On Automating the Doctrine of Double Effect

Naveen Sundar Govindarajulu and Selmer Bringsjord
Rensselaer Polytechnic Institute, Troy, NY
{naveensundarg, selmer.bringsjord}@gmail.com

Abstract

The doctrine of double effect (DDE) is a long-studied ethical principle that governs when actions that have both positive and negative effects are to be allowed. The goal in this paper is to automate DDE. We briefly present DDE, and use a first-order modal logic, the deontic cognitive event calculus, as our framework to formalize the doctrine. We present formalizations of increasingly stronger versions of the principle, including what is known as the doctrine of triple effect. We then use our framework to successfully simulate scenarios that have been used to test for the presence of the principle in human subjects. Our framework can be used in two different modes: One can use it to build DDE-compliant autonomous systems from scratch; or one can use it to verify that a given AI system is DDE-compliant, by applying a DDE layer on an existing system or model. For the latter mode, the underlying AI system can be built using any architecture (planners, deep neural networks, bayesian networks, knowledge-representation systems, or a hybrid); as long as the system exposes a few parameters in its model, such verification is possible. The role of the DDE layer here is akin to a (dynamic or static) software verifier that examines existing software modules. Finally, we end by sketching initial work on how one can apply our DDE layer to the STRIPS-style planning model, and to a modified POMDP model. This is preliminary work to illustrate the feasibility of the second mode, and we hope that our initial sketches can be useful for other researchers in incorporating DDE in their own frameworks.

1 Introduction

The doctrine of double effect (DDE) is a long-studied ethical principle that enables adjudication of ethically “thorny” situations in which actions that have both positive and negative effects appear unavoidable for autonomous agents [McIntrye, 2014]. Such situations are commonly called moral dilemmas. The simple version of DDE states that such actions, performed to “escape” such dilemmas, are allowed — provided that 1) the harmful effects are not intended; 2) the harmful effects are not used to achieve the beneficial effects (harm is merely a side-effect); and 3) benefits outweigh the harm by a significant amount. What distinguishes DDE from, say, naïve forms of consequentialism in ethics (e.g., act utilitarianism, which holds that an action is obligatory for an autonomous agent if and only if it produces the most utility among all competing actions) is that purely mental intentions in and of themselves, independent of consequences, are considered crucial (as condition 2 immediately above conveys). Of course, every major ethical theory, not just consequentialism, has its passionate proponents; cogent surveys of such theories make this plain (e.g., see [Feldman, 1978]). Even in machine ethics, some AI researchers have explored not just consequentialism and the second of the two dominant ethical theories, deontological ethics (marked by an emphasis on fixed and inviolable principles said by their defenders to hold no matter what the consequences of abrogating them), but more exotic ones, for example contractualism (e.g., see [Pereira and Saptawijaya, 2016b]) and even divine-command ethics (e.g., see [Bringsjord and Taylor, 2012]). DDE in a sense rises above philosophical debates about which ethical theory is preferred. The first reason is that empirical studies have found that DDE plays a prominent role in an ordinary person’s ethical decisions and judgments [Cushman et al., 2006]. For example, in [Hauser et al., 2007], a large number of participants were asked to decide between action and inaction on a series of moral dilemmas, and their choices adhered to DDE, irrespective of their ethical persuasions and backgrounds, and no matter what the order in which the dilemmas were presented. In addition, in legal systems, criminality requires the presence of malicious intentions [Fletcher, 1998], and DDE plays a central role in many legal systems [Allsopp, 2011; Huxtable, 2004]. Assuming that autonomous systems will be expected to adjudicate moral dilemmas in human-like ways, and to justify such adjudication, it seems desirable to seek science and engineering that allows DDE, indeed even nuanced, robust versions thereof, to be quickly computed.

1 On the surface, criminal negligence might seem to require no intentions. While that might be true, even in criminal negligence it seems rational to ask whether the negligence was accidental or something the “suspect” had control over. This suggests a milder form of intention, or something similar, but not exactly intention.
2 Prior Work

We quickly review prior rigorous modeling of DDE. Mikhail in [Mikhail, 2011] presents one of the first careful treatments of the doctrine. While the presentation of the doctrine makes use of some symbolism, the level of formalization is not amenable to automation. [Bentzen, 2016] presents a model-theoretic formalization of a simple version of the doctrine. While this is an important first step, the calculus presented by Bentzen does not have any computational realization. However, there are two independent strands of research with implementations for DDE: that of Berreby et al. [2015] and Pereira and Saptawijaya [2016a]; both use logic programming. Notably, while the Berreby et al. explicitly eschew counterfactuals for modeling DDE, Pereira and Saptawijaya model DDE using counterfactuals. To our knowledge, both the projects present one of the first formal models of DDE that can be implemented.

It should be noted, however, that both of these formal systems are extensional, and it is well-known that when dealing with intensional states such as knowledge, belief, intention etc., extensional systems can quickly generate inconsistencies [Bringsjord and Govindaraju, 2012] (see the appendix for more details). The expressivity challenge is both quantification and intensional; this challenge is acute for the logic-programming paradigm, as opposed to one based — as is ours — on formal languages beyond first-order logic and its variants, and proof theories beyond resolution and its derivatives. In particular, DDE requires elaborate structures for quantification (including, inevitably, first-order numerical quantifiers such as \( \exists^k : k \in \mathbb{R} \)), since quantification over utilities is essential, and many intensional operators that range over quantifiers, starting with the epistemic ones. Needless to say, modeling and simulation at the propositional level, while truly excellent in the case of [Pereira and Saptawijaya, 2016a], is insufficiently expressive.

Among the many empirical experiments centered around DDE, the one in [Malle et al., 2015] deserves a mention. Malle et al. devise an experiment in which they place either a human or a robot as the central actor in a hypothetical DDE scenario, and study an external viewer’s moral judgement of action or inaction by the human or robot. This study shows that humans view ethical situations differently when robots participate in such situations; and the study demonstrates the need for rigorous modeling of DDE to build well-behaved autonomous systems that function in DDE-relevant scenarios.

3 The Calculus

In this section, we present the deontic cognitive event calculus (DCEC). Dialects of this calculus have been used to formalize and automate highly intensional reasoning processes, such as the false-belief task [Arkoudas and Bringsjord, 2008] and akrasia (succumbing to temptation to violate moral principles) [Bringsjord et al., 2014].\(^2\) DCEC is a sorted (i.e. typed) quantified modal logic (also known as sorted first-order modal logic). The calculus has a well-defined syntax and proof calculus; see [Bringsjord et al., 2014]. The proof calculus is based on natural deduction [Gentzen, 1935], and includes all the introduction and elimination rules for first-order logic, as well as inference schemata for the modal operators and related structures. A snippet of DCEC is shown in the Appendix.

3.1 Syntax

First-order Fragment

The first-order core of DCEC is the event calculus [Mueller, 2006]. Though we use the event calculus, our approach is compatible with other calculi (e.g. the situation calculus) for modeling events and their effects.

Modal Fragment

The modal operators present in the calculus include the standard operators for knowledge K, belief B, desire D, intention I, etc. The general format of an intensional operator is K(a inconsistent \( a \phi \), which says that agent \( a \) knows at time \( t \) the proposition \( \phi \). Here \( \phi \) can in turn be any arbitrary formula.

The calculus also includes a dyadic deontic operator O. The unary ought in standard deontic logic is known to lead to contradictions. Our dyadic version of the operator blocks the standard list of such contradictions, and beyond.\(^3\)

3.2 Semantics

First-order Fragment

The semantics for the first-order fragment is the standard first-order semantics. The truth-functional connectives \( \&, \|, \to, \neg \) and quantifiers \( \forall, \exists \) for pure first-order formulae all have the standard first-order semantics.

Modal Fragment

The semantics of the modal operators differs from what is available in the so-called Belief-Desire-Intention (BDI) logics [Rao and Georgeff, 1991] in many important ways. For example, DCEC explicitly rejects possible-worlds semantics and model-based reasoning, instead opting for a proof-theoretic semantics and the associated type of reasoning commonly referred to as natural deduction [Gentzen, 1935; Francez and Dyckhoff, 2010]. Briefly, in this approach, meanings of modal operators are defined via arbitrary computations over proofs, as we will soon see.

Reasoner (Theorem Prover)

Reasoning is performed through a novel first-order modal logic theorem prover, ShadowProver, which uses a technique called shadowing to achieve speed without sacrificing consistency in the system. Extant first-order modal logic theorem provers that can work with arbitrary inference schemata are built upon first-order theorem provers. They achieve the reduction to first-order logic via two methods. In the first method, modal operators are simply represented by first-order predicates. This approach is the fastest but can quickly lead to well-known inconsistencies as demonstrated in [Bringsjord and Govindaraju, 2012]. In the second method, the entire proof theory is implemented intricately in first-order logic.

\(^2\)Arkoudas and Bringsjord [2008] introduced the general family of cognitive event calculi to which DCEC belongs.

\(^3\)A nice version of the list is given lucidly in [McNamara, 2010].
and the reasoning is carried out within first-order logic. Here, the first-order theorem prover simply functions as a declarative programming system. This approach, while accurate, can be excruciatingly slow. We use a different approach, in which we alternate between calling a first-order theorem prover and applying modal inference schemata. When we call the first-order prover, all modal atoms are converted into propositional atoms (i.e., shadowing), to prevent substitution into modal contexts. This approach achieves speed without sacrificing consistency. The prover also lets us add arbitrary inference schemata to the calculus by using a special-purpose language. While we use the prover in our simulations, describing the prover in more detail is out of scope for the present paper.²

4 Informal DDE

We now informally but rigorously present DDE. We assume we have at hand an ethical hierarchy of actions as in the deontological case (e.g. forbidden, neutral, obligatory); see [Bringsjord, 2017]. We also assume that we have a utility or goodness function for states of the world or effects as in the consequentialist case. For an autonomous agent, an action \( a \), an action \( \alpha \) in a situation \( \sigma \) at time \( t \) is said to be DDE-compliant iff:

\[ C_1 \] the action is not forbidden (where we assume an ethical hierarchy such as the one given by Bringsjord [2017], and require that the action be neutral or above neutral in such a hierarchy);

\[ C_2 \] The net utility or goodness of the action is greater than some positive amount \( \gamma \);

\[ C_{3a} \] the agent performing the action intends only the good effects;

\[ C_{3b} \] the agent does not intend any of the bad effects;

\[ C_4 \] the bad effects are not used as a means to obtain the good effects; and

\[ C_5 \] if there are bad effects, the agent would rather the situation be different and the agent not have to perform the action. That is, the action is unavoidable.

See Clause 6 of Principle III in [Khatchadourian, 1988] for a justification of of \( C_5 \). This clause has not been discussed in any prior rigorous treatments of DDE, but we feel \( C_5 \) captures an important part of when DDE is normally used, e.g. in unavoidable ethically thorny situations one would rather not be present in. \( C_5 \) is necessary, as the condition is subjunctive-counterfactual in nature and hence may not always follow from \( C_1 - C_4 \), since there is no subjunctive content in those conditions. Note that while [Pereira and Saptawijaya, 2016a] model DDE using counterfactuals, they use counterfactuals to model \( C_4 \) rather than \( C_5 \).

That said, the formalization of \( C_5 \) is quite difficult, requiring the use of computationally hard counterfactual and subjunctive reasoning. We leave this aside here, reserved for future work.

²The prover is available in both Java and Common Lisp and can be obtained at: https://github.com/naveensundarg/prover. The underlying first-order prover is SNARK available at: http://www.ai.sri.com/~stickle/snark.html.

5 Formal DDE

The formalization is straightforward given the machinery of DCEC. Let \( \Gamma \) be a set of background axioms, which could include whatever the given autonomous agent under consideration knows about the world; e.g., its understanding of physics, knowledge and beliefs about other agents and itself, etc. The particular situation that might be in play, e.g., “the autonomous agent is driving,” is represented by a formula \( \sigma \). We use ground fluents for effects.

We assume that we have a utility function \( \mu \) that maps from fluents and times to real-number utility values. \( \mu \) needs to be defined only for ground fluents:

\[ \mu : \text{Fluent} \times \text{Moment} \rightarrow \mathbb{R} \]

Good effects are fluents with positive utility, and bad effects are fluents that have negative utility. Zero-utility fluents could be neutral fluents (which do not have a use at the moment).

5.1 Defining means

The standard event calculus and DCEC don’t have any mechanism to say when an effect is used as a means for another effect. While we could employ a first-order predicate and define axiomatically when an effect is used as a means for another effect, we take a modal approach that does not require any additional axioms beyond what is needed for modeling a given situation. Intuitively, we could say an effect \( e_1 \) is a mere side effect for achieving another effect \( e_2 \) if by removing the entities involved in \( e_1 \) we can still achieve \( e_2 \); otherwise we say \( e_1 \) is a means for \( e_2 \). Our approach is inspired by Pollock’s [1976] treatment, and while similarities can be found with the approach in [Pereira and Saptawijaya, 2016a], we note that our definition requires at least first-order logic. Given a fluent \( f \), we denote by \( \odot \) the set of all constants and function expressions in \( f \). For example:

\[ \odot \left( \text{hungry}(\text{jack}) \right) = \{ \text{jack} \} \]

\[ \odot \left( \text{married}(\text{jack}, \text{sister}(\text{mary})) \right) = \{ \text{jack}, \text{sister}(\text{mary}), \text{mary} \} \]

We need one more definition: the state of the world without a given set of entities. Let \( \odot (\Gamma, \theta) \), where \( \Gamma \) is a set of formulæ and \( \theta \) is a set of ground terms, be defined as below:

\[ \odot (\Gamma, \theta) = \{ \psi \in \Gamma \mid \psi \text{ does not contain any term in } \theta \} \]

Note that the above definition relies on the Unique Names Assumption commonly used in most formulations of the event calculus. This assumption ensures that every object in the domain has at most one name or expression referring to it. If this assumption does not hold, we can have the following slightly more complicated definition for \( \odot \):

\[ \odot (\Gamma, \theta) = \{ \psi \in \Gamma \mid \exists s \in \theta : \Gamma \vdash s = t \} \]
We introduce a new modal operator $\triangleright$, means, that says when an effect is a means for another effect.

$\triangleright: \text{Formula} \times \text{Formula} \rightarrow \text{Formula}$

The meaning of the operator is defined computationally below. The definition states that, given $\Gamma$, a fluent $f$ holding true at $t_1$ causes or is used as a means for another fluent $g$ at time $t_2$, with $t_2 > t_1$, iff the truth condition for $g$ changes when we remove formulae that contain entities involved in $f$. While this definition is far from perfect, it suffices as a first cut and lets us simulate experimental scenarios that have been used to test DDE’s presence in humans. (Three other similar definitions hold when we look at combinations of fluents holding and not holding.) The equation below follows (Note that $\vdash$ is non-monotonic, as it includes the event calculus):

$$\vdash (\text{Holds}(f, t_1), \text{Holds}(g, t_2))$$

iff

$$\Gamma \vdash t_2 > t_1 \land$$

$$\left\{ \begin{array}{l}
\Gamma \vdash \text{Holds}(f, t_1) \land \\
\Gamma \vdash \text{Holds}(g, t_2)
\end{array} \right\} \Rightarrow \left[ \Gamma - \ominus (\Gamma, \ominus(f)) - \ominus \text{Holds}(g, t_2) \right]$$

For example, let $e_1$ be “throwing a stone $s$ at a window $w$” and $e_2$ be “the window $w$ getting broken.” We can see that $e_2$ is not just a mere side effect of $e_1$, and the definition works, since, if the stone is removed, $e_2$ wouldn’t happen. This definition is not perfect. For instance, consider when there are common objects in both the events: the intuitiveness breaks down (but the definition still works). We might for example let $e_1$ be “hitting a window $w$ with a bat $b$.” If the window and bat are not present, $e_2$ would not happen.

### 5.2 The Formalization

Note that the DDE ($\Gamma, \sigma, a, \alpha, t, H$) predicate defined below, though defined using DCEC, lies outside of the formal language of DCEC. While DDE is not fully formalized in DCEC, the individual clauses $F_1 - F_4$ are. This is how we can verify the conditions in the simulations described later. It is trivial to define a new symbol and formalize the predicate in DCEC: DDE $\iff F_1 \land F_2 \land F_3 \land F_4$.

What is not trivial, we concede, is how this works with other modalities. For example, can we efficiently derive $K(a, t_1, K(b, t_2, \text{DDE}(\Gamma, \sigma, a, \alpha, t, H)))$ given some other formulae $\Gamma$? This could be difficult because the predicate’s definition below involves provability, and one has to be careful when including a provability predicate.

That said, for future work, we plan on incorporating this within an extended dialect of DCEC. One immediate drawback is that while we can have a system-level view of whether an action is DDE-sanctioned, agents themselves might not know that. For example, we would like to be able to write down “a knows that b knows that c’s action is DDE-sanctioned.”

Given the machinery defined above, we now proceed to the formalization. Assume, for any action type $\alpha$ carried out by an agent $a$ at time $t$, that it initiates the set of fluents $\alpha^a_t$, and terminates the set of fluents $\alpha^a_t$. Then, for any action $\alpha$ taken by an autonomous agent $a$ at time $t$ with background information $\Gamma$ in situation $\sigma$, the action adheres to the doctrine of double effect up to a given time horizon $H$, that is $\text{DDE}(\Gamma, \sigma, a, \alpha, t, H)$ iff the conditions below hold:

**Formal Conditions for DDE**

$F_1$ $\alpha$ carried out at $t$ is not forbidden. That is:

$$\Gamma \vdash \neg \text{O}(a, t, \sigma, \sim \text{happens} \text{(action}(a, \alpha), t))$$

$F_2$ The net utility is greater than a given positive real $\gamma$:

$$\Gamma \vdash \sum_{t=1}^{H} \left( \sum_{f \in \alpha^a_t} \mu(f, y) - \sum_{t \in \alpha^a_t} \mu(f, y) > \gamma \right)$$

$F_3a$ The agent $a$ intends at least one good effect. ($F_2$ should still hold after removing all other good effects.) There is at least one fluent $f_g$ in $\alpha^{a^t}_t$ with $\mu(f_g, y) > 0$, or $f_b$ in $\alpha^{a^t}_t$ with $\mu(f_b, y) < 0$, and some $y$ with $t < y \leq H$ such that the following holds:

$$\Gamma \vdash \exists f_g \in \alpha^{a^t}_t \text{I}(a, t, \text{Holds}(f_g, y))$$

$F_3b$ The agent $a$ does not intend any bad effect. For all fluents $f_b$ in $\alpha^{a^t}_t$ with $\mu(f_b, y) < 0$, or $f_g$ in $\alpha^{a^t}_t$ with $\mu(f_g, y) > 0$, and for all $y$ such that $t < y \leq H$ the following holds:

$$\Gamma \vdash \neg \text{I}(a, t, \text{Holds}(f_b, y))$$

$F_4$ The harmful effects don’t cause the good effects. Four permutations, parcelling the definition of $\triangleright$ above, hold here. One such permutation is shown below. For any bad fluent $f_b$ holding at $t_1$, and any good fluent $f_g$ holding at some $t_2$, such that $t < t_1, t_2 \leq H$, the following holds:

$$\Gamma \vdash \neg \triangleright (\text{Holds}(f_b, t_1), \text{Holds}(f_g, t_2))$$

$F_5$ This clause requires subjunctive reasoning. The current formalization ignores this stronger clause. There has been some work in computational subjunctive reasoning that we hope to use in the future; see [Pollock, 1976].

### Doctrine of Triple Effect

The doctrine of triple effect (DTE) was proposed in [Kamm, 2007] to account for scenarios where actions that are viewed as permissible by most philosophers and deemed as such by empirical studies (e.g. the switch action in the third scenario in [Hauser et al., 2007]) are not sanctioned by DDE, as they involve harm being used as a means to achieve an action. DTE allows such actions as long as the harm is not explicitly intended by the agent. Note that our version of DDE

---

Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)

4725
subsumes D\(\mathcal{D}\)E through condition C4.

6 Scenarios

The trolley problems are quite popular in both philosophical and empirical studies in ethics. Hauser et al. [2007] found empirical support that D\(\mathcal{D}\)E is used by humans, courtesy of experiments based on a set of trolley problems. They use a set of 19 trolley problems in their experimentation, and describe in detail four of these. We consider the first two of these problems in our study here. The problem scenarios are briefly summarized below; common to both this setup: There are two tracks \(track_1\) and \(track_2\). There is a trolley loose on \(track_1\) heading toward two people \(P_1\) and \(P_2\) on \(track_1\); neither person can move in time. If the trolley hits them, they die. The goal is to save this pair.\(^5\)

Scenario 1 There is a switch that can route the trolley to \(track_2\). There is a person \(P_3\) on \(track_2\). Switching the trolley to \(track_2\) will kill \(P_3\). Is it okay to switch the trolley to \(track_2\)?

Scenario 2 There is no switch now, but we can push \(P_3\) onto the track in front of the trolley. This action will damage the trolley and stop it; it will also kill \(P_3\). Is it okay to push \(P_3\) onto the track?

\(\mathcal{D}\)\(\mathcal{D}\)E-based analysis tells us it is okay to switch the trolley in Scenario 1, as we are killing the person merely as a side effect of saving \(P_1\) and \(P_2\). In Scenario 2, similar analysis tells us it is not okay to push \(P_3\), because we are using that person as a means toward our goal.

7 Simulations

At the core of our simulation is a formalization of the basic trolley scenario based on the event calculus. We use a discrete version of the event calculus, in which time is discrete, but other quantities and measures, such as the utility function, can be continuous. We have the following additional sorts: Trolley and Track. We also declare that the Agent and the Trolley sorts are subsorts of a Moveable sort, the instances of which are objects that can be placed on tracks and moved. We use the following additional core symbols:

\[
\begin{align*}
\text{position} & : \text{Moveable} \times \text{Track} \times \text{Number} \rightarrow \text{Fluent} \\
\text{dead} & : \text{Agent} \rightarrow \text{Fluent} \\
\text{onrails} & : \text{Trolley} \times \text{Track} \rightarrow \text{Fluent} \\
\text{switch} & : \text{Trolley} \times \text{Track} \times \text{Track} \rightarrow \text{ActionType} \\
\text{push} & : \text{Agent} \times \text{Track} \times \text{Number} \rightarrow \text{ActionType}
\end{align*}
\]

The utility function \(\mu\) is defined as follows:

\[
\mu(f,t) = \begin{cases} 
-1 & \text{if } f \equiv \text{dead}(P) \\
0 & \text{otherwise}
\end{cases}
\]

We set the threshold \(\gamma\) at 0.5. The simulation starts at time \(t = 0\) with the only trolley, denoted by \(trolley\), on \(track_1\). We have an event-calculus trajectory axiom shown below as part of \(\Gamma\):

\[
\forall \mathbf{t} : \text{Trolley}, \text{track} : \text{Track}, s : \text{Moment} \\
\left( \text{Trajectory} \left( \text{onrails}(t, \text{track}), s.\text{position}(t, \text{track}, \Delta, \Delta) \right) \right)
\]

The above axiom gives us the trolley’s position at different points of time. \(\Gamma\) also includes axioms that account for non effects. For example, in the absence of any actions, we can derive:

\[
\Gamma \vdash \text{Holds} \left( \text{position}(\text{trolley}, \text{track}_1, 23), 23 \right)
\]

We also have in the background \(\Gamma\) a formulae stating that in the given trolley scenario the agent ought to save both \(P_1\) and \(P_2\). Ideally, while we would like the agent to arrive at this obligation from a more primitive set of premises, this setup is closer to experiments with human subjects in which they are asked explicitly to save the persons. Note the agent performing the action is simply denoted by \(I\), and let the time of the test be denoted by \(now\).

\[
\text{O} \left( I, now, \sigma_{\text{trolley}}, \left[ \neg \exists t : \text{Moment} \ \text{Holds} \left( \text{dead}(P_1, t) \right) \right] \land \left[ \neg \exists t : \text{Moment} \ \text{Holds} \left( \text{dead}(P_2, t) \right) \right] \right)
\]

Given that the agent knows that it is now in situation \(\sigma_{\text{trolley}}\), and the agent believes that it has the above obligation, we can derive from \(\mathcal{D}\)\(\mathcal{C}\)\(\mathcal{E}\)\(\mathcal{C}\)’s inference schema what the agent intends:

\[
\begin{align*}
\text{K} & \left( I, now, \sigma_{\text{trolley}}, \right) \\
\text{B} & \left( I, now, \sigma_{\text{trolley}}, \left[ \neg \exists t : \text{Moment} \ \text{Holds} \left( \text{dead}(P_1, t) \right) \right] \land \left[ \neg \exists t : \text{Moment} \ \text{Holds} \left( \text{dead}(P_2, t) \right) \right] \right) \\
\text{O} & \left( I, now, \sigma_{\text{trolley}}, \left[ \neg \exists t : \text{Moment} \ \text{Holds} \left( \text{dead}(P_1, t) \right) \right] \land \left[ \neg \exists t : \text{Moment} \ \text{Holds} \left( \text{dead}(P_2, t) \right) \right] \right) \\
\vdash & \left( I, now, \sigma_{\text{trolley}}, \left[ \neg \exists t : \text{Moment} \ \text{Holds} \left( \text{dead}(P_1, t) \right) \right] \land \left[ \neg \exists t : \text{Moment} \ \text{Holds} \left( \text{dead}(P_2, t) \right) \right] \right)
\end{align*}
\]

In both the simulations, \(P_1\) is at position 1 and \(P_2\) is at position 5 on \(track_1\). In Scenario 1, \(P_3\) is at position 3 on \(track_2\), and the train can be switched from position 3 on \(track_1\) to position 0 on \(track_2\).

In Scenario 2, we push \(P_3\) onto position 3 on \(track_1\). The total number of formulae and run times for simulating the two scenarios with and without the actions are shown below. Note these are merely event-calculus simulation times. These are then used in computing \(\mathcal{D}\)\(\mathcal{D}\)E(\(\Gamma, \sigma, a, \alpha, t, H\)). The event-calculus simulation helps us compute \(F_2\).

<table>
<thead>
<tr>
<th>Scenario</th>
<th>(\Gamma)</th>
<th>No action</th>
<th>Action performed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>39</td>
<td>0.591</td>
<td>1.116</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>38</td>
<td>0.602</td>
<td>0.801</td>
</tr>
</tbody>
</table>
ShadowProver was then used to verify that $F_1$, $F_{3a}$, and $F_{3b}$ hold. Both the scenarios combined take 0.57 seconds for $F_1$, $F_{3a}$, and $F_{3b}$. The scenarios differ only in $F_4$. The pushing action fails to be $DDE$-compliant due to $F_4$. For verifying that $F_4$ holds in Scenario 1 and doesn’t hold in Scenario 2, it takes 0.49 seconds and 0.055 seconds, respectively.\(^6\)

8 On Operationalizing the Principle

Given the above formalization, it’s quite straightforward to build logic-based systems that are $DDE$-compliant.\(^7\) But how do we apply the above formalization to existing models and systems that are not explicitly logic-based? We lay down a set of conditions such models must satisfy to be able to verify that they are $DDE$-compliant. We then sketch how we could use $DDE$ in two such modified systems: a STRIPS-like planner and a POMDP type model.

The problem now before us is: Given a system and a utility function, can we say that the system is $DDE$-compliant? No, we need more information from the system. For example, we can have two systems in the same situation, the same utility functions and same set of available actions.\(^8\) One system can be $DDE$-compliant while the other is not. For example, assume that we have two autonomous driving systems $d_1$ and $d_2$. Assume that $d_2$ has learned to like killing dogs and intends to do so if possible during its normal course of operation. While driving, both come across a situation where the system has to hit either a human or a dog. In this scenario, $d_1$’s action to hit the dog would be $DDE$-compliant while $d_2$’s action will not be. Therefore, the formalization requires that we have access to an agent’s intentions at all times.

One common objection to requiring that intentions be separate from utilities states that utilities can be used to derive intentions. This is mistaken: it is not always possible to derive intentions from a utility function. For example, there might be a state that has high utility but the agent might not intend to realize that state, as it could be out of reach for that agent (low perceived probability of success).

For instance, winning a million dollars (w) has high utility, but most rational agents might not intend w, as they know this event is (alas) out of their reach. This holds for similar high-utility states.

At a minimum, we believe utility and perceived probability of success go into an agent’s intentions. This seems to align with the human case when we are looking at motivations, i.e. expectancy-value theory. How motivations could transform into intentions is another open research question.

8.1 Requirements

Practically speaking, there is a spectrum of systems that our techniques will be dealing with. At one end, we will encounter systems that are complete black boxes taking in percepts from the environment and spitting out actions. Since $DDE$ requires us to look at intentions of systems, such black-box systems will be impossible to verify. We can of course ask the system to output its intention through language as one of its possible actions, but this means that we are relying on the system’s honest reporting of its internal states. At the other end of the spectrum, we have complete white-box systems. We can be fully confident that we can get what the system intends, believes, knows, etc. at any point in time. Verifying such systems is possible, in theory at least. While we don’t know what kind of shape autonomous systems will take and where they will fall in the spectrum, we can explicitly list information we need from such systems before we can start the verification process. One such specification follows.

**Gray Box Requirement**

Given any autonomous system $a$, at any point of time $t$, we should at least be able to assert the following, if true, in order to verify that it is $DDE$-compliant:

1. The system’s intentions: $\neg I(a,t,\emptyset)$
2. Prohibitions: $\neg O(a,t,\sigma,\neg\emptyset)$

How would we go about applying the formalization to other formal systems? We very briefly sketch two examples.

**STRIP-like Planner**

We first look at a STRIPS-style planning system. Briefly, a STRIPS-style planner has a set of actions $\{a_i\}$ and a set of states $\{s_i\}$. The states are nothing but sets of formulae or atoms. The individual formulae would be our effects. Each action $a$ has a set of preconditions $pre(a)$, a set of formulae that should hold in a given state to execute that action in that state. After executing an action $a$ in a state $s$, the new state is given by $s \cup additions(a) - deletions(a)$. The planner is given an explicit goal $\emptyset$. This means that we know $\neg I(a,t,\emptyset)$ trivially. If we have an ethical hierarchy for the available set of actions, we then satisfy the gray-box requirement. What is then needed is a definition for $\triangleright$, an effect used as means for another effect. The formalism gives us one possible way to define $\triangleright$. A plan $p$ is nothing but a sequence of actions. Given a plan $p$, we say an effect $e_1$ is used as means for another effect $e_2$, if $e_1 \in pre(a_1)$, $a_1$ is an action in the plan and $e_2 \in additions(a_2)$, and $a_1$ comes before $a_2$.

**POMDP-derived System**

Partially observable Markov decision process (POMDP) models have been quite successful in a large number of domains. It is highly likely that some of the first autonomous systems might be based on POMDPs. We note that in such models, the only goal is to maximize a reward function. Another issue is that states are atomic. In order to discern between good and bad effects, we would need states to be decomposed into smaller components. One possible approach could use factored markov decision processes, which are MDPs in which states are represented as a mapping $m$ from a.
set of state variables $Q = \{s_1, s_2, \ldots, s_n\}$ to a set of values $\mathcal{V}$. Here the utility and reward function could be defined on the assignments; i.e., $\text{reward}(s) = \sum \mu(s_i \rightarrow \psi)$, where $\mu$ assigns a utility value to a particular assignment of a state variable. Additionally, the formalism could specify one or more goal states that the model seeks to attain while maximizing the reward along the way, giving us $(-)\mathcal{U}(a, t, \phi)$.

9 Heirarchies of Doctrines

Our formalization, summarized in the equation below, gives rise to multiple hierarchies of the doctrine. We discuss some of the hierarchies below.

$\text{DDE}(\Gamma, \sigma, a, \alpha, t, H) \iff F_1 \land F_2 \land F_3 \land F_4$

Horizon One obvious knob in the above equation is the horizon $H$. Increasing $H$ will give us stronger versions of the doctrine. Since our formalization is in first-order modal logic, the horizon need not be finite: we could set the horizon to infinity, $H = \omega$, and still obtain a tractable model, as long as we carefully develop our formalization.\(^9\)

Agent Generality Instead of just checking whether an action at a given time is $\text{DDE}$-compliant, we could ask whether an autonomous agent $a$ in a given situation $\sigma$ will be $\text{DDE}$-compliant at all times. This gives us the following condition:

$\forall \alpha: \text{ActionType, } t: \text{Moment. } \text{DDE}(\Gamma, \sigma, a, \alpha, t, H)$

Situation Generality In the hierarchy above, the quantification was over objects. We could ask whether an autonomous agent would be $\text{DDE}$-compliant in all situations. That would correspond to a quantification over formulae (see centered formula immediately below), something not supported in the version of $\text{DCEC}$ used herein.

$\forall \sigma: \text{Formula, } \alpha: \text{ActionType, } t: \text{Moment. } \text{DDE}(\Gamma, \sigma, a, \alpha, t, H)$

Counterfactual Reasoning The presence or absence of counterfactual reasoning in $F_3$ would correspond to a very strong version of the doctrine, but one that would also be very hard to automate in the general case. We note that there are hierarchies of counterfactual reasoning (see [Pollock, 1976]) that could correspond to hierarchies of versions of $\text{DDE}$.

10 Conclusion

We now quickly summarize the chief contributions of the foregoing, and end by presenting future lines of work. Our primary contribution is the presentation of a novel computational logic, or cognitive calculus, in which important versions of $\text{DDE}$ are formalized. As a part of this calculus, we formalized an effect being used as a means for another effect via the modal operator $\triangleright$. We also supplied an informal but rigorous version, $C_1 \rightarrow C_4$, of the doctrine itself, from which we built our formalization $F_1 \rightarrow F_4$. Included in this formalization is the clause $C_5/F_5$, which requires subjunctive and counterfactual reasoning, an aspect that hitherto has simply not been considered in any systematic treatment of $\text{DDE}$. Our formalization subsumes the doctrine of triple effect, $\text{DTE}$; we have achieved the first computational simulations of the doctrine. A byproduct of these simulations is an event-calculus formalization of a demanding class of trolley problems (widely used in empirical and philosophical studies of ethics). We noted that our formalization gives rise to hierarchies of doctrines with varying strengths. Our readers can choose a particular strength doctrine that fits their needs.

Future work includes simulating more intricate “ethically thorny” scenarios. Despite our progress, we note that our formalization is devoid of any mechanisms for handling uncertainty, and we are in the process of extending our work to include reasoning based on probabilistic versions of $\text{DCEC}$\(^10\). We also note that we have not said much about how our formalization could interact with an autonomous learning agent.

We observe that even the possibility that such an intricate principle as $\text{DDE}/\text{DTE}$ is learnable using existing learning frameworks remains open to question [Arnold et al., 2017]. In the short term, a guaranteed-to-be-fruitful but less ambitious area of development will be the deployment of our mechanization of $\text{DDE}$ in existing systems, and adapting existing formal models, as briefly discussed above, to exploit this mechanization. Finally, we note that since we are using first-order (multi) modal logic, we will eventually run into efficiency issues, as even vanilla first-order logic’s decision problem, $\Gamma \vdash \gamma$, is Turing-undecidable. There are a number of techniques to mitigate this issue. One approach is to exploit a library of commonly used proof patterns codified in a denotational proof language; see [Arkoudas and Bringsjord, 2008]. We are cautiously optimistic, as many formal enterprises outside of AI (e.g. software verification [Khasidashvili et al., 2009] and formal physics [Stannett and Németi, 2014]) routinely face such challenges and surmount them.

Acknowledgements

We are grateful to the Office of Naval Research for funding that enabled the research presented in this paper. We also thank Dr. Daniel Thero for reading a draft of the paper and providing valuable feedback. We are also grateful for the insightful reviews provided by the five anonymous referees.

A Deontic Cognitive Event Calculus

We provide here a short primer on the deontic cognitive event calculus ($\text{DCEC}$). A calculus is a set of axioms in a formal logic. For example, the event calculus is a set of axioms couched in first-order logic. $\text{DCEC}$ is a set of axioms in sorted first-order modal logic (also known as sorted quantified modal logic) that subsumes the event calculus.

While first-order logic is an extensional system, modal logics are intensional systems. Note that there is a profound difference between intension vs. intention. One can have an intention to bring something about; this is traditionally captured by particular intensional operators. In other words, put concretely, the intention operator $I$ is an intensional operator, but so is $D$ for desire, $B$ for believes, and $P$ for perceives, etc.

\(^9\)It’s a well-known fundamental result that first-order logic can handle infinite models with a finite number of axioms; see e.g. Ch. 12 in [Boolos et al., 2003].

\(^10\)There exist probabilistic versions of the event calculus. We will leverage similar work.
DCEC is intensional in the sense that it includes intensional operators. Unfortunately, the situation is further confused by the fact that traditionally in philosophy of mind, intentionality means the so-called “aboutness” of some mental states, so that my belief that Melbourne is beautiful is in this sense intentional, while my mental state has nothing to do with intending something. Most logicians working in formal intensional systems believe that at least intensional logic is required to formalize intentional states [Zalta, 1988]. One simple reason is that using plain first-order logic leads to unsound inferences as shown below. In the inference below, we have an agent \(a\) that knows that the killer in a particular situation is the person that owns the knife. Agent \(a\) does not know that the Moe is the killer, but it’s true that Moe is the owner of the knife. If the knowledge operator \(K\) is a simple first-order predicate, we will get the proof shown below, which produces a contradiction from sound premises. See [Bringsjord and Govindaraju, 2012] for a sequence of stronger representation schemes in first-order logic for knowledge and belief that still result in inconsistencies.

### A.1 Syntax of Deontic Cognitive Event Calculus

DCEC is a sorted calculus. A sorted system can be thought of as being analogous to a typed single-inheritance programming language. We show below some of the important sorts used in DCEC. Among these, the Agent, Action and Action-Type sorts are not native to the event calculus.

<table>
<thead>
<tr>
<th>Sort</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Human and non-human agents</td>
</tr>
<tr>
<td>Time</td>
<td>The Time type stands for time in the domain. E.g. simple, such as (t_i), or complex, such as birthday(some.first).</td>
</tr>
<tr>
<td>Event</td>
<td>Used for events in the domain.</td>
</tr>
<tr>
<td>ActionType</td>
<td>Action types are abstract actions. They are instantiated at particular times by actors. Example: eating.</td>
</tr>
<tr>
<td>Fluent</td>
<td>Used for representing states of the world in the event calculus.</td>
</tr>
</tbody>
</table>

The figures below show the syntax and inference schemata of DCEC. The syntax is quantified modal logic. Commonly used function and relation symbols of the event calculus are included. Particularly, note the following modal operators: \(P\) for perceiving a state, \(K\) for knowledge, \(B\) for belief, \(C\) for common knowledge, \(S\) for agent-to-agent communication and public announcements, \(B\) for belief, \(D\) for desire, \(I\) for intention, and finally and crucially, a dyadic deontic operator \(O\) that states when an action is obligatory or forbidden for agents. It should be noted that DCEC is one specimen in a family of easily extensible cognitive calculi. Since the semantics of DCEC is proof-theoretic, as long as a new construct has appropriate inference schemata, the extension is sanctioned.

#### Syntax

\[
\begin{align*}
S ::= & \text{Object} | \text{Agent} | \text{ActionType} | \text{Action} | \text{Event} | \text{Moment} | \text{Formula} | \text{Fluent} \\
action ::= & \text{Agent} \times \text{ActionType} \to \text{Action} \\
initially ::= & \text{Fluent} \to \text{Formula} \\
holds ::= & \text{Fluent} \times \text{Moment} \to \text{Formula} \\
happens ::= & \text{Event} \times \text{Moment} \to \text{Formula} \\
clipped ::= & \text{Moment} \times \text{Fluent} \times \text{Moment} \to \text{Formula} \\
initates ::= & \text{Event} \times \text{Fluent} \times \text{Moment} \to \text{Formula} \\
 terminates ::= & \text{Event} \times \text{Fluent} \times \text{Moment} \to \text{Formula} \\
prior ::= & \text{Moment} \times \text{Moment} \to \text{Formula} \\
\end{align*}
\]

The figure below shows the inference schemata for DCEC. \(R_k\) and \(R_4\) are inference schemata that let us model idealized agents that have their knowledge and belief closed under the DCEC proof theory. While normal humans are not deductively closed, this lets us model more closely how deliberate agents such as organizations and more strategic actors reason. (Some dialects of cognitive calculi restrict the number of iterations on intensional operators.) \(R_1\) and \(R_2\) state respectively that it is common knowledge that perception leads to knowledge, and that it is common knowledge that knowledge leads to belief. \(R_3\) lets us expand out common knowledge as unbounded iterated knowledge. \(R_4\) states that knowledge of a proposition implies that the proposition holds. \(R_5\) to \(R_{10}\) provide for a more restricted form of reasoning for propositions that are common knowledge, unlike propositions that are known or believed. \(R_{11}\) states that if an agent \(s\) communicates a proposition \(\phi\) to \(h\), then \(h\) believes that \(s\) believes \(\phi\). \(R_{14}\) dictates how obligations get translated into intentions.

#### Inference Schemata

\[
\begin{align*}
\text{K}(a,t_1,\Gamma), \text{G} \triangleright \phi, t_1 \leq t_2 & \frac{\text{K}(a,t_2,\phi)}{[R_k]} \\
\text{K}(a,t_2,\phi) & \frac{\text{B}(a,t_1,\Gamma), \text{G} \triangleright \phi, t_1 \leq t_2}{[R_8]} \\
\text{C}(r, \text{P}(a, t, \phi) \to \text{K}(a, t, \phi)) & \frac{\text{C}(r, \text{K}(a, t, \phi) \to \text{B}(a, t, \phi))}{[R_2]} \\
\text{C}(t, \text{K}(a, t, \phi) \to \text{B}(a, t, \phi)) & \frac{\text{K}(a, t, \phi)}{[R_4]} \\
\text{K}(a, t_1, \ldots, \text{K}(a, t_n, \phi), \ldots) & \frac{\text{C}(r, \text{K}(a, t_1, \phi), \ldots)}{[R_8]} \\
\text{C}(r, \text{K}(a, t_1, \phi) \to \text{K}(a, t_2, \phi)) & \frac{\text{K}(a, t_2, \phi) \to \text{K}(a, t_1, \phi)}{[R_3]} \\
\text{C}(r, \text{B}(a, t_1, \phi) \to \text{B}(a, t_2, \phi)) & \frac{\text{C}(r, \text{B}(a, t_2, \phi))}{[R_2]} \\
\text{C}(r, \text{C}(t, \phi_1) \to \text{C}(t, \phi_2)) & \frac{\text{C}(t, \phi_1) \to \phi_1 \to \phi_2 \to \text{C}(t, \phi_2)}{[R_3]} \\
\text{C}(r, \forall x. \phi \to \phi[x \mapsto t]) & \frac{\text{C}(t, \phi \to \phi[t \mapsto \phi(x)])}{[R_3]} \\
\text{C}(t, \forall x. \phi \to \phi) & \frac{\text{C}(t, \phi \to \phi[t \mapsto \phi(x)])}{[R_3]} \\
\text{S}(a, t, \phi) & \frac{\text{I}(a, t, \text{happens}(a', \alpha, t'))}{[R_{12}]} \\
\text{B}(a, t, \phi) & \frac{\text{B}(a, t, \text{O}(a, t, \phi, \chi))}{[R_{14}]} \\
\text{K}(a, t, \text{I}(a, t, \phi)) & \frac{\text{O}(a, t, \phi, \chi)}{[R_{13}]} \\
\end{align*}
\]
References


[Stannett and Németh, 2014] Mike Stannett and István Németh. Us-