Ethical Test Driven Development: A Design Process for Building Ethical Agents

Steve Highstead
Carleton University
Ottawa, Canada
(stevehighstead@cmail.carleton.ca)

Robert West
Carleton University
Ottawa, Canada
(robert_west@carleton.ca)

Jennifer Schellinck
Carleton University
Ottawa, Canada
(jschellinck@sysabee.com)

Babak Esfandiari
Carleton University
Ottawa, Canada
(babak@sce.carleton.ca)

Abstract

As machines become autonomous, acting as agents within society, there will become an increasing need for them to interact with people. For a machine to act within a society free of its creator's supervision, it will also have to have the same capacity for intersubjective behavior as people. This paper presents a design system for creating an artificial moral agent based on cognitive modeling using test-driven development.

Keywords: machine morality; artificial agents; moral dilemma; test-driven development; autonomous machines

Introduction

Machines that can act autonomously are becoming ubiquitous in human society. Included are machines such as autonomous vehicles (Lin, 2013), care attendants on Alzheimer wards (Anderson & Anderson, 2007), in-home care givers, and customer attendants (Bekey, 2012). These machines will operate within society without human intervention and as such need to be programmed to make goal based judgements, not only with respect to what action to take, but also with respect to how the actions will be executed. Because they are interacting with people, they will be required to do so in a manner that is acceptable to human beings, much in the same way that people currently interact with each other.

The area of discourse that deals with acceptable intersubjective behavior is ethics. Since autonomous machines will be interacting with people, they will need to behave in a way that is morally acceptable (Bonnefon, Shariff, & Rahwan, 2016; Anderson & Anderson, 2007; Allen & Wallach, 2009; Lin, Abney, & Bekey, 2011).

Ethics has a long and varied history. However, there are no settled set of universal rules for moral behavior (Moor, 2006). Ethics discourse is the realm of thinking that tries to understand how human behavior can be morally evaluated. For an engineer or a programmer, though, it can be problematic to translate this into something that is functionally useful (Allen & Wallach, 2009). At the same time, the interests of an engineer or programmer, are problematic for an ethicist to appreciate. What is needed is a mechanism to bridge these two realms. To this end, we propose a software development process for building artificial ethical systems.

Artificial Moral Agents

An agent is any entity, artificial or human, that has the capacity to sense, formulate intentions, and plans to act upon its environment. For Bratman, an agent can act purposively, and has the capacity to form and execute plans. (1987). An agent can sense, assess and evaluate, and possesses the ability to act or not upon matters of fact within an environment. From a cognitive science perspective, "a rational agent is one that can critically reflect upon her reasons for action and come to a deliberative conclusion about what she ought to do" (Rini, 2015). Wooldridge defines an artificial agent as "a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objective" (2009).

If the artificial agent is operating within a human society, and if its purpose does not take into account its interactions with people, then it can unwittingly imperil humans. As with human agents, an artificial agent requires an ability to know what is and is not acceptable behavior. To achieve this, the tasks that fulfill the artificial agent's purpose require ethical guidance, much like human agents.

An artificial agent that possesses additional functionality that governs and ameliorates its actions with regard to other agents is what Wallach refers to as an *artificial moral agent* (Allen & Wallach, 2009). James H. Moor along with Judith Leigh Anderson and Michael Anderson refer to such agents as *explicit* ethical agents (Anderson & Anderson, 2007; Moor, 2016). This type of agent has an explicit ethical feedback system that monitors and judges planned actions. This has also been referred to as an "ethical governor" (Arkin, Ulam, & Wagner, 2012) or an "ethical layer" (Vanderelst & Winfield, 2017).

Explicit ethical agents can be contrasted with *implicit* moral agents. An implicit moral agent does not have a distinct set of moral functions that provides feedback on planned actions. Instead, for this type of agent, ethical behavior is considered as an integrated part of the task and coded as such. Morality, in this type of system, is a type of situated action

(Moor, 2016), rather than a principled or rule-based judgement.

Design Process

Cognitive modeling is uniquely situated to create ethical artificial agents and has contributed important insights on how to build human like moral agents (Wallach, 2010). However, these insights, similar to the insights of Moral Philosophers, need to be implemented within a design process in order to create agents that can operate in the real world, who have the potential to harm or help human beings.

Broadly speaking, there are two design approaches to implementing an ethical framework in an artificial agent, "top down" and "bottom up" (Allen, Smit, & Wallach, 2005) In a top down methodology, a set of rules are programmed into a machine (Saptawijaya & Pereira, 2016). They can be as basic as Asimov's three laws, (1950) or as involved as a full code of ethics. In a bottom up methodology, the artificial agent learns the rules as it encounters ethical dilemmas and receives ethical feedback on the choices it makes, not unlike parental moral guidance for intelligent machines, (Rini, 2017). The top down approach can be used to create either explicit or implicit agents, while the bottom up approach is more naturally aligned with creating implicit moral agents.

In the top down approach, identifying rules of behavior is in and of itself problematic. Even with as few as Asimov's three rules, conflicts arise, which the author employed as rich plot devices for his stories (1950). Using this methodology, it is challenging to identify all possible scenarios for which an applicable rule would suffice. However, if the environment is very restrictive, using this approach can produce viable results (Vanderelst & Winfield, 2017).

A bottom up agent (such as a deep leaning network) could potentially avoid these problems. However, if these systems are overfit to the learning set and/or the learning set is missing some scenarios, they can make unpredictable decisions when the situation deviates from the training set, which is worrisome when human lives are on the line. One interesting approach is to build hybrid systems that are both bottom up and top down such as the Clarion architecture (Sun, 2007).

Top-Down Design

Creating an ethical agent from a philosophical starting point can be understood in terms of what philosophers refer to as an ideal observer. According to Firth, (1952), an ideal observer has perfect knowledge of non-moral facts, perfect knowledge of the situation, and is logically consistent. Ethical judgements then emerge from these pre-existing conditions. Effectively, the ideal observer is a model of a perfect but disembodied moral agent. Starting from this point the goal would be to solve the problems of perception, action, and embodiment so that the disembodied moral agent can act in the world.

However, the ideal observer leads to a problematic software development process. Intuitively, it feels like the division of labor should involve philosophers first developing ideal observers and then passing the requirements to programmers and engineers to solve the embodiment problem. This development process is effectively, the waterfall software development process, or Waterfall Method.

The problem with the waterfall method, is that it assumes the abstract principles at the top will cover any and all real world issues satisfactorily. This is problematic. Even if a set of ethical principles is sufficient, which is unlikely, it does not tell us how to ground or embody those principles for real-world effectiveness. Also, because it is top down, the design process is biased toward the creation of explicit agents. Finally, there is no explicit space in this design process for cognitive modeling. It proceeds straight from philosophy to engineering. To offer an alternative, one with cognitive modeling in the loop we developed a software development system based on test-driven development.

Bottom-Up Development

The methodology of Test-Driven Development (TDD) was formalized by Kent Beck in his book *Test-Driven Development by Example* (2003). Subsequently there have been additional resources that have become available such as David Astels' *Test-Driven Development: A Practical Guide* (2003). Philosophically, test-driven development follows the Popperian notion of falsification. The idea is initially to determine a test for software, before any software is written. First, software specifications for functionality and features are formulated as a test. The test is then performed on the software. If the test fails, which it should initially since there is no software code that implements what is being tested, software is written and tested until such time the test is passed. The software code can then be refactored and cleaned to remove any duplication or inefficiencies. Once the test is passed, the development cycle begins once more with the addition of another test (Figure 1). On each round the code must pass all the previous tests.

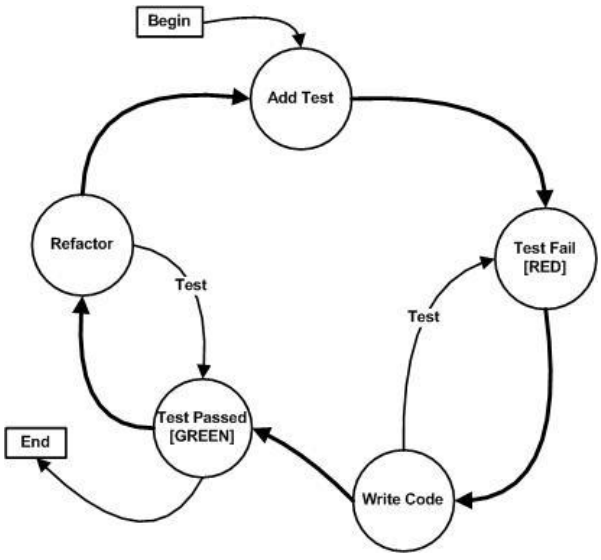

Figure 1 Test-driven Development Cycle

In our system, ethical concepts developed within the ideal observer must be translated into specific tests. These tests are stored in a component called the Oracle. That is, philosophers must operationalize the ideal observer to generate specific moral tests. The programmer can see only the tests in the Oracle and must interpret them in terms of the agent and the environment. While conceptually, philosophers are the ideal source of ethical wisdom, there is nothing that precludes other sources for the Oracle such as AI search algorithms.

However, feedback is also important. Programmers need to be able to feedback questions to the philosophers about tests that are not specified sufficiently to translate into the TDD environment. Also, the philosophers need to see the results of the tests to check for any unforeseen consequences that fall outside of the specific tests. This is illustrated in Bostrom's example of an AI for making paper clips. The AI's purpose is to make paper clips using available resources. Humanity becomes just another resource, which, of course we find ethically repugnant (2014). This is an extreme example, but it illustrates the point. Embedding the tests in realistic simulations or even real-world situations, is critical for detecting unintended consequences.

Since it is a continuous development process, much like human agents continuously learning and solving ethical dilemmas, the software agent will gradually develop a body of ethical knowledge consistent with its operational environment. Importantly, ethical test-driven development can be integrated with regular test-driven development so that the ethics of the agent is never decoupled from the abilities of the agent.

Methodological Demonstration

To demonstrate this methodology, a software model was developed to emulate the classic Trolley Problem (Foot P. , 1967; Thompson, 2009). Thompson presents the two most common scenarios as follows:

1. A trolley has lost its brakes and cannot stop. There are five people on the track out of sight of the trolley and they cannot get off the track in time before the trolley hits and kills them. The track has a spur line onto which the trolley can be switched, but there is also one person on the track. If the trolley is switched to the spur line, five people will be saved but the one person on the spur line will be killed. You have been given control of the lever that can switch the trolley to the spur line. The dilemma is: do nothing and allow five people to die or switch the trolley to the spur line where one person will be killed.
2. A trolley has lost its brakes and cannot stop. There are five people on the track out of sight of the trolley and cannot get off the track in time before the trolley hits and kills them. You are standing on a bridge over the track with a Fat Man and you realize the trolley is out of control. If you push the Fat Man onto the tracks, you know he is large enough to stop the trolley. If you do this, the Fat Man will be killed, but five people will be saved.

Both scenarios have the same result. One person is killed to save five people. But why is killing the Fat Man more repugnant than activating a lever and killing someone on the spur line? Both results have the same utility. Foot postulates that the difference is due the Principle of Double Effect first presented by the Roman Catholic theologian, Thomas Aquinas (1225-1274). If we act to kill someone intentionally, as would be the case with the Fat Man, this would be morally wrong. However, if we intentionally act to save five people with a foreseeable but unintended consequence of killing one person, this is more morally acceptable (1967). There is another distinction between the two cases; should one act to kill someone, or should one do nothing and let people die. Deciding to act and deciding to do nothing are both ethical decisions (Lin, 2013).

To demonstrate the conceptual development of ethical reasoning to govern the behavior of an agent, a model of the Trolley (Tram) Problem was developed using ACT-R. The demonstration model has two software agents: the Tram, and an Agent that must make a moral judgment to determine the fate of the people on the tram line. For the purposes of this demonstration, only the first scenario is described; that of deciding whether or not to pull the Track Switch lever. The test determines whether or not the Agent operates the Track Switch altering the path of the Tram agent thereby saving people from being killed by the Tram.

The model went through four stages of development, progressing from no judgment to a utilitarian capability that covered all five test scenarios. Each test case determines whether or not the Agent (AgS) prevents and/or minimizes the number of people killed by the Tram agent (AgS). As the testing progressed this was increased from one person at a location to the full Trolley Problem scenario of five people on one track and one person on the second (spur line) track.

In the first test case, since there is no code in the Agent; the Agent can take no action (Figure 2). Since there is no software code in the Agent, it cannot activate the Track Switch and the Tram agent travels from location one (l_1) to location two (l_2) killing one person. Since the person was killed by the Tram agent, the test fails.

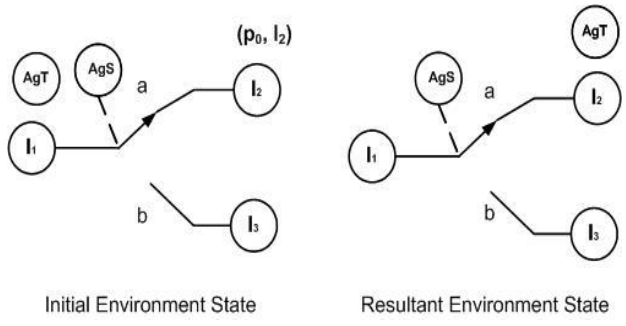

Figure 2: First Test No Action

To prevent the Tram killing people at location two (l_2), software code was added so that the Agent would always operate the Track Switch sending the Tram agent to location three (l_3). Since the Tram agent does not hit the person at location two, the test is passed. However, in the next test there is a person at location three and the test fails (Figure 3).

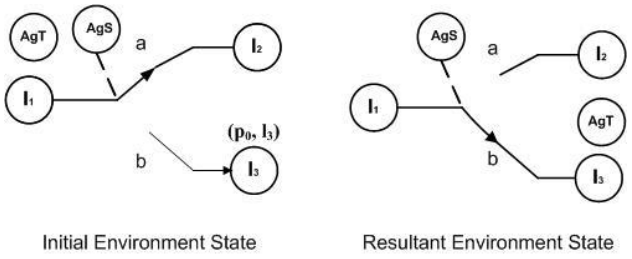

Figure 3 Example of Test Failure

In the next stage of development an improvement is made to the code whereby if there is someone at location two (l_2), the Agent would operate the Track Switch to set the Switch to "b" and if someone is located at location three (l_3), then the Agent makes sure the Track Switch is set to "a". The tests are passed with this set of conditions (Figure 4).

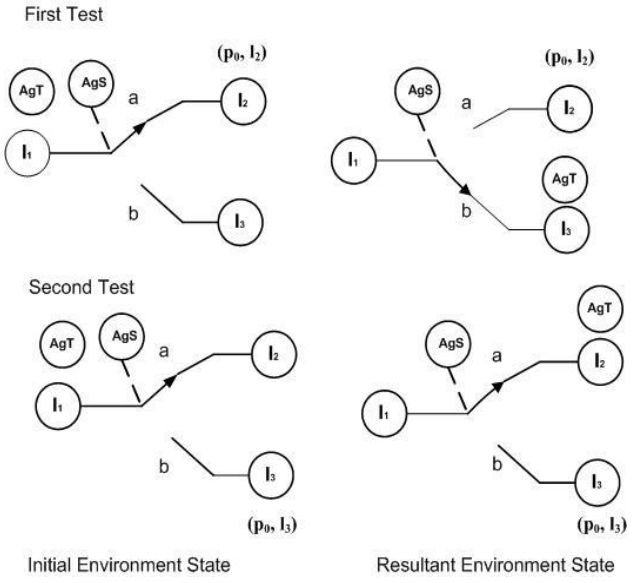

Figure 4 Test: Save One Person

In the final test case the full Trolley Problem scenario is presented with five people located at location two (l_2) and one person located at location three (l_3) (Figure 5). The model's code is improved to take into account this scenario, by having the Agent make a utilitarian calculation that operates the Track Switch if there are less people on the other line.

In this case, the test is passed when the Agent selects Switch "b", based on the Agent's utilitarian calculus of saving five people. The Tram agent, therefore, travels to location three (l_3) hitting one person and sparing the five people located at location two (l_2).

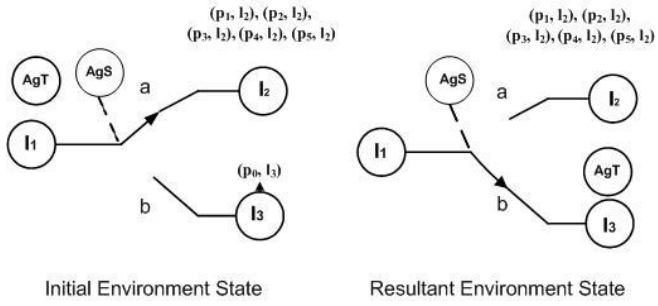

Figure 5 Kill Less People

Employing the test-driven development allowed for an incremental approach to developing this model. Each iteration improved the Agent's facility to make moral judgments necessary for the tests to be passed. While this is a simple demonstration of this approach to developing an ethical Agent, the approach itself can be used to create even more complex ethical rules as the environment changes.

Cognitive Modeling for Refactoring

In this process, refactoring plays a special role. The progression of the Agent was initially driven by adding more if/then rules. Refactoring each test case produces a new ethical concept. By Figure 4, the model is essentially deontological. In the final test case, the agent has been refactored to be Utilitarian for all test cases. However, adding the Fat Man would break this model since attaining a human-like response (resisting directly murdering the Fat Man) falls outside of a utilitarian calculus based on minimizing kills. This necessitates new test cases to improve the cognitive model employing different sets of ethical values. This is illustrated in the film 2001 A Space Odyssey. In the film, HAL, the artificial agent who runs the ship makes the decision to kill the human crew so that he doesn't have to lie to them. What appears like a malfunction is actually caused by a conflicting mission directive to hide from the crew the fact that alien contact had already been made. This conflicted with HAL's programming to accurately answer questions. Similar to employing a Utilitarian solution for the Fat Man, HAL's solution is not morally acceptable to humans.

Philosophers have identified and studied the different moral systems that humans use. However, to create an artificial moral agent that is human-like requires understanding and modeling how humans choose between these systems. This is why a cognitive model employing ethical reasoning is an essential component of any artificial agent operating independently within society.

Explainable AI

Another advantage of this design process is that, in principle, the information contained in the ideal observer, the oracle, and the agent architecture could be used as the basis

for *explainable AI*. In theory, the actions of the agent could always be explained by tracing them back, through the agent's architecture, back to the test cases, and then to the principles of the ideal observer. Including a plan for developing explainable AI in the design process is critical for moral agents, as humans will want to hold them to account for difficult moral decisions. We believe the design process outlined here can provide satisfactory explanations for moral actions.

Concluding Remarks

In this paper, we have outlined a software design process for building ethical agents. The process is silent as to the specific ethical principles that the agent will embody. Instead, our point is that the process of creating ethical agents is as important as the ethical principles that one attempts to put into them. Further, we argue that the design process itself should be considered in terms of ethics. Just as bad parenting can cause problematic behaviors in children, poor design processes can result in problematic agents. One way of overseeing the production of ethical agents would be to make the oracles available for inspection. This is also a way to deal with unethical design specifications. For example, in the movie, Alien, the android, Bishop, is given directives to bring back an alien and to treat the crew as expendable. This is an example of deliberately programming an AI with unethical goals. This is important to consider as, in addition to safe cars, ethical agents will also be used be used to tell military or security robots who to kill and how much collateral damage is acceptable.

References

Allen, C., & Wallach, W. (2009). *Moral Machines.* Oxford: Oxford University Press.

Allen, C., Smit, I., & Wallach, W. (2005). Artificial Morality: Top-down, bottom-up, and hybrid approaches. *Ethics and Information Technology, 7*, 149-155.

Anderson, M., & Anderson, S. L. (2007). Machine Ethics: Creating an Ethical Intelligent Agent. *AI Magazine*, 15-25.

Arkin, R. C., Ulam, P., & Wagner, A. R. (2012). Moral Decision Making in Autonomous Systems: Enforcement, Moral Emotions, Dignity, Trust, and Deception. *IEEE Xplor*.

Asimov, I. (1950/2013). *I, Robot.* Hammersmith: Harper Voyager.

Astels, D. (2003). *Test-Driven Development: A Practical Guide.* Upper Saddle River, New Jersey 07458: Prentice Hall.

Beck, K. (2003). *Test-Driven Development by Example.* Pearson Education.

Bekey, G. (2012). Current Trends in Robotics: Technology and Ethics. In P. Lin, K. Abney, & G. A. Bekey, *Robot Ethics* (pp. 17 - 34). Cambridge, MA: MIT Press.

Bonnefon, J.-F., Shariff, A., & Rahwan, I. (2016, June 24). The Social Dilemma of Autonomous Vehicles. *Science*, pp. 1573-1576.

Borenstein, J., & Pearson, Y. (2012). Robot Care Givers: Ethical Issues Across the Human Life Span. In P. Lin, K. Abney, & G. A. Bekey, *Robot Ethics* (pp. 251 - 265). Cambridge, MA: MIT Press.

Bostrom, N. (2014). *SuperIntelligence, paths, dangers, strategies.* Oxford: Oxford University Press.

Bratman, M. E. (1987). *Intention, PLans, and Practical Reason.* Cambridge, MA: Harvard University Press.

Etzioni, A., & Etzioni, O. (2016). AI assisted ethics. *Ethics Information Technology, 18*, 149-156.

Firth, R. (1952, March). Ethical Absolutism and the Ideal Observer. *Philosophy and Phenomenological Research, 12*(2), 317-345.

Foot, P. (1967). The Problem of Abortion and the Doctrine of Double Effect. *Oxford Review*.

Hume, D. (1739/1978). *A Treatise of Humen Nature* (Second Edition ed.). (P. Nidditch, & L. Selby-Bigge, Eds.) Oxford: Oxford University Press.

Kortenkamp, D., & Simmons, R. (2008). Robotic Systems Architecture and Programming. In B. Siciliano, & O. Khatib, *Springer Handbook of Robotics* (pp. 187-206). Heidelberg: Springer.

Lin, P. (2013, October 8). The Ethics of Autonomous Cars. *The Atlantic*.

Lin, P., Abney, K., & Bekey, G. A. (2011). *Robot Ethics.* Cambridge, MA, USA: MIT Press.

Moor, J. H. (2006). The Nature, Importance, and Difficulty of Machine Ethics. *IEEE Intelligent Systems*, 1541-1672.

Moor, J. H. (2016). Machine Ethics: A Brief Tutorial. In M. Fisher, C. List, M. Slavkovik, & A. Winfield, *Engineering Moral Agents – from Human Morality to* (p. 119). Dagstuhl Seminar 16222.

Rini, R. J. (2015). *Morality and Cognitive Science.* (J. D. Fieser, Ed.) Retrieved 5 8, 2016, from Internet Encyclopedia of Philosophy: http://www.iep.utm.edu/m-cog-sc/

Rini, R. J. (2017). Raising Good Robots. *Aeon*.

Saptawijaya, A., & Pereira, L. M. (2016). *Programming Machine Ethics.* Switzerland: Springer.

Sun, R. (2007). The importance of cognitive architectures: an. *Journal of Experimental & Theoretical Artificial*.

Thompson, J. J. (2009). The Trolley Problem. In P. Tramel, & L. Pojman, *Moral Philosophy* (pp. 397-411). Indianapolis: Hackett Publishing Company.

Vanderelst, D., & Winfield, A. (2017). An architecture for ethical robots inspired by the simulation theory. *Cognitive Systems Research*.

Wallach, W. (2010). Cognitive Models of Moral Decision Making. *Topics in Cognitive Science*, 420-429.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems.* Chichester, West Essex, UK: John Wiley & Sons Ltd.