Three Implementations of the Categorical Imperative

Lavanya Singh

April 16, 2021

Contents

imports *Main*

1 Introduction

As artifical reasoners become increasingly powerful, computers become increasingly able to perform complex ethical reasoning. The field of machine ethics [18] is attractive for two reasons. First, the proliferation of artifically autonomous agents is creating and will continue to create a demand for ethical autonomous agents. These agents must be able to model and reason about complex ethical theories that withstand philosophical scrutiny. Second, just as automated mathematical reasoning gives mathematicians new powers and proof-finding ability, automated ethical reasoning is a tool that philosophers can use when reasoning about ethics. Many contradictions or paradoxes with an ethical system may not be immediately obvious to the human eye, but can be easily tested using an automated theorem prover.

Modelling ethics without sacrificing the intracacies and complexities of an ethical theory is a challenging computational and philosophical problem. Simple and intuitive computational approaches, such as encoding ethical rules as constraints in a constraint satisfaction problem, fail to capture the complexity of most philosophically plausible systems. On the other hand, it is not immediately clear how to formalize many complex moral theories, like virtue ethics.

Computational ethics also requires a sophisticated ethical theory to model. Constraint satisfaction systems often default to some version of utiliarianism, the principle of doing the most good for the most people. Alternatively, they model basic moral principles such as "do not kill," without modelling the theory that these principles originated from. Modelling a more complex ethical theory will not only enable smarter philosophical machines, it will also empower philosophers to study more complex ethical issues with the computer's help. The entire field of philosophy is devoted to developing and testing robust ethical theories. Plausible machine ethics must draw on plausible moral philosophy.

The ideal candidate ethical theory will be both philosophically interesting and easy to formalize. Kantian ethics, often described as "formal," has been often floated as such a candidate [15] [1]. The categorical imperative, Kant's universal law of morality, is a moral rule that can be used to guide action.

This project's objective is to automate reasoning about Kantian ethics. I present two different formalizations of Kant's categorical imperative implemented and tested in the Isabelle/HOL [13] theorem prover. Each formalization is modelled as an extension of Carmo and Jones' Dyadic Deontic Logic (DDL). The corresponding DDL formalization is then embedded in higher-order logic (HOL) and implemented in Isabelle. Section 3.1 implements and tests the naive formalization, a control group that is equivalent to DDL itself. Section 3.2 examines a more sophisticated formalization inspired by Moshe Kroy's partial formalization of thecategorical imperative.

I contribute implementations of two different interpretations of the categorical imperative, examples of how each implementation can be used to model and solve ethical scenarios, and tests that examine ethical and logical properties of the system, including logical consistency, consistency of obligation, and possibility of permissibility. The implementations themselves are usable models of ethical principles and the tests represent the kind of philosophical work that formalized ethics can contribute.

2 System Overview

The goal of this project is to automate sophisticated ethical reasoning. This requires three components. First, the choice of an ethical theory that is both intuitively attractive and lends itself to formalization. Second, the choice of formal logic to model the theory in. Third, the choice of automation engine to implement the formal model in. Section 2.1 introduces Kantian ethics, Section 2.2 explains Carmo and Jones's Dyadic Deontic Logic [6] as a base logic, and Section 2.3 presents the Isabelle/HOL implementation of the logic.

2.1 Kantian Ethics

Kantian ethics is an attractive choice of ethical theory to automate for two reasons. The first is that Kant's writings have been a major inspiration for much of Western analytic philosophy. In addition to the rich neo-Kantian program, almost all major philosophical traditions after Kant have engaged with his work. Much of Western libertarian political thought is inspired by Kant's deontology, and his ethics have bled into household ethical thought. Deontology is seen as one of the three major schools of Western normative philosophy.

Understanding the ethic's potential for formalization requires understanding Kant's system. Kant argues that if morality exists, it must take the form of an inviolable rule, which he calls the "categorical imperative." He presents three formulations of the categorical imperative, as well as a robust defense of them in his seminal work on ethics [9]. He argues that all three formulations are equivalent.

The first formulation of the categorical imperative is the "Formula of Universal Law."

Definition *(Formula of Universal Law)*

I ought never to act except in such a way that I could also will that my maxim should become a universal law [9]

A "maxim" is roughly a moral rule like "I can murder someone to take their job". "Willing" a maxim is equivalent to acting on that rule. The FUL creates a thought experiment called the universalizability test: to decide if a maxim is permissible, imagine what would happen if everyone willed that maxim. If your imagined world

yields a contradiction, the maxim is prohibited. Intuitively, the FUL asks the question, "What would happen if everyone did that?" [10].

The universalizability test makes the "formal" nature of Kant's ethics immediately clear. Formal logic has the tools to universalize a maxim (apply a universal quantifier) and to test for contradictions (test for inconsistency).

Kant also presents two additional formulations of the categorical imperative.

Definition *(Formula of Humanity)*

The Formula of Humanity (FUH) is to act in such a way "that you use humanity, whether in your own person or in the person of any other, always at the same time as an end, never merely as a means."[9]

Definition *(Kingdom of Ends)*

The third formulation of the categorical imperative states that "we should so act that we may think of ourselves as legislating universal laws through our maxims."[10]

The last two formulations are not as obviously formal as the FUL, but they can still be modelled in logic. Because Kantian ethics presents a series of rules, a logical system can encode the theory by modelling each rule as an axiom.

The above outline is a brief and incomplete picture of a rich philosophical tradition. Kantian scholars debate the meaning of each formulation of the categorical imperative and develop views far more nuanced than those above. For the purposes of this paper, I will adopt Kant's three original formulations presented above. Note additionally that Kantian ethics is widely disputed. I do not present a defense of Kant's ethic in this paper. My approach to formalizing ethics can be applied to other theories as well.

This paper aims to formalize Kant's ethic as faithfully as possible. This is an important choice. While it is tempting to modify or simplify an ethical theory in seemingly insignificant way, these choices often have ramifications. The entire field of neo-Kantian thought has been exploring versions of Kant's ethical theory for centuries. I will not attempt to present a radically new conception of Kant's ethic, but will instead draw on philosophical expertise regarding the content and justification of the categorical imperative.

2.2 Dyadic Deontic Logic

2.2.1 Deontic Logic

Traditional modal logics include $\Box$ operators to represent necessity. In simple modal logics like S5 [7] using the Kripke semantics, $\Box p$ is true at a world w if p is true at all of w's neighbors. These logics also usually contain the $\diamond$ operator, representing possibility, where $\diamond p \iff \sim \Box \sim p$. Additionally, modal logics support operators of propositional logic like $\sim, \wedge, \vee, \rightarrow$.

A deontic logic is a special kind of modal logic designed to reason about obligation.

Standard deontic logic [7] [11] essentially replaces $\Box$ with the obligation operator O, and $\diamond$ with the permissibility operator P. Using the Kripke semantics for O, Op is true at w if p is true at all ideal deontic alternatives to w. The O operator in SDL takes a single argument (the formula that is obligatory), and is thus called a monadic deontic operator.

While SDL is appreciable for its simplicity, it suffers a variety of well-documented paradoxes, including contrary-to-duty paradoxes [16]. In situations where duty is violated, the logic breaks down and produces paradoxical results. Thus, I use an improved deontic logic instead of SDL for this work.

2.2.2 Dyadic Deontic Logic

I use as my base logic Carmo and Jones's dyadic deontic logic, or DDL, which improves on SDL [6]. It introduces a dyadic obligation operator $O\{A|B\}$ to represent the sentence "A is obligated in the context B". This gracefully handles contrary-to-duty conditionals. The obligation operator uses a neighborhood semantics [17][12], instead of the Kripke semantics. Carmo and Jones define a function ob that maps from worlds to sets of sets of worlds. Intuitively, each world is mapped to the set of propositions obligated at that world, where a proposition p is defined as the worlds at which the p is true.

DDL also includes many other modal operators. In addition to $\Box$ and $\diamond$, DDL also has a notion of actual obligation and possible obligation, represented by operators O_a and O_p respectively. These notions are accompanied by the corresponding modal operators $\Box_a, \diamond_a, \Box_p, \diamond_p$. These operators use a Kripke semantics, with the functions av and pv mapping a world w to the set of corresponding actual or possible versions of w.

For more of fine-grained properties of DDL see [6] or this project's source code. DDL is quite a heavy logic and contains many modal operators that aren't necessary for my analysis. While this expressivity is powerful, it may also cause performance impacts. In particular, DDL has a large set of axioms involving quantification over complex higher-order logical expressions. Proofs involving these axioms will be computationally expensive. Benzmueller and Parent warned me that this may become a problem if Isabelle's automated proof tools begin to time out.

2.3 Isabelle/HOL Implementation

Isabelle/HOL is an interactive proof assistant [13] built on Haskell and Scala. It allows the user to define types, functions, definitions, and axiom systems. It has built-in support for both automatic and interactive/manual theorem proving.

I started my project by reimplementing Benzmueller, Farjami, and Parent's [2] [3] implementation of DDL in Isabelle/HOL. This helped me learn how to use Isabelle/HOL, and the implementation showcased in the next few sections demonstrates the power of Isabelle.

BFP use a shallow semantic embedding. This kind of embedding models the semantics of DDL as constants in HOL and axioms as constraints on DDL models. This document will contain a subset of my implementation that is particularly interesting and relevant to understanding the rest of the project. For the complete implementation, see the source code in `paper22.thy`.

2.3.1 System Definition

The first step in embedding a logic in Isabelle is defining the relevant terms and types.

typedecl i — i is the type for a set of worlds.

type-synonym $t = (i \Rightarrow bool)$ — t represents a set of DDL formulas.
— A set of formulae is defined by its truth value at a set of worlds. For example, the set "True" would be true at any set of worlds.

The main accessibility relation that I will use is the *ob* relation:

consts $ob::t \Rightarrow (t \Rightarrow bool)$ — set of propositions obligatory in this "context"
— ob(context)(term) is True if t is obligatory in this context

2.3.2 Axiomatization

For a semantic embedding, axioms are modelled as restrictions on models of the system. In this case, a model is specificied by the relevant accessibility relations, so it suffices to place conditions on the accessibility relations. These axioms can be quite unweildy, so luckily I was able to lift BFP's [2] implementation of Carmo and Jones's original axioms directly. Here's an example of an axiom:

and *ax-5d*: $\forall X\, Y\, Z.\ ((\forall w.\ Y(w) \longrightarrow X(w)) \wedge ob(X)(Y) \wedge (\forall w.\ X(w) \longrightarrow Z(w)))$
$\longrightarrow ob(Z)(\lambda w.(Z(w) \wedge \neg X(w)) \vee Y(w))$
— If some subset Y of X is obligatory in the context X, then in a larger context Z, any obligatory proposition must either be in Y or in Z-X. Intuitively, expanding the context can't cause something unobligatory to become obligatory, so the obligation operator is monotonically increasing with respect to changing contexts.

2.3.3 Syntax

The syntax that I will work with is defined as abbreviations. Each DDL operator is represented as a HOL formula. A formula defined with the `abbreviation` command is unfolded in every instance. While the shallow embedding is generally performant (because it uses Isabelle's original syntax tree), abbreviations may

possibly hurt performance. In some complicated proofs, we want controlled unfolding. Benzmueller and Parent told me that eventually, the performance cost of abbreviations can be mitigated using a definition instead.

Modal operators will be useful for my purposes, but the implementation is pretty standard.

abbreviation *ddlbox*::$t \Rightarrow t$ ($\Box$)
 where $\Box\, A \equiv \lambda w. \forall\, y.\, A(y)$
abbreviation *ddldiamond*::$t \Rightarrow t$ ($\Diamond$)
 where $\Diamond A \equiv \neg(\Box(\neg A))$

The most important operator for our purposes is the obligation operator.

abbreviation *ddlob*::$t \Rightarrow t \Rightarrow t$ ($O\{\text{-}|\text{-}\}$)
 where $O\{B|A\} \equiv \lambda\, w.\, ob(A)(B)$
— O$\{B|A\}$ can be read as "B is obligatory in the context A"

While DDL is powerful because of its support for a dyadic obligation operator, in many cases we need a monadic obligation operator. Below is some syntactic sugar for a monadic obligation operator.

abbreviation *ddltrue*::t ($\top$)
 where $\top \equiv \lambda w.\, True$
abbreviation *ddlob-normal*::$t \Rightarrow t$ ($O\ \{\text{-}\}$)
 where $(O\ \{A\}) \equiv (O\{A|\top\})$
— Intuitively, the context `True` is the widest context possible because `True` holds at all worlds.

Validity will be useful when discussing metalogical/ethical properties.

abbreviation *ddlvalid*::$t \Rightarrow bool$ ($\models\text{-}$)
 where $\models A \equiv \forall\, w.\, A\ w$

2.3.4 Syntactic Properties

One way to show that a semantic embedding is complete is to show that the syntactic specification of the theory (axioms) are valid for this semantics - so to show that every axiom holds at every world. BFP [2] provide a complete treatment of the completeness of their embedding, but I will include selected axioms that are particularly interesting here. This section also demonstrates many of the relevant features of Isabelle/HOL for my project.

Consistency

lemma *True* **nitpick** [*satisfy*,*user-axioms*,*format*=*2*] **by** *simp*
— Isabelle has built-in support for Nitpick, a model checker. Nitpick successfully found a model satisfying these axioms so the system is consistent.[8]
— Nitpick found a model for card i = 1:

Empty assignment

Nitpick [5] can generate models or countermodels, so it's useful to falsify potential theorems, as well as to show consistency. by simp indicates the proof method. In this case, simp indicates the Simplification proof method, which involves unfolding definitions and applying theorems directly. HOL has $True$ as a theorem, which is why this theorem was so easy to prove.

Modus Ponens

lemma *modus-ponens*: **assumes** $\models A$ **assumes** $\models (A \rightarrow B)$
shows $\models B$
using *assms*(*1*) *assms*(*2*) **by** *blast*
— Because I have not defined a "derivable" operator, inference rules are written using assumptions.
— The rule blast is a classical reasoning method that comes with Isabelle out of the box. [13]
— This is an example of a metalogical proof in this system using the validity operator.

Another relevant operator for our purposes is $\Box$, the modal necessity operator. In this system, $\Box$ behaves as an S5 modal necessity operator.

lemma *K*:
shows $\models ((\Box(A \rightarrow B)) \rightarrow ((\Box A) \rightarrow (\Box B)))$ **by** *blast*

lemma *T*:
shows $\models ((\Box A) \rightarrow A)$ **by** *blast*

lemma *5*:
shows $\models ((\Diamond A) \rightarrow (\Box(\Diamond A)))$ **by** *blast*

As mentioned earlier, the obligation operator is most interesting for my purposes. Here are some of its properties.

lemma *O-diamond*:
shows $\models (O\{A|B\} \rightarrow (\Diamond(B \wedge A)))$
using *ax-5b ax-5a*
by *metis*
— A is only obligatory in a context if it can possibly be true in that context. This is meant to prevent impossible obligations in a context.

lemma *O-nec*:
shows $\models (O\{B|A\} \rightarrow (\Box O\{B|A\}))$
by *simp*
— Obligations are necessarily obligated. This axiom is faithful to Kant's interpretation of ethics and is evidence of Deontic Logic's power in representing Kant's theory. Kant claimed that the categorical imperative was not contingent on any facts about the world, but instead a property of the concept of morality itself [9]. Under this view, obligation should not be world-specific.

Below is an example of a more involved proof in Isabelle. This proof was almost completely automatically generated. The property itself here is not very interesting for my purposes because I will rarely mix the dyadic and monadic obligation operators.

lemma *O-to-O*:
shows $\models(O\{B|A\}\rightarrow O\{(A\rightarrow B)|\top\})$
proof−
have $\forall X\ Y\ Z.\ (ob\ X\ Y \wedge (\forall w.\ X\ w \longrightarrow Z\ w)) \longrightarrow ob\ Z\ (\lambda w.(Z\ w \wedge \neg X\ w) \vee Y\ w)$
— I had to manually specify this subgoal, but once I did Isabelle was able to prove it automatically.
by (*smt ax-5d ax-5b ax-5b''*)
— Isabelle's proof-finding tool, Sledgehammer [14], comes with out-of-the-box support for smt solving [4].
thus *?thesis*
proof −
have *f1*: $\forall\ p\ pa\ pb.\ ((\neg\ (ob\ p\ pa)) \vee (\exists\ i.\ (p\wedge(\neg\ pb))\ i)) \vee (ob\ pb\ ((pb\wedge(\neg\ p))\vee\ pa))$
using ‹$\forall X\ Y\ Z.\ ob\ X\ Y \wedge (\models(X\rightarrow Z)) \longrightarrow ob\ Z\ (\ (Z\wedge(\neg\ X))\vee\ Y)$› **by** *force*
obtain *ii* :: $(i \Rightarrow bool) \Rightarrow (i \Rightarrow bool) \Rightarrow i$ **where**
$\forall\ x0\ x2.\ (\exists\ v3.\ (x2\wedge(\neg\ x0))\ v3) = (x2\wedge(\neg\ x0))\ (ii\ x0\ x2)$
by *moura*
then have $\forall\ p\ pa\ pb.\ ((\neg\ ob\ p\ pa) \vee (p\wedge(\neg\ pb))\ (ii\ pb\ p)) \vee ob\ pb\ (\ (pb\wedge(\neg\ p))\vee\ pa)$
using *f1* **by** *presburger*
then show *?thesis*
by *fastforce*
qed
— This entire Isar style proof was automatically generated using Sledgehammer.
qed

The implementation of DDL showcases some of the useful features of Isabelle. Abbreviations allow us to embed the syntax of DDL into HOL without defining an entire abstract sytax tree. Automated support for proof-finding using Sledgehammer makes proving lemmas trivial, and proving more complex theorems far easier. Nitpick's model finding ability is useful to check for consistency and create countermodels.

References

[1] M. M. Bentzen and F. Lindner. A formalization of kant's second formulation of the categorical imperative. *CoRR*, abs/1801.03160, 2018.

[2] C. Benzmüller, A. Farjami, and X. Parent. Dyadic deontic logic in hol: Faithful embedding and meta-theoretical experiments. In M. Armgardt, H. C. Nordtveit Kvernenes, and S. Rahman, editors, *New Developments in Legal Reasoning and Logic: From Ancient Law to Modern Legal Systems*, volume 23 of *Logic, Argumentation & Reasoning*. Springer Nature Switzerland AG, 2021.

[3] C. Benzmüller, X. Parent, and L. W. N. van der Torre. Designing normative theories of ethical reasoning: Formal framework, methodology, and tool support. *CoRR*, abs/1903.10187, 2019.

[4] J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending sledgehammer with smt solvers. In *Proceedings of the 23rd International Conference on Automated Deduction*, CADE'11, page 116–130, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] J. C. Blanchette and T. Nipkow. *Nitpick: A Counterexample Generator for Higher-Order Logic Based on a Relational Model Finder*, volume 6172, page 131–146. Springer Berlin Heidelberg, 2010.

[6] J. Carmo and A. Jones. Completeness and decidability results for a logic of contrary-to-duty conditionals. *J. Log. Comput.*, 23:585–626, 2013.

[7] M. J. Cresswell and G. E. Hughes. *A New Introduction to Modal Logic*. Routledge, 1996.

[8] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

[9] I. Kant. *Groundwork of the Metaphysics of Morals*. Cambridge University Press, Cambridge, 1785.

[10] C. Korsgaard. Kant's Formula of Universal Law. *Pacific Philosophical Quarterly*, 66:24–47, 1985.

[11] P. McNamara and F. Van De Putte. Deontic Logic. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2021 edition, 2021.

[12] R. MONTAGUE. Universal grammar. *Theoria*, 36(3):373–398, 1970.

[13] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher Order Logic*. Springer-Verlag Berlin Heidelberg, Berlin, 2002.

[14] L. Paulson and J. Blanchette. Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. 02 2015.

[15] T. M. Powers. Prospects for a kantian machine. *IEEE Intelligent Systems*, 21(4):46–51, 2006.

[16] D. Rönnedal. Contrary-to-duty paradoxes and counterfactual deontic logic. *Philosophia*, 47, 09 2019.

[17] D. Scott. Advice on modal logic. In K. Lambert, editor, *Philosophical Problems in Logic: Some Recent Developments*, pages 143–173. D. Reidel, 1970.

[18] S. Tolmeijer, M. Kneer, C. Sarasua, M. Christen, and A. Bernstein. Implementations in machine ethics. *ACM Computing Surveys*, 53(6):1–38, Feb 2021.