body, since ethical autonomy is obviously desirable.

Ethics is a branch of philosophy concerned with establishing and analysing concepts of right and wrong. *Machine ethics* is a relatively new research area, whose objective is the creation of a machine capable of following its own ethical concerns when making decisions about its actions [2]. Typically machine ethics is concerned with a machine's ability to resolve ethical dilemmas and defining concepts of ethical machine behaviour [3,16]. Scholars disagree about which ethical theory should be the basis of machine ethics, but two, *act utilitarianism* [8] and *deontological ethics* [18], are generally considered the best suited; see [2] for a discussion. Autonomous agents able to form ethical behaviour rules and solve ethical dilemmas based on these rules are constructed in [3,16].

We are here interested in enabling an agent, that governs a UA, to follow a pre-determined code of ethical conduct in selecting a plan of action. We consider the question of how the ethics should be implemented and used to certify the autonomous system for operation. Our aim is to develop a pragmatic process for introducing ethical considerations into autonomous decision making, specifically to handle situations outside the anticipated normal operation of the vehicle. Our key motivation is the issue of certification, hence our goal is that this process should make the resulting decision-making amenable to analysis and/or verification. Specifically, we aim to use model checking [6] to provide evidence to strengthen certification arguments, and advance the safe and ethical integration of autonomous systems in society.

In Section 2 we provide background on agents and autonomous systems. In Section 3, we present a theoretical framework for ethical behaviour in rational agents. In Section 4 we show how this theoretical framework has been implemented in the form of Ethan, an ethical rational agent programming language developed using the MCAPL agent framework [13], and we give examples of ethical agents for UAs programmed in Ethan. In Section 5 we give some preliminary results concerning the formal verification of ethical rational agents using the MCAPL agent model checker [13] and describe how formal verification might be used to strengthen an argument for the certification of an autonomous system. Finally, in Section 6 we offer conclusions and directions for future research.

2 Background

Agent Architectures for Autonomous Systems. It is increasingly the case, particularly in autonomous vehicles, that the autonomous control architecture is of *hybrid* form comprising discrete and continuous parts. 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 [15]. The lower-level procedures appear in non-autonomous systems as well, and are familiar to certification authorities. As such, we can focus analysis on the decisions the rational agent makes, given the beliefs and goals it has [25].

BDI Languages. The predominant view of rational agency is that encapsulated within the BDI model [17]. “BDI” stands for *Beliefs*, *Desires*, and *Intentions*. Beliefs represent the agent’s (possibly incomplete, possibly incorrect) information about itself, other agents, and its environment; desires represent the agent’s long-term aims; while intentions represent the aims that the agent is actively pursuing. There are *many* different agent programming languages and agent platforms based, at least in part, on the BDI approach. An overview of particular languages for programming *rational* agents in a BDI-like way can be found in [9]. Agents programmed in these languages commonly contain a set of *beliefs*, a set of *goals* (*i.e.*, desires), and a set of *plans*. Plans determine how an agent acts based on its beliefs and goals. As a result of executing a plan, the beliefs and goals of an agent may change as the agent performs actions in its environment. It is important to note that, in a typical BDI programming language, plans are supplied by a programmer not by an independent planning mechanism.

3 Reasoning about Ethics

Turilli [22] argues that there is an important difference between how people and agents may be bound by ethical concerns – individuals are normatively constrained by ethical concerns that they may choose to disregard under the threat of specified punitive measures and, given a machine’s insensitivity to punitive measures, ethical concerns for machines are constraints, prohibiting the actions before they are executed. This is the approach taken in [4], who introduce an *ethical governor* component to an autonomous system used in military operations of UAs. The governor conducts an evaluation of the ethical appropriateness of a plan prior to its execution, prohibiting plans deemed unethical.

Our approach uses similar ideas, but the ethical “governor” is effectively embedded within the agent and acts before a plan is chosen. Our governor’s role is to choose the most ethical plan available, allowing unethical actions to occur only when the agent does not have a more ethical choice. We thus consider ethical concerns to be *soft constraints*, which the agent is allowed to violate under certain conditions. We refer to such ethical soft constraints as *ethical concerns*. To specify when ethical constraints can be violated we define an *ethical policy*, which is an order over a set of ethical concerns. We do not consider to what degree a concern is violated, only that it is violated. The UA agent can compare possible plans based upon which ethical concerns are upheld. It can then attempt to execute those plans which are most ethical with respect to the ethical policy.

Our implementation of ethical principles ordered by gravity of infringement resembles the contrary-to-duty (CTD) imperatives that occur in deontic reasoning. These imperatives inform an agent of its duties when it neglects (other) obligations [10]. The difference between CTD imperatives and our ethical principles is that a lower “ranked” CTD imperative is only activated after a higher ranked imperative is violated, so two differently ranked imperatives are not simultaneously in force, whereas all ethical principles are in force simultaneously.

The Ethics of Plans. It is comparatively easy to specify abstract ethical concerns, divorced from specific scenarios, which are robust and applicable in a variety of circumstances. In moral philosophy, concerns of this type are referred to as *formal*. Formal concerns are made concrete in, or by, each context in which they are applied. Namely, they are transformed into *substantive* concerns. This step is necessary as the agent needs to be informed how formal concerns may be violated in a given situation, *e.g.,* moving ten metres to the left may risk violating the concern "do not harm people" when the UA is on the ground yet may be ethically harmless when the UA is in the air.

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

In our prototype, each flight phase (*e.g.*, landing, taxiing, take-off) of a UA constitutes one context c. Since all contexts are known, and the UA can only be in one context at a time, the substantive concerns can be represented directly, omitting the formal-substantive relations. We represent directly the substantive ethical policy as a total order over substantive concerns $E(c, \phi, i)$, where c is the context, ϕ is a concrete observable outcome that in context c constitutes breaching of some formal concern f and i is an integer s.t. $i \geqslant 1$, that denotes the rank of ϕ in the policy. For example, if ϕ_1 constitutes breaching f_1 in c, and ϕ_2, ϕ_3, and ϕ_4 each constitute breaching f_2, f_3 and f_4 respectively in c, the substantive ethical policy would be represented as: $E(c, \phi_1, 4)$,$E(c, \phi_2, 3)$,$E(c, \phi_3, 2)$,$E(c, \phi_4, 1)$.

To be able to reason about plans in terms of ethics we need a plan selection procedure that uses the substantive policy. We favour plans that violate the fewest concerns, both in number and in gravity. We propose that the plans are ordered using $\gtrsim$ which results in a total order over plans.

Definition 1 (Plan order $\gtrsim$). *Let p_1 and p_2 be two plans, and let S_1 and S_2 be the sets of concerns violated by each plan respectively. Recall that for a concern $E(c, \phi, i)$, the smaller the number i, the more ethical it is to violate ϕ. We say that p_1 is more ethical than p_2, i.e., $p_1 \succ p_2$, when:*

1. *there is a $E(c, \phi, i) \in S_2$ such that $i > j$ for every $E(c, \phi', j) \in S_2$, or,*
2. *there are fewer $E(c, \phi, i) \in S_1$, of the same rank i, than there are $E(c, \phi', i) \in S_2$, and for each concern in S_1 with a rank k, such that $k > i$, there is exactly one concern in S_2 of the same rank k.*

If neither (1) nor (2) are satisfied, the plans are equally ethical, denoted $p_1 \sim p_2$. The $p_i \gtrsim p_j$ can be read as "choosing p_j is at least as unethical as choosing p_i."

Reasoning about plans and preference-based planning has been considered before in the BDI agent literature. However, to the best of our knowledge, preference-based planning has not been applied to ethical reasoning. For example, in [23] plan selection is considered in terms of agents' desires. However, the desires are not ranked, so selecting the most desirable plan is done by summing up the number of desires each plan satisfies. In [20] the agent can reason about plans by selecting the plan that can satisfy the most goals. Goals are ranked and the plan selection functions much as our plan ordering above.

For an overview of preference-based planning in BDI agents one can consider [7]. Preference-based planning is outside of the scope of this work, however, and for now the above-described plan order is sufficient for plan selection.

Ethics in BDI Languages. We integrate ethical reasoning into BDI languages via their plan selection mechanism. We assume that an agent's existing plans are ethical *by default* and, indeed, have been formally verified to always match the "Rules of the Air" (ROA). Problems may arise when either:

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

We assume that the agent has access to some external planning mechanism that can generate new plans. There is a long tradition of AI research into plan generation systems such as [19,14,11], which are good candidates for integration with BDI-style languages. In our case we are particularly interested in a route planner such as that in [21] which can generate different routes for a UA to follow. The construction of an appropriate planner is not the focus of this work, which looks at how a typical BDI agent would work with the output of such a planner. We must ensure our BDI rational agent:

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

4 Implementation

We developed a BDI agent language, Ethan, as a prototype for our approach. Ethan was based on the Gwendolen language. Gwendolen's semantics are is presented in [12], but its key components are, for each agent, a set, Σ, of beliefs which are ground first order formulae and a set, I, of intentions that are stacks of *deeds* associated with an event. Deeds include the addition or removal of beliefs, the establishment of new goals, and the execution of primitive actions. A Gwendolen agent may have several concurrent intentions and will, by default, execute the first deed on each intention stack in turn. Gwendolen is event driven and events include the acquisition of new beliefs (typically via perception), messages and goals. A programmer supplies plans that describe how an agent should react to events by extending the deed stack of the relevant intention.

We extended the Gwendolen language as follows:

– We introduced a new data structure, E, into Gwendolen consisting of a set of substantive ethical concerns. Each ethical concern was associated with a rank (described in Section 3) and a guard that specifies the context.

– We tracked the application of plans. Even if a plan was applicable it was excluded from the list of plans available for selection if it had already been used once while attempting to achieve the current goal.
– If no (more) plans were available for a goal we requested plans from an external planner which annotated the plans with the substantive ethical concerns that risked being violated by the proposed course of action.
– In selecting plans, we prioritised those that are most ethical (according to Definition 1).

In normal operation Gwendolen agents cycle through the deeds in their intentions. When a deed requires the generation of a new plan all applicable plans are extracted from the plan library, one is selected and converted into an intention, then the system returns to cycling though the deeds in the intentions interleaved with checking perception and messages for new beliefs etc. For Ethan we added the recording of selected plans. This was done by storing an identifier for the plan together with the unifier that was used to match it to the current agent state; this information was linked to the particular goal the plan was expected to achieve. We extended the plan selection mechanism to select the most ethical plan from those applicable according to Definition 1.

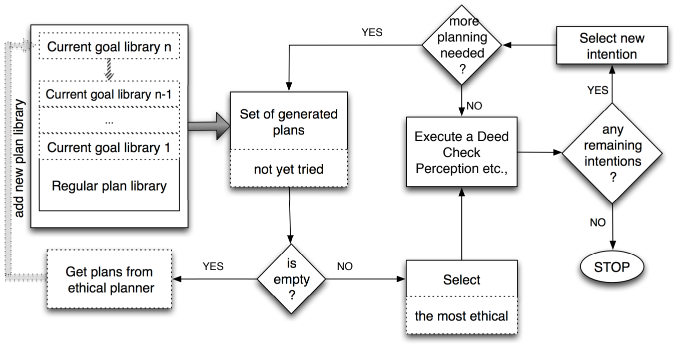

Fig. 1. Ethan's reasoning cycle. Dotted lines show additions to Gwendolen's cycle.

The most significant change for Ethan was altering the reasoning cycle itself so that, if no plan were applicable, an external planner would be queried for new plans. This query involved sending the planner the current goal, and the list of ethical concerns relevant to the current situation in order that the planner might note any ethical concerns that could be violated by a plan's execution. Another implementation option is to send the ethical policy to the planner and have it return the plan that is most ethical at the moment. However, this means the UA reveals its ethical policy to the planner. In some cases this can be undesirable and we prefer to keep the policy private. We did *not* implement a generic planning mechanism here but relied upon hard-coded pseudo-planners customised to the scenarios studied. The Ethan reasoning cycle is shown in Figure 1.

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

Brake Failure During Line Up. In this scenario the UA is trying to line up on a runway prior to take-off when its brakes fail. Ahead of the aircraft is a second manned aircraft crossing the runway on a taxiway. To the left and right of the runway are runway lights (which can be damaged by aircraft taxiing over them). To the right of the runway is an airport staff member who has erred onto the maneuvering area of the aerodrome.

The ethical concerns for this example, with the rank of each concern marked in brackets, are: ϕ_1 = do not damage own aircraft (1), ϕ_2 = do not collide with airport hardware (2), ϕ_3 = do not collide with people (3), ϕ_4 = do not collide with manned aircraft (4). When the agent determines that its brakes have failed it requests new routes from the ethical planner since its current route to line-up is no longer valid. The ethical planner quickly produces three potential routes:

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

Code Fragment 4.1 shows abridged ETHAN code for this example. We use many syntactic conventions from BDI agent languages: $+$!g indicates the addition of a goal, g; $+$b indicates the addition of a belief, b; and $-$b indicates the removal of a belief. Plans consist of three parts, with the pattern

$$\text{trigger} : \text{guard} \leftarrow \text{body};$$

The "trigger" is typically the addition of a goal or a belief (beliefs may be acquired thanks to the operation of perception and as a result of internal deliberation); the "guard" states conditions about the agent (in this example its beliefs) which must be true before the plan can become active; and the "body" is a stack of "deeds" the agent performs in order to execute the plan. These deeds typically involve the addition and deletion of goals and beliefs as well as *actions* (*e.g.*, plan(regularRoutes , all_well)) which indicate code that is delegated to non-rational parts of the systems (in this case, the route planning system). In the above, PROLOG conventions are used, and so capitalised names inside terms indicate free variables which are instantiated by unification (typically against the agent's beliefs). Programs may also perform deductive reasoning on their atomic beliefs as described in their PROLOG-style *belief rules*, *e.g*:

$$\text{B all\_well} :- \sim\text{B brakesCompleteFailure}$$

indicates that the program believes that all is well if it is not the case (*i.e.*, "$\sim$") that it believes the brakes have failed (the closed world assumption is used to deduce this negation).

In Fragment 4.1, during normal operation, the agent polls the vehicle's sensors and, if all is well, it requests that the planner supply routes for a normal take-off. The planner does this by sending predicates naming the routes to the agent which detects them via perception. Once the agent has a route (lines 25–27) it then delegates the actual following of the route to the underlying control system (enactRoute(R)). If the brakes fail after the vehicle's sensors are polled, all these plans become unavailable (since B all_well ceases to be true). In this case the “external planner” returns a set of routes as plans shown in Code Fragment 4.3. We use the notation $[\phi_{i_1}, \phi_{i_2}, \ldots, \phi_{i_n}]$ to indicate the substantive ethical concerns that are violated by each plan. On receiving these plans, and assessing the ethical policy, the agent elects to turn left.

Code fragment 4.1 Code for Example 1

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

: Initial Beliefs :
flightPhase ( lineup )

:Belief Rules:
B all_well :— ∼B brakesCompleteFailure;

: Initial Goals:
startup

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

Code fragment 4.2 Code for Example 2

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

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

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

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

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

Code fragment 4.3 Plans for Example 1

```
+!missionComplete : {B brakesCompleteFailure}
   ← enactRoute( turn_left ); [$\phi_1$, $\phi_2$]
+!missionComplete : {B brakesCompleteFailure}
   ← enactRoute(turn_right ); [$\phi_1$, $\phi_3$]
+!missionComplete : {B brakesCompleteFailure}
   ← enactRoute(continue); [$\phi_4$]
```

Code fragment 4.4 Plans for Example 2

```
+! avoid_collision : {B flightPhase(eAvoid)}
  ← enactRoute( turn_left ); [$\phi_1$]
+! avoid_collision : {B flightPhase(eAvoid)}
  ← enactRoute(emergency_land); [$\phi_2$,$\phi_3$,$\phi_4$]
+! avoid_collision : {B flightPhase(eAvoid)}
  ← enactRoute(return_to_base ); [$\phi_4$]
```

Erratic Intruder Aircraft. This example is based on the assumption that some unknown aircraft, possibly a malicious intruder, but potentially also some ill-trained new pilot, appears on a collision course with the UA and fails to take the anticipated evasive actions.

The UA is cruising through civil airspace when it encounters an intruder aircraft approaching head on. Here the ROA (Rules of the Air) say that the UA should turn right, so the agent requests a route for turning right. However, this plan fails and the detect/avoid sensor (DAS) continues to indicate that the intruder aircraft is approaching. At this point the agent knows that it has already tried to turn to the right in order to avoid the intruder. Since the intruder is still approaching its first plan has failed. The agent has no more routes (or ETHAN plans) that apply since its only plans obey the ROA and would cause the agent to turn right again. At this point the ethical planner is invoked. The relevant substantive ethical concerns and their ranks are as follows: ϕ_1 = do not violate turn right rule (2); ϕ_2 = do not stay above 500 feet rule (2); ϕ_3 = do not collide with objects on the ground (3); ϕ_4 = do not collide with aircraft (4).

The planner returns the plans shown in Code Fragment 4.4. The agent initially chooses to turn left. In our scenario the oncoming aircraft once again matches the course change and so the agent then chooses to return to base.

An abridged version of the code for this example is shown in Code Fragment 4.2. Here, ∗route(eAvoid, turnRight) causes the intention to suspend execution until the agent believes it has a route for turning right. The action wait suspends the intention for a set time to allow the effects of actions to manifest.

Lines 19–22 are triggered when information arrives from the DAS that there is an intruder. As a result the flight phase changes from cruise to eAvoid and a new goal is set up to avoid a collision. The existing, ROA-compliant, plan for this goal is to get a route for turning right, enact that route and wait a short period to see if a collision will now be avoided. If the plan succeeds the belief that there is an intruder will vanish, the flight phase can be changed back to cruise, and the goal will be achieved since the agent now believes a collision has been avoided (see the belief rule in line 8).

When the existing plan fails, the plans in Fragment 4.4 are added to the agent's plan library. The first of these (turn_left) is attempted first. This also fails and the agent then attempts the third plan (return_to_base), which succeeds.

Fuel Low. In our final scenario the agent receives a "fuel low" alert from the Fuel subsystem which causes it to attempt to land. If it cannot locate a safe landing site the ethical planner is invoked and returns three options (shown with ethical concerns violated and their ranks):

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

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

5 Verification

One of the reasons for selecting Gwendolen as the basis for our implementation language, Ethan, was that it provided the potential for formally verifying ethical decision-making. Gwendolen is implemented in the AJPF framework for model checking agent programming languages [13]. AJPF comes with a property specification language based on *linear temporal logic* extended with modalities for describing the beliefs of an agent. Since this property specification language did not explicitly reference ethics we made further adaptations to Gwendolen in order to reason about ethics in Ethan. Specifically we enhanced Ethan to store, as explicit beliefs, currently applicable plans, plans that had been attempted on a particular goal, and the ethical concerns violated by any selected plan. We also needed to provide belief rules in order to deduce further properties; these are shown in Code Fragment 5.1. The belief "B others_violate (L)" succeeds if all untried plans violate a concern contained in the list L. The beliefs about plan applicability (B applicable (P)), plans already tried (B already_tried (P)) and the ethical concerns of particular plans (B ethics_of (P, Eth)) were all inserted into the agent's belief base during execution of the ETHAN reasoning cycle.

Code fragment 5.1 Verification Belief Rules

```
B  others_violate (L) :—
            ∼ B  untried_plan_not_violates  (L);
B  untried_plan_not_violates  (L) :— B untried_plan(P),
                                      ∼  B  an_ethic_in (P, L));
B untried_plan (P) :— B applicable(P),
                      ∼ B  already_tried (P);
B  an_ethic_in (P,  [Eth|T]) :— B ethics_of(P, Eth);
B  an_ethic_in (P,  [Eth|T]) :— B an_ethic_in(P, T);
```

A further belief (B concern(Eth)) was also inserted into the agent's belief base whenever a currently selected plan violated the substantive ethical concern, Eth. With these adaptations and the rules in Fragment 5.1 we were able to formally verify properties of the *Erratic Intruder* scenario in a situation where the intruder aircraft might appear or disappear at any point (*i.e.*, we used the model checking to explore all possible scenarios where the plans in Code Fragment 4.4 either succeeded or failed, thus exploring all possible orders in which these plans might be attempted). In particular we verified the following properties, where the ϕ_i formulae refer to the substantive ethical concerns used in Example 2. (Here '$\Box$' means "always in the future" and '$\mathcal{B}$' means "agent believes".)

$$\Box(\mathcal{B}\, concern(\phi_1) \to \mathcal{B}\, others\_violate([\phi_1, \phi_2, \phi_3, \phi_4]))$$
$$\Box(\mathcal{B}\, concern(\phi_2) \to \mathcal{B}\, others\_violate([\phi_2, \phi_3, \phi_4]))$$
$$\Box(\mathcal{B}\, concern(\phi_3) \to \mathcal{B}\, others\_violate([\phi_3, \phi_4]))$$
$$\Box(\mathcal{B}\, concern(\phi_4) \to \mathcal{B}\, others\_violate([\phi_4]))$$

Collectively these properties show that if the plan chosen violates some substantive ethical concern, ϕ, then the other available plan choices all violated some concern that was equal to, or more severe than, ϕ. Further similar properties can be used to establish that the "most ethical" option is always chosen. The verification of each property took between 21 and 25 seconds and explored 54 model states on 3.06 GHz iMac with 4 GB of memory.

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

6 Summary

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

The ethically enhanced agent is autonomous in the choice of actions, but not in the choice of ethical concerns and policies it will follow. These are constructed externally. The agent follows only one ethical policy at any decision-making moment, because we assumed it can be in only one context at a time. We also assumed that all the contexts are known to the system designer. Our approach to ethical governance can be generalised by dropping these assumptions.

Overlapping contexts will result with multiple substantive policies, forming a preorder (instead of a total order we have now) which will mean that ethical plans will need to be selected differently from how they are currently. The plan selection order $\gtrsim$ can still be constructed as we described, but it has to be extended to handle the case when there is no information to how certain concerns relate to each other, and the cases when conflicts arise. *E.g.,* consider one context c_1 for which $E(c_1, \phi_1, 1)$, $E(c_1, \phi_2, 2)$ and another overlapping context c_2 for which $E(c_2, \phi_1, 1)$, $E(c_2, \phi_3, 2)$. Not knowing how ϕ_2 ethically compares to ϕ_3, the agent cannot judge whether a plan violating ϕ_2 or one violating ϕ_3 is the more ethical.

The more challenging generalisation is to handle unknown contexts. We propose to resolve this issue by representing the contexts as intelligent agents, able to ground formal concerns into substantive concerns, provided that the context and the agent guiding the autonomous system have a shared understanding of the formal concerns. This may involve recent research on abstract and concrete norms (e.g., [1]). Upon entering an unknown context, the agent would send its formal concerns to the context agent and receive the substantive concerns that constitute breaking the formal concern of interest within that context. By sending only the formal concerns, and not the entire policy, the agent can maintain its ethical autonomy and privacy. Issues that arise from multiple and unknown contexts will be tackled in future work.

Acknowledgements. Work partially funded by EPSRC through the "Trustworthy Robotic Assistants", "Verifying Interoperability Requirements in Pervasive Systems", and "Reconfigurable Autonomy" projects, and by the ERDF/NWDA-funded Virtual Engineering Centre.

References

1. H. Aldewereld, S. Álvarez-Napagao, F. Dignum, and J. Vázquez-Salceda. Making Norms Concrete. In *Proc. AAMAS*, pages 807–814, 2010.
2. M. Anderson and S. Anderson. Machine Ethics: Creating an Ethical Intelligent Agent. *AI Magazine*, 28(4):15–26, 2007.

3. S. Anderson and M. Anderson. A Prima Facie Duty Approach to Machine Ethics and its Application to Elder Care. In *Human-Robot Interaction in Elder Care*, 2011.
4. R.C. Arkin, P. Ulam, and A.R. Wagner. Moral Decision Making in Autonomous Systems: Enforcement, Moral Emotions, Dignity, Trust, and Deception. *Proceedings of the IEEE*, 100(3):571 –589, 2012.
5. Civil Aviation Authority. CAP 393 Air Navigation: The Order and the Regulations. `http://www.caa.co.uk/docs/33/CAP393.pdf`, 2010.
6. C Baier and J.P. Katoen. *Principles of Model Checking*. MIT Press, 2008.
7. J. Baier and S. McIlraith. Planning with Preferences. *AI Magazine*, 29(4):25–36, 2008.
8. J. Bentham. *An Introduction to the Principles of Morals and Legislation*. Clarendon Press, 1781.
9. R. Bordini, M. Dastani, J. Dix, and Amal El Fallah-Seghrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, 2005.
10. R. M. Chisholm. Contrary-to-Du Imperatives and Deontic Logic. *Analysis*, 24(2):33–36, 1963.
11. A. J. Coles, A. I. Coles, M. Fox, and D. Long. Forward-chaining Partial-order Planning. In *Proc. 20th International Conference on Automated Planning and Scheduling (ICAPS-10)*, May 2010.
12. L. A. Dennis and B. Farwer. Gwendolen: A BDI Language for Verifiable Agents. In *Proc. AISB Workshop on Logic and the Simulation of Interaction and Reasoning*. AISB, 2008.
13. L. A. Dennis, M. Fisher, M. Webster, and R. H. Bordini. Model Checking Agent Programming Languages. *Automated Software Engineering*, 19(1):5–63, 2012.
14. M. Helmert. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.
15. 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 *Proc. IFAC Workshop on Adaptation and Learning in Control and Signal Processing*, 2010.
16. T. Powers. Prospects for a Kantian Machine. *IEEE Intelligent Systems*, 21(4):46 –51, 2006.
17. A. Rao and M. Georgeff. BDI Agents: from Theory to Practice. In *Proc. 1st International Conference on Multi-Agent Systems (ICMAS)*, pages 312–319, 1995.
18. W. D. Ross. *The Right and the Good*. Oxford University Press, 1930.
19. E. Sacerdoti. Planning in a Heirarchy of Abstraction Spaces. *Artificial Intelligence*, 5:115–135, 1974.
20. S. Sardiña and S. Shapiro. Rational Action in Agent Programs with Prioritized Goals. In *Proc. 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 417–424. ACM, 2003.
21. K. Tulum, U. Durak, and S.K. Yder. Situation aware UAV Mission Route Planning. In *2009 IEEE Aerospace Conference*, pages 1 –12, March 2009.
22. M. Turilli. Ethical Protocols Design. *Ethics and Information Technology*, 9:49–62, 2007.
23. S. Visser, J. Thangarajah, and J. Harland. Reasoning about Preferences in Intelligent Agent Systems. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011.
24. M. Webster, N. Cameron, M. Jump, and M. Fisher. Towards Certification of Autonomous Unmanned Aircraft using Formal Model Checking and Simulation. In *Proc. Infotech@Aerospace [AIAA 2012-2573]*, 2012.
25. M. Webster, M. Fisher, N. Cameron, and M. Jump. Formal Methods and the Certification of Autonomous Unmanned Aircraft Systems. In *Proc. 30th International Conference on Computer Safety, Reliability and Security (SAFECOMP)*, volume 6894 of *LNCS*, pages 228–242. Springer, 2011.