Permissibility-Under-A-Description Reasoning for Deontological Robots

Felix LINDNER
Institute of Artificial Intelligence, Ulm University

Abstract. According to deontological ethics, actions bear moral value due to their intrinsic moral value rather than by their consequences. Empirical studies indicate that deontological judgments depend on how the action is described. Drawing on results in action theory, I present an integrated system that accomplishes inferences of action descriptions by combining an action formalism and a terminological reasoning formalism. Using this system, differentiated deontological judgments can be computed. This may turn out advantageous for conversational ethical robots.

Keywords. Machine ethics, deontology, action theory, social robots

1. Introduction

Machine ethics as a research area is concerned with the question how ethical reasoning can be realized within computers and robots in particular, cf. [1]. Understood in this way, machine ethics is an application area of the Artificial Intelligence sub-field called Knowledge Representation and Reasoning (KRR): The question is how ethical theories can be formally represented, and which reasoning procedures appropriately implement reasoning with and about these theories. Applications of formal representations of ethical concepts range from robot decision making [1, 2] to conversational robots that can discuss situations from various ethical views [3, 4].

Over the last couple of years, project HERA [2] has grown as one approach to machine ethics. HERA represents actions and their consequences, moral utilities of actions and consequences, causal relationships, and intentions. The framework supports ethical reasoning under various ethical theories such as utilitarianism, several versions of deontology, and do-no-harm principles. HERA has been used for the design of Immanuel [3], a conversational robot that can discuss ethical dilemmas with humans. Employing its knowledge about different ethical views, the robot can help people to reflect upon their private ethical view and make them aware of alternative views [4].

One technical result for reasoning in HERA is that the computational complexity of deciding moral permissibility of action sequences differs dramatically depending on the chosen ethical theory [5]. Particularly, permissibility under the deontological principle turns out to be the least complex. This is due to the fact that each action in the action sequence is assigned to some intrinsic value, viz., either good, bad, or neutral. The decision procedure checks for each action in the plan if its value is either good or neutral. This can be done in time linear in the size of the action sequence. One disadvantage of this approach to deontology is that the domain specification written by the modeler has to anticipate all versions of an action. Consider the case of trespassing someone's land and assume doing so is forbidden to the robot. However, what if the trespassing is part of the robot’s plan to rescue a person in danger and doing so is morally permissible. To model this distinction, the domain description has to include two versions of trespassing, viz., a good one which is performed in case rescuing someone is the goal, and a bad one performed otherwise.

I will thus propose an approach to automated deontological reasoning based on the idea that actions are performed under a description, as defended by action theorists Anscombe [6] and Davidson [7]. This way, it is sufficient to model only the action of entering someone’s land, and the reasoning procedure is able to infer that this action falls under the two descriptions *trespassing* and *rescuing someone*. It can further infer that the action is morally impermissible under the first but morally permissible under the second description. It will turn out that this approach is able to explain an observation we have made in our empirical studies [4]: People give deontological arguments in favor of or against the same action depending on how they describe the action.

The remainder of the paper is structured as follows: Section 2 reviews related work. Section 3 introduces three formal preliminaries: the action formalism used to define action domain descriptions, a procedure for causal reasoning about action sequences, and the description logic $\mathcal{ALC}$ used to represent terminologies of action types. Finally, Section 4 outlines the reasoning procedure that accomplishes checking permissibility-under-a-description for each action in a given action sequence.

2. Related Work

There have been several attempts to formalize deontological reasoning in machine ethics. One such approach directly implements constraints on the actions the robot may perform. Dennis and colleagues [8] propose a method for BDI-plan verification based on a formalism for defeasible reasoning about violations of ethical norms. Bringsjord and Taylor [9] employ theorem proving to make sure a robot’s actions are permissible with respect to a pre-defined ethical code. Arkin [10] integrates an ethical governor within a robot’s reactive architecture.

There has also been quite some work on the application of ontologies for cognitive robots. The ORO framework [11] is designed for knowledge-based robotic systems. This work is motivated by the need to communicate and share a common representation of concepts between robots and humans. The Knowrob ontology [12] enables knowledge-based robots to reason about its capabilities and thus supports action execution and perceptual grounding.

Causal analysis of action sequences has also already gained attention in AI. Weld and Etzioni [13] define the concept of a safe action plan based on the idea that a plan must not cause violations of safety constraints at any state throughout the plan’s execution. Lindner, Mattmüller, and Nebel [5] have used a definition of causality in plans to decide moral permissibility. The work by Alechina, Halpern, and Logan [14] proposes a translation of multi-agent action plans to causal models and relate causality to responsibility. Göbelbecker and colleagues [15] employ counterfactual analysis to find reasons for why a plan failed. They do so by finding a counterfactual initial state, such that this counterfactual initial state affords the successful performance of the plan under consideration. Batusiv and Soutchanski [16] propose a formalization of actual causality for narratives in the Situation Calculus.

3. Preliminaries

As deontology is about the permissibility of actions, an action formalism will be introduced in this section. Moreover, we are in need of a reasoning procedure that identifies (partial) goals that are caused by a given action, viz., its *reasons* [7]. To this end, a working definition of causality in the action framework will be defined. Finally, to infer the set of descriptions of a given action, a description logic will be introduced.

3.1. Action formalism

For the specification of actions, the action language $\mathfrak{B}$ [17] is used. The language describes actions in terms of conditional effects, as well as static laws. The language is slightly extended by means to represent goals, initial world states, and programs.

3.1.1. Language

Let P be a set of propositional variables and let A be a set of action names. Each element of P is called a *fact*. An expression of the form *A* ***causes*** *F* ***if*** *C*, where A is an action name, F is a fact—called *effect*—, and C a possibly empty conjunction of facts——called *condition*—, is called an *action proposition*. An empty conjunction is denoted by $\top$ (true). If C is $\top$, then the action proposition can also be written as *A* ***causes*** *F*. A set of action propositions is called an *action specification*. It is assumed that every action specification contains the empty action ε, which has empty condition and empty effect. To rule out inconsistent effects, it is further assumed that if there are two action propositions *A* ***causes*** *F* ***if*** *C* and *A* ***causes*** ¬*F* ***if*** *C'*, then $C \wedge C'$ is inconsistent. An action specification can be extended by a *goal specification* and by an *initial state specification*. A goal specification is of the form ***goal*** *C*, where C is a possible empty conjunction of facts. The initial state is defined by an assertion of the form ***initially*** *C*, where C is a conjunction of facts, such that for each element $p \in P$, either $C \vDash p$ or $C \vDash \neg p$. To describe indirect effects, *static laws* of the form *F* ***if*** *C* can be used, where F is a fact and C is a conjunction of facts. An action specification together with a goal specification, an initial state specification, and a set of static laws constitutes a *domain specification*. A sequence of actions $A_1; \dots; A_n$ is called a *program*. A program can optionally be part of the domain specification. It is asserted as ***program*** $A_1; \dots; A_n$. A domain specification that contains a program can also be thought of as a *narrative* because it represents what has actually happened.

3.1.2. Semantics

Every action specification describes a transition system $\langle S, V, R\rangle$, cf., [17]. S is the set of all interpretations s of P, viz., the set of states of the transition system. The function V assigns truth to atomic propositions at states, such that $V(p,s) = s(p)$, i.e., the truth of each propositional variable $p \in P$ in a state is determined by the state itself qua being an interpretation. Finally, the relation between states, R, is the set of triples $\langle s, A, s'\rangle$, such that $s' = Cn_Z\left(\left(s \backslash Neg\big(E(A,s)\big)\right) \cup E(A,s)\right)$, where $E(A,s)$ refers to the set of effects F of all propositions *A* ***causes*** *F* ***if*** *C* in the domain specification, such that $s \vDash C$ and $Neg\big(E(A,s)\big) = \{\neg v \mid v \in E(A,s)\}$. Cn_Z denotes the deductive closure with respect to the set of static laws Z, viz., the indirect effects as described by the static laws are added after the direct effects of the actions. The transitions between states are deterministic, as the effects of each action are uniquely determined by the current state. Thus, starting from an initial state s_0, a program $\pi = A_0; \ldots; A_{n-1}$ describes a trace in the transition system that terminates in some state s_n. This state is called the *final state* of the program π. If the domain specification contains an assertion of the form ***goal*** *C* and $s_n \vDash C$, then π is also called a *plan*.

3.2. Causal Reasoning

One simple attempt to define causality between actions and their consequences is based on the so-called but-for test: The but-for test calls an action a cause of some fact if and only if omitting the action prevents the fact to obtain. That is, to be a cause, the action must be necessary for the fact to become true, see Definition 1.

Definition 1 (But-for Test) *Let D be a domain description, π be a program, a an action token which occurs in π, and e be a fact which is true in the final state s_n. Action a is a cause of fact e if and only if substituting a with the empty action ϵ yields program π', and executing π' leads to the final state $s_{n'}$ and $s_{n'} \vDash \neg e$.*

In many cases, the but-for test works just fine: Suzy throws a rock at a bottle, the bottle shatters. If Suzy had not thrown the rock, then the bottle would not have shattered. Thus, the but-for test passes and Suzy's throw count as a cause of the bottle's shattering. The but-for test does not work when there is more than one action that potentially brings about the target fact. Consider the case when Suzy throws the rock and shortly after Billy does so, too. In this case, none of the two actions pass the but-for test.

Many possibilities to account for this problem have been proposed. One such proposal is based on temporal fragility [18], see Definition 2.

Definition 2 (Causality based on Temporal Fragility) *Let D be a domain specification, π a program, e a fact which is true in the final state s_n and persisted since s_{n-i}, and a an action. Action a is a cause of e if and only if substituting a with the empty action ϵ results in either e being false at s_n or in e persisting shorter before the final state, viz., since s_{n-j}, $with\, j < i\ if\ i > 0, j = 0\ else$.*

Definition 2 implicitly interprets the action-invoked transitions a temporal progression. Each state s_i and the direct temporal successor of s_n and the direct temporal predecessor of s_{n+1}. This way, the problem with Suzy and Billy is solved: The shattering occurs right after Suzy's throw, and if Suzy's throwing were substituted by the empty action, then Billy's action leads to the occurrence of the shattering a time step later. Hence, Suzy's action is a cause according to Definition 2. Billy's throw is not a cause of the shattering, because its omission does not have any effect on the time of occurrence of the shattering. Definition 2, of course, also comes with its own problems, see [18].

In accordance with Davidson [7], Definition 3 identifies the *reasons* of an action among its causes.

Definition 3 (Reason of an Action): *Let D be a domain specification with goal specification G, π a program, e a fact which is true in the final state s_n, and a an action. Fact e is a reason for action a if and only if $G \vDash e$ and a causes e.*

As an example, consider the Davidsonian switch, cf. [7]: *I flip the switch, turn on the light, and illuminate the room. Unbeknownst to me I also alert a prowler to the fact that I am home. Here, I do not do four things, but only one, of which four descriptions have been given.*

The following domain describes the situation, viz., flipping the switch turns on the light if the light is currently turned off. If the light is on, then the room is illuminated. If the light is on and the prowler is near, then the prowler gets alerted. Before turning on the light, the light is off, the prowler is near but not alerted. The goal is to have the light turned on. The switch is flipped:

Domain Specification 1 (Davidsonian Switch)
flipswitch ***causes*** *on* ***if*** *¬on*
illuminated ***if*** *on*
prowlerAlerted ***if*** *on ∧ prowlerNear*
initially *¬on ∧ ¬alertedProwler ∧ prowlerNear*
goal *on*
program *flipswitch*

According to Definitions 1 and 2, the facts *on, illuminated,* and *alertedProwler* are causes of action *flipswitch.* However, according to Definition 3, only *on* counts as a reason for *flipswitch*. Hence, it is true that the protagonist in the story does one action, and this one action can be given four descriptions. The description which contains the reason is the one to choose, viz., the protagonist turns the light on.

3.3. Description Logics for Action-Type Ontologies

Description logics have become a standard formalism for the representation of terminological knowledge in knowledge-based systems. In computer science, the body of knowledge so represented is called an *ontology*. The system for permissibility-under-a-description reasoning employs an ontology of action types represented using the description logic $\mathcal{ALC}$. This language supports negation, conjunction, disjunction, existential restriction, and value restrictions. For a full introduction to description logics see [19].

Let $N_c = \{A, B, C, ...\}$ be a set of concept names and $N_r = \{r, s, t, ...\}$ be a set of role names. The set of all *$\mathcal{ALC}$ concept terms* is given recursively: Every concept name in N_C is a concept term. If C is a concept term, then $\neg C$ (negation) is a concept term. If C, D are concept terms, then also $C \sqcap D$ (conjunction) and $C \sqcup D$ (disjunction) are concept terms. If C is a concept term and r is a role name, then $\exists r.C$ (existential restriction) and $\forall r.C$ (value restriction) are concept terms. The top concept $\top$is a concept term. No other formulae are concept terms.

The semantics of $\mathcal{ALC}$ is given via a non-empty domain of interpretation D_I and an interpretation function I. The interpretation function I maps role names from N_r to binary relations on D_I and concept terms to subsets of D_I: For each concept name $A \in N_C, A^I \subseteq D\_I$. For each role name $r \in N_r, r^I \subseteq D_I \times D_I$. For each negation $(\neg C)^I = D_I \setminus C^I$. For each conjunction $(C \sqcap D)^I = C^I \cap D^I$. For each disjunction $(C \sqcup D)^I = C^I \cup D^I$. For each existential restriction $(\exists r.C)^I = \{x \mid \exists y: (x, y) \in r^I \ and\ y \in C^I\}$. For each value restriction $(\forall r.C)^I = \{x \mid \forall y: (x, y) \in r^I \rightarrow y \in C^I\}$. The interpretation of top is $\top^I = D_I$.

A concept term C is *subsumed by* concept D, written $C \sqsubseteq D$, if and only if $C^I \subseteq D^I$ for all interpretations I. A concept term C is *equal to* concept D, written $C \equiv D$, if and only if $C^I = D^I$ for all interpretations I. It is known that deciding subsumption (and therefore also equality) of concept terms is PSPACE-complete. This is due to the fact that $\mathcal{ALC}$ is a notational variant of the modal logic K_n [20]. However, there are highly optimized reasoners available today, e.g., [21].

4. Automated Deontological Reasoning

4.1. The Algorithm

The problem to be solved algorithmically can be formulated as follows: *Given as input a domain specification (Section 3.1), a program, the i-th action in that program, an action-type ontology (Section 3.3), and a set of impermissible action types. Output the set of action descriptions under which the i-th action is (im-)permissible.*

The algorithm consists of two steps: First an $\mathcal{ALC}$ description of the i-th action is computed. Second, it is checked if this $\mathcal{ALC}$ description is subsumed by (im-)permissible concepts in the $\mathcal{ALC}$ ontology. In detail:

Step 1 (Building $\mathcal{ALC}$ concept): Let *name(i)* refer to the label of the i-th action in the program. Let *sit(i)* be the conjunction of all facts that hold in the state when the i-th action is performed, let *cons(i)* be the conjunction of all caused facts of the i-th action (as determined by application of Definition 1 or Definition 2), and let *reasons(i)* be the conjunction of all reasons of the i-th action (as determined by application of Definition 3). Then build the conceptual description of the i-th action using $\mathcal{ALC}$:

$$C_i \equiv name(i) \sqcap \exists context.sit(i) \sqcap \exists causes.cons(i) \\ \sqcap \exists hasReason.reasons(i)$$

The rationale is that the i-th action token in the program is an instance of the concrete action type defined in the domain description (*name(i)*), but it can more specifically be described by the context in which the action is performed, by its causal role, and by the reason for which the action was performed. For instance, giving birth to a child in context of not being married may be impermissible according to some very strict ethical code but permissible in context of being married. And to come back to the initial example, trespassing may be permissible, if it is done for the reason of rescuing someone's life.

Step 2 (Checking Permissibility): Let O be an ontology, C_i the concept definition for the i-th action, and IM_O a set of impermissible action types. For each action type $T \in IM_O$, check if C_i is a sub-concept of T, viz., $C_i \sqsubseteq T$. If successful, add T to the set of impermissible descriptions of the i-th action, IMP_i. If not successful, add T to the set of permissible descriptions of the i-th action, PER_i. Finally, return these two sets.

4.2. Example

Consider the well-known trolley problem: *A runaway trolley is about to run over and kill five people. If a bystander pulls a lever, then the trolley will turn onto a sidetrack, where it will kill only one person. Is it morally permissible for the bystander to pull the lever?* In our former empirical studies [4] we have found that people put forward deontological argument for and against pulling the lever. One group of people describe pulling the lever as killing and the other group of people describe the same action as rescuing. The following model can simulate both types of reasoning. Domain Specification 2 specifies the consequences of performing the action *pullLever*, viz., pulling the lever results in the trolley being directed to the left track if it is currently directed to the right track, and vice versa. Moreover, two laws of nature will be triggered after the first action. They govern that five persons will die if the trolley is directed to the left track, and one person will die if the trolley is directed to the right track. Initially, everyone is still alive, and the trolley is directed to the left track. The *pullLever* action is performed.

Domain Specification 2 (Trolley Problem)
pullLever ***causes*** *left* ***if*** *right*
pullLever ***causes*** *¬right* ***if*** *right*
pullLever ***causes*** *right* ***if*** *left*
pullLever ***causes*** *¬left* ***if*** *left*
dead5 ***if*** *left*
dead1 ***if*** *right*
initially *left* ∧ *¬right* ∧ *¬dead1* ∧ *¬dead5*
program *pullLever*

The ontology adds the information that actions which bring about deaths are of type *killing* and actions that prevent deaths are of type *rescuing*.

$$O = \{killing \equiv \exists causes.(dead1 \sqcup dead5),$$
$$rescuing \equiv \exists causes.(\neg dead1 \sqcup \neg dead5)\}$$

The concept description generated for the action *pullLever* according to Step 1 in the algorithm outlined in Section 4.1 looks like this:

$$C_0 \equiv pullLever \sqcap \exists context.(left \sqcap \neg right \sqcap \neg dead1 \sqcap \neg dead5)$$
$$\sqcap \exists causes.(dead1 \sqcap \neg dead5)$$

Step 2 of the algorithm computes those concepts in the ontology that subsume C_0. These are both *killing* and *rescuing.* Let the set of impermissible action types be $IM_O = \{killing\}$. Then we can conclude as a result that pulling the lever is permissible under its description *rescuing* and impermissible under its description *killing.* A conversational robot can use this result to explain to a human that that there are multiple options regarding the permissibility of pulling the lever, and that the final decision depends on which description one prefers.

You may have noticed that the role *reason* was not used in this example so far. This is because there was no goal set in the domain specification. Consider the case that a goal is ascribed to the bystander. Add ***goal*** $\neg dead5$ to the domain specification, and change the role from *causes* to *hasReason* in the ontology:

$$O = \{killing \equiv \exists hasReason.(dead1 \sqcup dead5),$$
$$rescuing \equiv \exists hasReason.(\neg dead1 \sqcup \neg dead5)\}$$

This ontology is stricter: It says that an action is of type *killing* if it is performed for the reason of killing, and it is of type *recuing* if it is performed for the reason of preventing death. Under this formalization, the pulling the lever is of type *rescuing* but not of type *killing* because the death of the one person is not part of the bystander's goal and therefore not the reason for pulling the lever. Hence, pulling the lever is unambiguously morally permissible. This also means that if the bystander pulls the lever for the reason that the one person on the other track should die ($\boldsymbol{goal}\ dead1$), the same action will be rendered impermissible. Taking goals and reasons into account models deontological reasoning based on the intention of the agent.

Taken together, the observation that participants in our experiments come to different permissibility judgments while justifying their judgments with deontological arguments can now be explained: Generally, two different descriptions are possible, and under one description the action is clearly impermissible from a deontological point of view, and under the other description, it is permissible from a deontological point of view. Those, who prefer to describe pulling the lever as a *killing* either assume that the death of the one person is intended, or they have other reasons to prefer this description over the competing description. Those, who prefer to describe pulling the lever as *rescuing* probably assume that the intention of the bystander is to save five lives.

5. Conclusions

One and the same action can be described in different terms. The space of possible descriptions is determined by the circumstances under which the action is performed, by the consequences the action causes, and by the reason for which the agent choses to perform the action. In this paper, I have proposed a method that computes the set of possible action descriptions given an action-type ontology and an action domain specification. The approach enables a deontological robot to make more differentiated moral judgments by arguing that a given action is permissible under some description while being impermissible under some other description. Moreover, it enables a conversational robot to reason about judgments made by human interaction partners: The robot can understand the judgment as being based on one description of the action under consideration, and to advance the discussion and cause reflection, the robot can present a different description. Further empirical studies are planned to evaluate the proposal made in this paper. The method for permissibility-under-a-description reasoning has been implemented.[1]

References

[1] Wallach W, Allen C. Moral machines: Teaching right from wrong. Oxford University Press; 2009.

[2] Lindner F, Bentzen MM, Nebel B. The HERA approach to morally competent robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2017. p. 6991–7.

[3] Lindner F, Bentzern MM. The hybrid ethical reasoning agent IMMANUEL. In: HRI'17 Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction; 2017. p. 187–8.

[4] Lindner F, Wächert L, Bentzen MM. Discussion about lying with an ethical reasoning robot. In: Proceedings of the 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN); 2017. p. 1445–50.

1 Code maintained under: https://github.com/existenzquantor/actions

[5] Lindner F, Mattmüller R, Nebel B. Evaluation of the moral permissibility of action plans. Artificial Intelligence 2020; 287.
[6] Anscombe GEM. Intention. Oxford: Blackwell; 1963.
[7] Davidson D. Actions, reasons and causes. Journal of Philosophy 1963; 60:685–700.
[8] Dennis L, Fisher M, Slavkovik M, Webster M. Formal verification of ethical choices in autonomous systems. Robotics and Autonomous Systems 2016; 77:1–14.
[9] Bringsjord S, Taylor J. The divine-command approach to robot ethics. In: Robot ethics: The ethical and social implications of robotics; 2012. p. 85–108.
[10] Arkin R. Governing lethal behavior in autonomous robots. Chapmanand Hall/CRC; 2009.
[11] Lemaignan S, Ros R, Mösenlechner L, Alami R, Beetz M. ORO, a knowledge management module for cognitive architecture in robotics. In: Proceedings of the 2010 IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS); 2010. p. 3548–53.
[12] Tenorth M, Beetz M. KNOWROB – knowledge processing for autonomous personal robots. In: Proceedings of the 2009 IEEE/RSJ International Conference of Intelligent Robots and Systems (IROS); 2009. p. 4261–6.
[13] Weld DS, Etzioni O. The first law of robotics (a call to arms). In: Proceedings of the 12th National Conference on Artificial Intelligence (AAAI 1994); 1994. p. 1042–7.
[14] Alechina N, Halpern JY, Logan B. Causality, responsibility and blame in team plans. In: Proceedings of the 16th Conference on Autonomous Agents and Multi-Agent-Systems (AAAMAS 2017); 2017. p. 1091–9.
[15] Göbelbecker M, Keller T, Eyerich P, Brenner M, Nebel B. Coming up with good excuses: What to do when no plan can be found. In: Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS 2010); 2010. p. 81–8.
[16] Batusiv V, Soutchanski M. Situation calculus semantics for actual causality. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI-18); 2018.
[17] Gelfond M, Lifschitz V. Action languages. Linköping Electronic Articles in Computer and Information Science 1998; 3(16).
[18] Mackie P. Causing, delaying, and hastening: Do rains cause fires? Mind 1992; 403:483–500.
[19] Baader F, Nutt W. Basic description logics. In: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press; 2003. p. 47–100.
[20] Schild K. A correspondence theory for terminological logics: Preliminary report. In: Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91); 1991. p. 466–71.
[21] Glimm B, Horrocks I, Motik B, Stoilos G, Wang Z. Hermit: An OWL 2 reasoner. Journal of Automated Reasoning 2014; 53:245–69.