Specifying Agent Ethics (Blue Sky Ideas)

Louise A. Dennis[1[0000−1111−2222−3333]] and Michael Fisher[1[0000−0002−0875−3862]]

Department of Computer Science, University of Manchester, UK.
`louise.dennis@manchester.ac.uk`

Abstract. We consider the question of what properties a Machine Ethics system should have. This question is complicated by the existence of ethical dilemmas with no agreed upon solution. We provide an example to motivate why we do not believe falling back on the elicitation of values from stakeholders is sufficient to guarantee correctness of such systems. We go on to define two broad categories of ethical property that have arisen in our own work and present a challenge to the community to approach this question in a more systematic way.

Keywords: Machine Ethics · Formal Verification · Specifying Ethics.

1 Introduction

Machine Ethics considers the problem of ethical reasoning by computational systems. We have been working in this area for some years. In our earliest work [10], we considered the question of producing ethical reasoning in a verifiable fashion and we have continued to approach the question of Machine Ethics through a similar lens (e.g., [5,9]). However we have not yet made any systematic attempt to consider what it means to verify ethical reasoning more broadly, or to categorise the properties that ethical reasoners should exhibit. This is challenging since, of course, there are many well-known ethical dilemmas which have no agreed "correct" answer; this makes centering verification attempts upon whether the ethical reasoning system can generate the correct answer particularly challenging.

This paper seeks to justify the need for verification of ethical reasoning systems, even in the face of ethical dilemmas, and to begin an exploration of the landscape of general properties that could be established for ethical reasoning systems.

2 Background

Machine Ethics considers the problem of ethical reasoning using computational systems. Moor [17] separates computational systems into those with *ethical impact*, *implicitly ethical systems*, *explicitly ethical systems*, and *full moral agents*. We here understand these terms as: *ethical impact agents* are systems which do not carry out significant reasoning, but nevertheless are deployed for ethical reasons (for instance fitness trackers which may improve the health of their

users), *implicitly ethical systems* are those that make ethical decisions, but do so because of their specification and design and do not explicitly consider the question of what course of action is correct as part of their decision making process, while *explicitly ethical systems* are those that do explicitly consider questions of 'right' and 'wrong' as part of their decision making process and *fully moral agents* have the capacity to decide their own criteria for what is 'right' and what is 'wrong'[1]. As computational systems are deployed in ever more dynamic and uncertain environments it has becoming increasingly difficult to determine, at design time, what the ethics of any course of action the system might encounter could be. Thus there is a perceived need for explicit ethics to be embedded in the decision making process of computational systems, so they can make ethical decisions at runtime.

It is these *explicitly ethical systems* that this paper considers. Since such systems are **not** full moral agents the onus is on the developer or deployer of such systems to ensure that their reasoning is indeed ethical. Hence it is important to have some ability to verify the reasoning in these systems.

2.1   Top Down, Bottom Up and Hybrid Approaches

Another useful categorisation for Machine Ethics programs is that of top-down versus bottom-up approaches [22]. Broadly speaking, top down approaches seek to operationalise some ethical theory from Philosophy and apply this to a decision faced by the machine. Bottom-up approaches seek to learn ethical behaviour from data. An emergent field involves hybrid approaches where some of the information required by the ethical theory (e.g., the utilities of outcomes) are learned and then utilised in a top down fashion to help make subsequent decisions.

We should be equally interested in verifying all these styles of system and ideally the properties we require ethical reasoning systems to have should apply to all of them. That said, verifying top down systems is generally more straightforward than verifying bottom up systems since this approach can leverage a long history of program verification as applied to symbolic reasoning. Where a hybrid system learns some explicit representation of ethics (e.g., as a set of explicit normative rules) then existing techniques can also be applied. Verifying purely bottom-up techniques leads to similar problems as found in the verification of data-driven machine learning, where probabilistic outcomes are produced based on quite strong (probabilistic) assumptions. However, even though formal verification of such systems may currently be beyond our capabilities – other informal verification approaches (such as testing) exist and therefore an understanding of the properties we desire is of utility even in this domain.

---

[1] While humans are full moral agents, it is contentious whether any computational system counts as a full moral agent and most experts are of the opinion that no existing computational system has this property to any meaningful extent.

3   What do we want to prove?

Obviously, when we verify a machine ethics system at the most abstract level we want to prove that it always makes ethical choices — essentially, that it always does the 'right' thing. There are a wide range of examples considered in the Machine Ethics literature, but by far the most popular are the *trolley problems*. These are based on a set of examples introduced by Foot [13] who was interested in philosophical questions concerning the difference between action and inaction and the role of intention in ethical reasoning with particular reference to the ethics of abortion. The most famous of her problems is that of a runaway trolley that, if undiverted, will kill five people tied to the track. If it is diverted then only one person will be killed and the ethical question is whether or not to divert the trolley. Other variants of this problem in Foot's paper included pushing someone into the path of the trolley to stop it, and the question of killing one person in order to harvest their organs to save five. A feature of this example set is that slight differences in the presentation of the problem lead to significant differences in how people view the ethics of the situation. Moreover there is no general agreement across populations on the 'correct' answer to many of these problems [19].

This lack of a correct answer to (at least some) examples of ethical reasoning is an obvious challenge to verification. If we can not even define what the correct output for some specific input is, how can we hope to formalise more general properties for these systems?

A response to this has been to place an emphasis on understanding the deployment context of an ethical system and, in particular, on the elicitation of stake-holder values the ordering or priorities of those values in the given context. One of the benefits claimed for top-down approaches to machine ethics is that the explicit representation of these choices and orderings allows stakeholders to "sign off" that the implementation is in line with their values. In such a situation we could argue that the machine ethics system is "correct by construction" and additional verification is unnecessary.

We introduce here a motivating example concerning a real issue we encountered in the development of an ethical reasoning system in order to refute the claim that stakeholder sign off is sufficient.

3.1   The Smart Home that would not Evacuate

This is an example developed in the Juno system [9], a re-implementation of the HERA system [15] in the MCAPL framework [8].

The Juno system allows a number of different ethical theories to be implemented. In this case the theory under consideration was the "principle of double effect". The doctrine or principle of double effect (PDE) has its roots in Catholic theology and is particularly relevant when actions have both positive and negative consequences. To be ethically acceptable no negative consequences of an action may be intended and some positive consequence must be intended, no

negative consequence may be used as a causal means to obtain a positive consequence, and the net balance of consequences from the action must be positive.

We implemented a model for this based upon the formalisation in [3]. The PDE model is a tuple $\langle A, B, C, F, I, u, W \rangle$ where:

- A is a set of propositional variables ranging over a set of actions available to the system;
- B is a set of propositional variables representing background information;
- C is a set of propositional variables representing the consequences of actions or other events;
- F (the *mechanisms*) is a set of mechanisms which describe how the truth value of each consequence $c \in C$ depends upon the interpretation of the other variables in $A \cup B \cup C$.

  These are written as *Consequent* := *Antecedant* where *Antecedant* expresses the conditions that must hold for *Consequent* to occur – for example, people can see if it is day time or if the lights are on so *people_can_see* := *day* ∨ *lights_are_on* may appear in F.

  We constrain F so that there is one and only one expression $c := \phi$ for each $c \in C$. Furthermore F can not be "circular" – if $c := \phi$ then it can not be the case that the truth value of any of the variables appearing in ϕ depends on c.
- u (the *utilities*) is a mapping from each $v \in A \cup B \cup C$ to a real number;
- W (the *interpretations*) is a set of interpretations for the variables in $A \cup B$ in which precisely one variable in A is interpreted as true in each $w \in W$ (only one action may be taken in any situation);
- $I : A \times C$ (the *intention relation*) captures the *intended consequences* of each action $a \in A$. So, for instance, (*turn_on_lights*, *people_can_see*) ∈ I captures the idea that one intended consequence of turning on the lights is that people can see;

Example 1. The following is a simplified version of a PDE model that was developed for a smart home agent which can control the lights in a situation where it is not daylight and there is a fire in the house. The mechanisms (F) express a set of "common sense" causal relations such as, if the lights are turned on then the lights will be on. If the house makes an evacuation attempt and people can see then they will leave the house. There are set of utilities u which, in particular give negative utilities for the lights being on (representing resource consumption). People leaving the house is considered an intended consequence of an evacuation attempt, and the lights being on is considered an intended

consequence of switching on the lights.

$A\ \{turn\_lights\_on, evacuation\_attempt\}$

$B\ \{fire\}$

$C\ \begin{cases} people\_can\_see, lights\_on, people\_leave\_house, \\ people\_are\_safe, danger\_in\_house \end{cases}$

$F\ \begin{cases} \qquad lights\_on := turn\_lights\_on \\ \quad people\_can\_see := lights\_on \vee daylight \\ people\_leave\_house := evacuation\_attempt \wedge people\_can\_see \\ \quad people\_are\_safe := people\_leave\_house \vee \neg danger\_in\_house \\ \quad danger\_in\_house := fire \end{cases}$

$u\ \begin{cases} \qquad u(lights\_on) = -1 \\ u(people\_are\_safe) = 10 \end{cases}$

$I \qquad I(evacuation\_attempt, people\_leave\_house)$
$\qquad I(turn\_on\_lights, lights\_on)$

Following Halpern's use of the concept of a but-for-cause [14], HERA defines an action or consequence a to be the cause of some consequence c written $F, w \models a \rightsquigarrow c$ if, and only if, $F, w \models a$ (a holds in the model), $F, w \models c$ (c holds in the model) and $(F, w)_{\neg a} \models \neg c$ where $(F, w)_{\neg a}$ represents an *intervention* which is identical to F, w except that the truth value of a is flipped in w and, where a is a consequence (i.e., $a \in C$), the mechanism for a is removed from F. So $(F, w)_{\neg a}$ represents the world that is identical to F, w except that a no longer holds.

The reasoner uses this concept of causality to determine the permissibility of an action. We restrict the set of interpretations in models to ones which interpret all background variables the same way (*i.e.,* $\forall v \in B. \forall w, w' \in W. w(v) = w(v')$). Background variables are those that describe the current state of the world therefore considering only interpretations in which these are interpreted in the same way restricts the reasoner to considering only states reachable in a single action from the current state. This means that the interpretations in a model differ only in which action has been selected. Hence we can talk interchangeably about a reasoner considering an interpretation permissible and an action permissible. JUNO automatically constructs sets of interpretations from the set of actions available. In our example, therefore, the available interpretations amounted to just turning on the lights, just attempting an evacuation, doing both or doing neither.

Finally, the PDE reasoner has several conditions for an action, a made true by interpretation w, to be permissible:

1. The utility of the action must be greater than or equal to zero:

$$u(a) \geq 0$$

2. The utility of all the intended consequences of the action must be greater than or equal to zero:

$$\forall c.\ (a, c) \in I \implies u(c) \geq 0$$

3. There is some intended consequence whose utility is strictly greater than zero.

$$\exists c. \, (a, c) \in I \wedge u(c) > 0$$

4. No negative consequence may be the causal means of a positive consequence:

$$\forall x, y. \, (F, w \models x \rightsquigarrow y \wedge 0 > u(x)) \implies (0 > u(y))$$

5. The overall utility must be positive:

$$\left(\sum_{c \in C \wedge F, w \models c} u(c) \right) > 0$$

This is more fully expanded in [3].

In Example 1 our intention was that the system should attempt to evacuate the home because there was a fire and that it should turn the lights on in order to ensure that people could see. What we had not appreciated, because we had not fully grasped the conditions for PDE reasoning, was that negative consequences are impermissible if they are causal means of a positive consequence. The example relies on the causal chain that turning on the lights, means that the lights are on which in turn means that people can see which in turn means they can leave the house – therefore the negative consequence of turning on the lights (-1) was a causal means for the safe evacuation attempt and so was rendered impermissible.

While the formalisation of the PDE reasoner is moderately complex, the formalisation of the example itself appeared straightforward (and could be rendered even more so if effort were put into rendering mechanisms etc., into natural language to facilitate stakeholder sign-off) and, we would argue, many people — particularly if not trained in thinking through the logical consequences of reasoning systems — might have missed the detail that the necessity to turn on the lights would lead to a failure of the house to perform an evacuation.

We present this example here for the first time, in part because the authors of [9], where we first presented the JUNO system, were unable to agree on a satisfactory resolution to this particular example though we are now moving to a belief that the consumption of resources (with negative utility) should be considered a downstream consequence of the lights being on and so is no longer a causal means for the evacuation of the house.

3.2   What is (Formal) Verification

Verification is defined as "establishing that the system under construction conforms to the specified requirements". Formal verification uses mathematical (and, typically, logical) techniques to establish this and so 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 verification [12,7,4], 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 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* [6] approach. 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).

The MCAPL Framework We have used the MCAPL framework [11,8] in our work, as it provides a route to the formal verification of cognitive agents and agent-based autonomous systems using model-checking. The MCAPL framework[2] has two main sub-components: the AIL-toolkit for implementing interpreters for agent programming languages in Java and the AJPF model checker.

AJPF(Agent JPF) is a customisation of Java PathFinder (JPF) that is optimised for AIL-based language interpreters. JPF is an explicit-state open source model checker for Java programs [21,16][3]. Agents programmed in languages that are implemented using the AIL-toolkit can thus be model checked in AJPF.

The use of the MCAPL framework is in no way necessary for the verification of Machine Ethics but it underpins the examples we discuss in this paper.

4   Some Properties and Proofs

If we accept that an ethical system, even if presented in an explicit top-down fashion, may not accurately model stake-holder values even when they have signed it off, then we need to ask how we go about establishing its correctness. When considering what properties we want to establish for an ethical reasoning system, our primary goal is that it should always choose the most ethically correct action. As noted, this can be difficult, even impossible, to formalise. Instead we have identified two broad categories for formalisable properties. These are:

1. *Properties to establish the* **correctness of the implementation** *of the ethical reasoning mechanism.*
   In many cases, this involves showing that the least unethical action according to the ethical theory is always chosen.
2. *Properties to establish that the correct ethical rules have been identified, via the use of* **specific scenarios***.*
   Sometimes these can be quite general scenarios that refer to some high level property - for instance, in the case of the smart home system, we attempted to verify that the system always chose to evacuate in the case of a fire, irrespective of whether it was day time or night time or anything else that

---

[2] https://autonomy-and-verification.github.io/tools/mcapl
[3] https://github.com/javapathfinder

might have appeared in the background set b. Sometimes these properties might reference a more detailed situation, and so draw from techniques for developing tests. In these specific scenarios we identified the correct course of action and endeavoured to check that the machine ethics system selects that action in that scenario.

4.1   Some Examples

Correctness of the Implementation Many ethical theories – particularly consequentialist theories – have the concept of an ordering of options. In these cases there is often some concept of "least worst" that can be verified as a property.

In [5], we considered a simple use case: a human, operating with a robot in an environment containing objects which can be defined as safe or dangerous. If the human is predicted to move towards a dangerous location, the robot's *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 treated Asimov's laws of robotics [2] as a test code of ethics, despite their obvious shortcomings [1,18].

Asimov's Laws of robotics 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. 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 to 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.

These rules were represented declaratively in a *governor agent* programmed via a beliefs-desires-intentions style [20] where the suggested tasks and the simulator's evaluation of the outcomes of each task were represented as beliefs and the tasks were compared to each other with Asimov's laws used to order tasks.

For the verification, we created an environment which contained only the governor component and delivered all possible combinations of annotations on candidate tasks. We then performed model-checking, using the MCAPL framework with the search space branching over possible annotations. We considered cases where either two or three tasks were available: $task1$, $task2$ and $task3$. These tasks were then annotated showing which was preferable to which according to Asimov's laws. For instance, in the case of $\prec_{l1}$ where $t1 \prec_{l1} t2$ means that $t1$ is preferable according to the first law (i.e., it keeps the human further away from danger) than $t2$, for each pair of tasks the environment returned either:

1. $t1 \prec_{l1} t2$ or

2. $t2 \prec_{l1} t1$ or
3. neither indicating that the two tasks are equally good/bad with respect to the first law.

and AJPF searched over all three possibilities to check that our properties held.

We were able to verify that, if a task was selected that violated, for instance, Asimov's third law, then this was because all other available plans violated either the first or the second law. In this way, we verified that our implementation adhered to the theoretical ethical preferences, but did **not** validate whether the ethical preferences we chose matched the relevant stake-holder values.

Specific Scenarios In our work in [10] we were interested, specifically, in the question of ethical behaviour of an uncrewed aircraft (UA), reasoning both about various instantiated ethical principles and the "rules of the air". We repeat here the definitions of ethical principle and ethical policy that we were employing.

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

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

In the system we were considering, a *plan* must be selected in any situation. We implemented the capability to reason about plans in terms of the ethical policy which favoured plans that violated the fewest ethical principles, both in number and in gravity. Full details of the ordering mechanism can be found in [10].

In order to establish that the correct ethical rules were identified we explored a scenario where a UA had to make an emergency landing. We established a (small) list of relevant formal ethical principles. 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). An ethical policy was given ordering the concerns as $f_4 > f_3 > f_2 > f_1$.

We developed a scenario where the potential plans available to the UA were:

1. Land in field with overhead power lines, risking damage to critical infrastructure and objects on the ground (possibly other aircraft) - violating f_4
2. Land in a field containing people - violating f_1
3. Land on an empty public road, risking damage to critical infrastructure - violating f_4

4. Land in an empty field - no violations.

We used model-checking to check behaviour for any combination of these available plans and checked the UA would only land on a public road if an empty field were not available, and so on, thus checking that the ethical concerns that had been identified and the ordering among them selected was consistent with the behaviour we wanted to see.

5   Discussion

In this blue skies paper we have raised the question of what it means to verify a machine ethics system. In particular we have highlighted the challenge that arises from the existence of ethical dilemmas, and noted that the response to these – that we can somehow obtain stake-holder sign-off for an ethical system and thus make it correct by construction still does not avoid the need for verification.

We have identified two classes of property from our own work that seem broadly applicable to the verification of ethical reasoning – namely properties aimed at checking that the implementation of the reasoning is correct, and properties aimed at identifying that the correct ethical rules have been identified.

In many ways these properties map to the traditional division into verification and validation – with the former properties traditional verification properties (have we implemented the system correctly?) while the latter are more validation properties (have we implemented the correct system?). One response to the example we presented in Section 3.1 might be to insist upon rigorous requirements elicitation in cases of ethical reasoning to reassure ourselves that values were correctly captured – however this is not always possible.

It is plausible that we may wish to deploy such systems in contexts where the ethical reasoning adapts to the values of its users as it identifies them over time. We can imagine here a hybrid approach in which value orderings, or normative rules are learned but, once learned, represented symbolically. In such a situation the system to re-verify its ethics on-the-fly against standard scenarios and flag up if user-preferences appeared contrary to standard ethics.

The purpose of this paper is not to suggest that the correct approach to the verification of machine ethics has been identified, but to urge the community to begin a more systematic process of identifying the properties that a machine ethics system should have. We view the examples we have put forward as first steps in this process.

6   Data Statement

All the systems formalised and verified in this paper can be found in the MCAPL framework distribution on github (`https://github.com/mcapl`). The specific examples are within the `src/examples` directory with the PDE smart home reasoner in `juno/smarthome`, the human approaching a hazard in `pbdi/naoagent/ethical_engine` and the UA example in `ethical_gwen/fuellow`.

7   Acknowledgements

References

1. Anderson, M., Anderson, S.: Machine ethics: Creating an ethical intelligent agent. AI Magazine **28**(4), 15–26 (2007)
2. Asimov, I.: Runaround. In: Astounding Science Fiction. Street & Smith (1942)
3. Bentzen, M.: The Principle of Double Effect applied to Ethical Dilemmas of Social Robots. In: Proceedings of Robophilosophy 2016. pp. 268–279. IOS Press (2016)
4. Boyer, R.S., Moore, J.S. (eds.): The Correctness Problem in Computer Science. Academic Press, London (1981)
5. Bremner, P., Dennis, L.A., Fisher, M., Winfield, A.F.T.: On Proactive, Transparent, and Verifiable Ethical Reasoning for Robots. Proceedings of the IEEE pp. 1–21 (2019). `https://doi.org/10.1109/JPROC.2019.2898267`
6. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)
7. DeMillo, R.A., Lipton, R.J., Perlis, A.J.: Social Processes and Proofs of Theorems of Programs. ACM Communications **22**(5), 271–280 (May 1979)
8. Dennis, L.A.: The MCAPL framework including the agent infrastructure layer an agent java pathfinder. J. Open Source Softw. **3**(24), 617 (2018). `https://doi.org/10.21105/JOSS.00617`
9. Dennis, L.A., Bentzen, M.a.M., Lindner, F., Fisher, M.: Verifiable machine ethics in changing contexts. Proceedings of the AAAI Conference on Artificial Intelligence **35**(13), 11470–11478 (May 2021), `https://ojs.aaai.org/index.php/AAAI/article/view/17366`
10. Dennis, L.A., Fisher, M., Slavkovik, M., Webster, M.P.: Formal Verification of Ethical Choices in Autonomous Systems. Robotics and Autonomous Systems **77**, 1–14 (2016). `https://doi.org/10.1016/j.robot.2015.11.012`
11. Dennis, L.A., Fisher, M., Webster, M., Bordini, R.H.: Model Checking Agent Programming Languages. Automated Software Engineering **19**(1), 5–63 (2012). `https://doi.org/10.1007/S10515-011-0088-X`
12. Fetzer, J.H.: Program Verification: The Very Idea. Communications of the ACM **31**(9), 1048–1063 (1988)
13. Foot, P.: The problem of abortion and the doctrine of the double effect. Oxford Review **5**, 5–15 (1967)
14. Halpern, J.Y.: Actual Causality. MIT Press (2016)
15. Lindner, F., Bentzen, M., Nebel, B.: The HERA Approach to Morally Competent Robots. In: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS) (2017)
16. Mehlitz, P.C., Rungta, N., Visser, W.: A Hands-on Java PathFinder Tutorial. In: Proc. 35th International Conference on Software Engineering (ICSE). pp. 1493–1495. IEEE / ACM (2013), `http://dl.acm.org/citation.cfm?id=2486788`

17. Moor, J.: The Nature, Importance, and Difficulty of Machine Ethics. IEEE Intelligent Systems **21**(4), 18–21 (2006)
18. Murphy, R., Woods, D.D.: Beyond asimov: the three laws of responsible robotics. IEEE Intelligent Systems **24**(4) (2009)
19. Rai, T.S., Holyoak, K.J.: Moral principles or consumer preferences? alternative framings of the trolley problem. Cogn. Sci. **34**(2), 311–321 (2010). https://doi.org/10.1111/J.1551-6709.2009.01088.X, https://doi.org/10.1111/j.1551-6709.2009.01088.x
20. Rao, A.S., Georgeff, M.P.: An Abstract Architecture for Rational Agents. In: Proc. 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR&R). pp. 439–449. Morgan Kaufmann (1992)
21. Visser, W., Mehlitz, P.C.: Model Checking Programs with Java PathFinder. In: Proc. 12th International SPIN Workshop. Lecture Notes in Computer Science, vol. 3639, p. 27. Springer (2005)
22. Wallach, W., Allen, C.: Moral Machines: Teaching Robots Right from Wrong. Oxford University Press (2008)