22

Modeling Morality with Prospective Logic

Luís Moniz Pereira and Ari Saptawijaya

Introduction

Morality no longer belongs only to the realm of philosophers. Recently, there has been a growing interest in understanding morality from the scientific point of view. This interest comes from various fields, for example, primatology (de Waal 2006), cognitive sciences (Hauser 2007; Mikhail 2007), neuroscience (Tancredi 2005), and other various interdisciplinary perspectives (Joyce 2006; Katz 2002). The study of morality also attracts the artificial intelligence community from the computational perspective and has been known by several names, including machine ethics, machine morality, artificial morality, and computational morality. Research on modeling moral reasoning computationally has been conducted and reported on, for example, at the AAAI 2005 Fall Symposium on Machine Ethics (Guarini 2005; Rzepka and Araki 2005).

There are at least two reasons to mention the importance of studying morality from the computational point of view. First, with the current growing interest to understand morality as a science, modeling moral reasoning computationally will assist in better understanding morality. Cognitive scientists, for instance, can greatly benefit in understanding complex interaction of cognitive aspects that build human morality; they may even be able to extract moral principles people normally apply when facing moral dilemmas. Modeling moral reasoning computationally can also be useful for intelligent tutoring systems, for instance, to aid in teaching morality to children. Second, as artificial agents are more and more expected to be fully autonomous and work on our behalf, equipping agents with the capability to compute moral decisions is an indispensable requirement. This is particularly true when the agents are operating in domains where moral dilemmas occur, for example, in health care or medical fields.

Our ultimate goal within this topic is to provide a general framework to model morality computationally. This framework should serve as a toolkit to codify arbitrarily chosen moral rules as declaratively as possible. We envisage that logic programming is an appropriate paradigm to achieve our purpose. Continuous and

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

active research in logic programming has provided us with necessary ingredients that look promising enough to model morality. For instance, default negation is suitable for expressing exception in moral rules; abductive logic programming (Kakas et al. 1998; Kowalski 2006) and stable model semantics (Gelfond and Lifschitz 1998) can be used to generate possible decisions along with their moral consequences; and preferences are appropriate for preferring among moral decisions or moral rules (Dell' Acqua and Pereira 2005, 2007).

In this paper, we present our preliminary attempt to exploit these enticing features of logic programming to model moral reasoning. In particular, we employ prospective logic programming (Lopes and Pereira 2006; Pereira and Lopes 2007), an ongoing research project that incorporates these features. For the moral domain, we take the classic trolley problem of Foot (1967). This problem is challenging to model, because it contains a family of complex moral dilemmas. To make moral judgments on these dilemmas, we model the principle of double effect as the basis of moral reasoning. This principle is chosen by considering empirical research results in cognitive science (Hauser 2007) and law (Mikhail 2007) that show the consistency of this principle to justify similarities of judgments by demographically diverse populations when given this set of dilemmas. Additionally, we also employ prospective logic programming to model another moral principle, the principle of triple effect (Kamm 2006). The model allows us to explain computationally the difference of moral judgments drawn using these two similar but distinct moral principles.

Our attempt to model moral reasoning on this domain shows encouraging results. Using features of prospective logic programming, we can conveniently model the moral domain (that is, various moral dilemmas of the trolley problem), the principle of double effect, and the principle of triple effect, all of those in a declarative manner. Our experiments on running the model also successfully deliver moral judgments that conform to the human empirical research results.

We organize the paper as follows. First, we discuss briefly and informally prospective logic programming. Then, we explain the trolley problem and the double and triple effect principles, respectively. We detail how we model them in prospective logic programming together with the results of our experiments regarding that model in the subsection that follows. Finally, we conclude and discuss possible future work.

Prospective Logic Programming

Prospective logic programming enables an evolving program to look ahead prospectively into its possible future states and to prefer among them to satisfy goals (Lopes and Pereira 2006; Pereira and Lopes 2007). This paradigm is particularly beneficial to the agents community, because it can be used to predict an agent's future by employing the methodologies from abductive logic

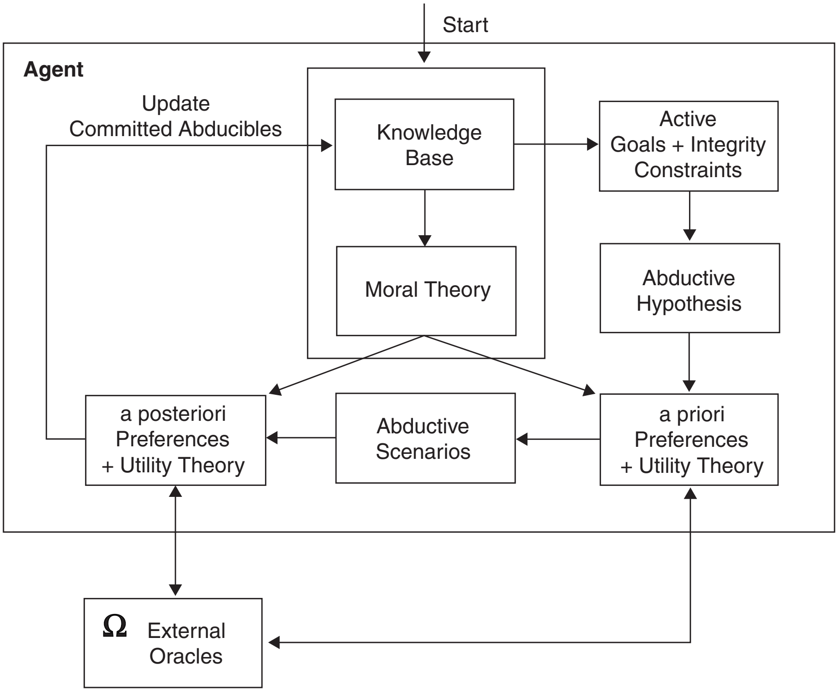

Figure 22.1. Prospective logic agent architecture.

programming (Kakas et al. 1998; Kowalski 2006) in order to synthesize and maintain abductive hypotheses.

Figure 22.1 shows the architecture of agents that are based on prospective logic (Pereira and Lopes 2007). Each prospective logic agent is equipped with a knowledge base and a moral theory as its initial theory. The problem of prospection is then of finding abductive extensions to this initial theory that are both relevant (under the agent's current goals) and preferred (with respect to preference rules in its initial theory). The first step is to select the goals that the agent will possibly attend to during the prospective cycle. Integrity constraints are also considered here to ensure the agent always performs transitions into valid evolution states. Once the set of active goals for the current state is known, the next step is to find out which are the relevant abductive hypotheses. This step may include the application of a priori preferences, in the form of contextual preference rules, among available hypotheses to generate possible abductive scenarios. Forward reasoning can then be applied to abducibles in those scenarios to obtain relevant consequences, which can then be used to enact a posteriori preferences. These preferences can be enforced by employing utility theory and, in a moral situation, also moral theory. In case additional information is needed to enact preferences, the agent may consult external oracles. This greatly benefits agents in giving them the ability to probe the outside environment, thus providing better informed choices, including the making of experiments. The mechanism to consult oracles is realized by posing questions to external systems, be they other agents, actuators, sensors, or other procedures. Each oracle mechanism may have certain conditions specifying whether it is available for questioning. Whenever the agent acquires additional information, it is possible that ensuing side effects affect its original search, for example, some already-considered abducibles may now be disconfirmed and some new abducibles are triggered. To account for all possible side effects, a second round of prospection takes place.

ACORDA is a system that implements prospective logic programming and is based on the aforementioned architecture. ACORDA is implemented based on the implementation of EVOLP (Alferes et al. 2002) and is further developed on top of XSB Prolog.[1] In order to compute abductive stable models (Dell' Acqua and Pereira 2005, 2007), ACORDA also benefits from the XSB-XASP interface to Smodels.[2]

In this section, we discuss briefly and informally prospective logic programming and some constructs from ACORDA that are relevant to our work. For a more detailed discussion on prospective logic programming and ACORDA, interested readers are referred to the original paper (Lopes and Pereira 2006; Pereira and Lopes, 2007).

Language

Let $\mathcal{L}$ be a first-order language. A domain literal in $\mathcal{L}$ is a domain atom A or its default negation *not* A. The latter is to express that the atom is false by default (close world assumption). A domain rule in Λ is a rule of the form:

$$A \leftarrow L_1,\ldots,L_t. \qquad (t \geq 0)$$

where A is a domain atom and $L_1,\ldots,L_t$ are domain literals. An integrity constraint in $\mathcal{L}$ is a rule of the form:

$$\perp \leftarrow L_1,\ldots,L_t. \qquad (t > 0)$$

where $\perp$ is a domain atom denoting falsity, and $L_1,\ldots,L_t$ are domain literals. In ACORDA, $\leftarrow$ and $\perp$ will be represented by `<-` and `falsum`, respectively.

A (logic) program P over $\mathcal{L}$ is a set of domain rules and integrity constraints, standing for all their ground instances.

Abducibles

Every program P is associated with a set of abducible atoms $A \subseteq \mathcal{L}$. Abducibles can be seen as hypotheses that provide hypothetical solutions or possible explanations of given queries.

[1] XSB Prolog is available at `http://xsb.sourceforge.net`

[2] Smodels is available at `http://www.tcs.hut.fi/Software/smodels`

An abducible A can be assumed only if it is a considered one, that is, it is expected in the given situation and, moreover, there is no expectation to the contrary (Dell' Acqua and Pereira 2005, 2007).

$$consider\ (A) \leftarrow expect\ (A),\ not\ expect\_not\ (A).$$

The rules about expectations are domain-specific knowledge contained in the theory of the program and effectively constrain the hypotheses that are available.

In addition to mutually exclusive abducibles, ACORDA also allows sets of abducibles. Hence, an abductive stable model may contain more than a single abducible. To enforce mutually exclusive abducibles, ACORDA provides predicate exclusive/2. The use of this predicate will be illustrated later, when we model morality in a subsequent section.

A Posteriori Preferences

Having computed possible scenarios represented by abductive stable models, more favorable scenarios can be preferred among them a posteriori. Typically, a posteriori preferences are performed by evaluating consequences of abducibles in abductive stable models. The evaluation can be done quantitatively (for instance, by utility functions) or qualitatively (for instance, by enforcing some rules to hold). When currently available knowledge is insufficient to prefer among abductive stable models, additional information can be gathered, for example, by performing experiments or consulting an oracle.

To realize a posteriori preferences, ACORDA provides predicate select/2 that can be defined by users following some domain-specific mechanism for selecting favored abductive stable models. The use of this predicate to perform a posteriori preferences in a moral domain will be discussed in a subsequent section.

The Trolley Problem

Several interesting results have emerged from recent interdisciplinary studies on morality. One common result from these studies shows that morality has evolved over time. In particular, Hauser (2007), in his recent work, argues that a moral instinct, which generates rapid judgments about what is morally right or wrong, has evolved in our species.

Hauser (2007) and Mikhail (2007) propose a framework of human moral cognition known as universal moral grammar that is analogous to Chomsky's universal grammar in language. Universal moral grammar, which can be culturally adjusted, provides universal moral principles that enable an individual to unconsciously evaluate what actions are permissible, obligatory, or forbidden. To support this idea, Hauser and Mikhail independently created a test to assess moral judgments of subjects from demographically diverse populations using the classic trolley problems. Despite their diversity, the result shows that most subjects widely share moral judgments when given a moral dilemma from the trolley-problem suite. Although subjects are unable to explain the moral rules in their attempts at justification, their moral judgments are consistent with a moral rule known as the principle of double effect.

The trolley problem presents several moral dilemmas that inquire whether it is permissible to harm one or more individuals for the purpose of saving others. In all cases, the initial circumstances are the same (Hauser, 2007):

"There is a trolley and its conductor has fainted. The trolley is headed toward five people walking on the track. The banks of the track are so steep that they will not be able to get off the track in time."

Given this initial circumstance, in this work we consider six classical cases of moral dilemmas, employed for research on morality in people. These six cases are visually depicted in Figure 22.2.

1. **Bystander**. Hank is standing next to a switch, which he can throw, that will turn the trolley onto a parallel side track, thereby preventing it from killing the five people. However, there is a man standing on the side track with his back turned. Hank can throw the switch, killing him; or he can refrain from doing this, letting the five die. Is it morally permissible for Hank to throw the switch?
2. **Footbridge**. Ian is on the footbridge over the trolley track. He is next to a heavy object, which he can shove onto the track in the path of the trolley to stop it, thereby preventing it from killing the five people. The heavy object is a man, standing next to Ian with his back turned. Ian can shove the man onto the track, resulting in death; or he can refrain from doing this, letting the five die. Is it morally permissible for Ian to shove the man?
3. **Loop Track**. Ned is standing next to a switch, which he can throw, that will temporarily turn the trolley onto a loop side track. There is a heavy object on the side track. If the trolley hits the object, the object will slow the train down, giving the five people time to escape. The heavy object is a man, standing on the side track with his back turned. Ned can throw the switch, preventing the trolley from killing the five people, but killing the man; or he can refrain from doing this, letting the five die. Is it morally permissible for Ned to throw the switch?
4. **Man-in-front**. Oscar is standing next to a switch, which he can throw, that will temporarily turn the trolley onto a side track. There is a heavy object on the side track. If the trolley hits the object, the object will slow the train down, giving the five people time to escape. There is a man standing on the side track in front of the heavy object with his back turned. Oscar can throw the switch, preventing the trolley from killing the five people, but killing the man. Or he can refrain from doing this, letting the five die. Is it morally permissible for Oscar to throw the switch?
5. **Drop Man**. Victor is standing next to a switch, which he can throw, that will drop a heavy object into the path of the trolley, thereby stopping the trolley and preventing it from killing the five people. The heavy object is a man,

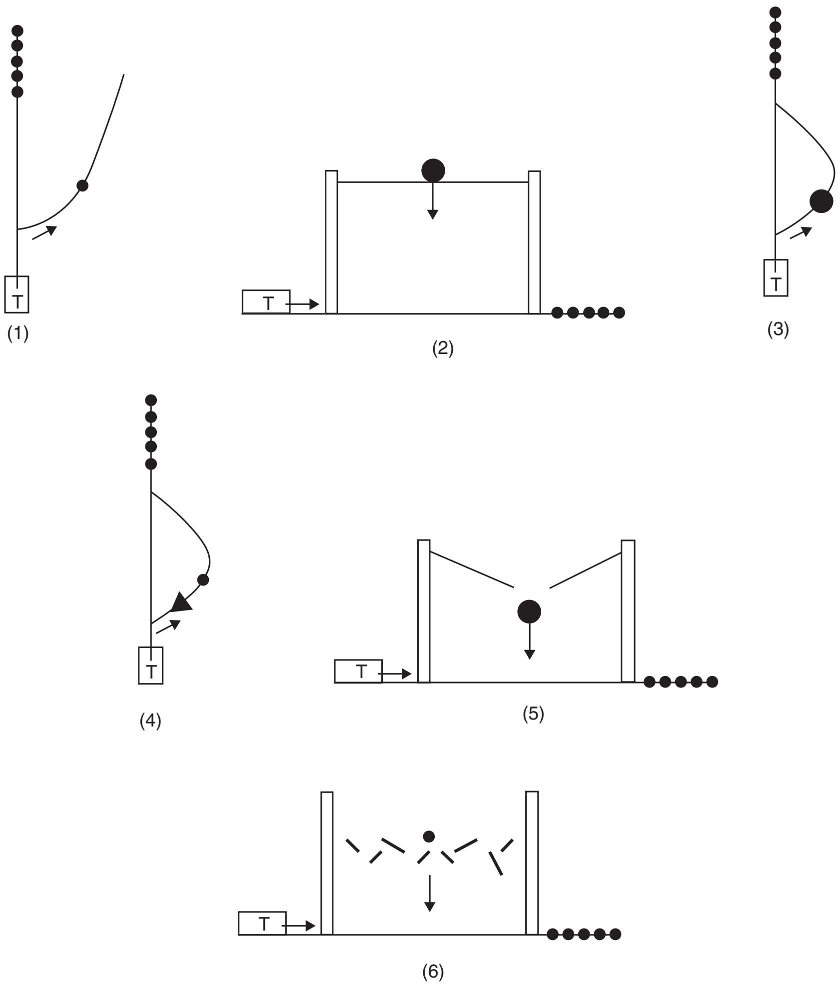

Figure 22.2. The six trolley cases: (1) Bystander, (2) Footbridge, (3) Loop Track, (4) Man-in-front, (5) Drop Man, (6) Collapse Bridge.

who is standing on a footbridge overlooking the track. Victor can throw the switch, killing him; or he can refrain from doing this, letting the five die. Is it morally permissible for Victor to throw the switch?

6. **Collapse Bridge**. Walter is standing next to a switch, which he can throw, that will collapse a footbridge overlooking the tracks into the path of the trolley, thereby stopping the train and preventing it from killing the five people. There is a man standing on the footbridge. Walter can throw the switch, killing him; or he can refrain from doing this, letting the five die. Is it morally permissible for Walter to throw the switch?

Table 22.1. *Summary of moral judgments for the trolley problem*

Case	Judgment
1. Bystander	Permissible
2. Footbridge	Impermissible
3. Loop Track	Impermissible
4. Man-in-front	Permissible
5. Drop Man	Impermissible
6. Collapse Bridge	Permissible

Interestingly, although all cases have the same goal (i.e., to save five, but which requires killing one), subjects come to different judgments on whether the action to reach the goal is permissible or impermissible. As reported by Mikhail (2007), the judgments appear to be widely shared among demographically diverse populations, the summary being given in Table 22.1.

The Double and Triple Effect

Although subjects have difficulties uncovering which moral rules they apply for reasoning in these cases, their judgments appear to be consistent with the so-called principle of double effect. The principle can be expressed as follows:

> Harming another individual is permissible if it is the foreseen consequence of an act that will lead to a greater good; in contrast, it is impermissible to harm someone else as an intended means to a greater good. (Hauser 2007)

The key expression here is "intended means." We shall refer in the subsequent sections to the action of harming someone as an intended means, as an intentional killing.

In contrast to the result shown in Table 22.1, Otsuka (2008) in his work states that diverting the trolley in the Loop Track case strikes most moral philosophers as permissible. In his work, he refers to Kamm's principle of triple effect (Kamm 2006) to explain permissibility of diverting the trolley onto a loop side track. The triple effect principle is a revised version of the double effect principle. This moral principle refines the double effect principle, in particular on harming someone as an intended means. In this case, the triple effect principle distinguishes an action that is performed *in order* to bring about an evil from an action performed *that directly causes* an evil to occur without production of evil being its goal. The latter is a new category of action that neither treats the occurrence of evil as a foreseen, unintended consequence nor as an action performed in order to intentionally bring about an evil. Similarly to the double effect principle, the triple effect principle classifies an action performed in order to intentionally bring about an evil as an impermissible action. Yet, this moral principle is more tolerant to the third effect, that is, it is *permissible* for an action to be performed because an evil will occur, if not intended as such.

In the footbridge case, both the double and the triple effect principles arrive at the same moral judgment, namely that it is impermissible to shove a man in order to cause the trolley to hit the man. On the other hand, different moral judgments are delivered for the Loop Track case when employing the double and the triple effect principles. Whereas it is impermissible by the double effect to divert the trolley, this action is *permissible* by the triple effect principle. According to the triple effect principle, the action of diverting the trolley onto the loop side track is performed because it will hit the man, not in order to hit the man (Kamm 2006). As in the other cases, the goal of the Loop Track case is to prevent the trolley from killing the five people. This goal can be achieved by throwing the switch to divert the trolley onto the side track. However, diverting the trolley raises another problem because the side track is a loop track that will take the trolley back to the main track and will still hit the five people. The action of diverting the trolley is not necessarily intended in order to hit the man on the side track. Instead, this action is performed because it will hit the man, eliminating the new problem created by diverting the trolley (Kamm 2006).

To better distinguish an action that is performed in order to bring about an evil and that is performed because an evil will occur, the Loop Track case can be compared with the following case. Consider a variant of the Loop Track case in which the loop side track is empty. Note that, instead of only diverting the trolley onto the empty loop side track, an ancillary act of shoving a man onto the loop side track can be performed. How can this case be morally distinguished from the original Loop Track case? In the original Loop Track case, the man has already been standing on the loop side track when the action of diverting the trolley is carried out, causing the death of the man. In the other case, the death of the man is not merely caused by diverting the trolley (the loop side track is initially empty). Instead, the man dies as a consequence of a further action, that is, shoving the man. This action is intentionally performed to place the man on the side track in order for the trolley to hit him, hence preventing the trolley from killing the five people.

We shall show in the subsequent sections, how we can computationally distinguish these two similar cases in the context of the triple effect principle. In particular, we discuss a feature available in ACORDA to explain the previous moral reasoning computationally.

Modeling Morality in ACORDA

It is interesting to model the trolley problem in ACORDA due to the intricacy that arises from the dilemma itself. Moreover, there are similarities and also differences between cases and between the moral principles employed. Consequently, this adds complexity to the process of modeling them in order to deliver appropriate moral decisions through reasoning. By appropriate moral decisions, we mean the ones that conform with those the majority of people make in adhering to the principle of double effect (though we also consider the principle of triple effect, as a comparison to the principle of double effect).

We model each case of the trolley problem in ACORDA separately. The principle of double effect and triple effect are modeled via a priori constraints and a posteriori preferences. To assess how flexible our model is of the moral rule, we additionally model another variant for the cases of Footbridge and Loop Track. Even for these variants, our model of the moral rules allows the reasoning to deliver moral decisions as expected.

In each case of the trolley problem, there are always two possible decisions to make. One of these is the same for all cases, that is, letting the five people die by merely watching the train go straight. The other decision depends on the cases, for example, throwing the switch, shoving a heavy man, or the combination of them, with the same purpose: to save the five people, but also harming a person in the process.

In this work, these possible decisions are modeled in ACORDA as abducibles. Moral decisions are made by computing abductive stable models and then preferring among them those models with the abducibles and consequences that conform to the principle of double effect or triple effect.

In subsequent sections we detail the model for all six cases of the trolley problem in ACORDA. We also show how to model the principle of double effect and triple effect. Then we present the results of running our models in the ACORDA system.

Modeling the Bystander Case

Facts to describe that there is a side track and that a man (here, named John) is standing on that side track can be modeled simply as the following:

```
side_track.
on_side(john).
human(john).
```

The clauses "`expect(watching)`" and "`expect(throwing_switch)`" in the following model indicate that watching and throwing the switch, respectively, are two available abducibles that represent possible decisions Hank has. Because the purpose of the switch is to turn the trolley onto the side track, the action "`throwing_switch`" is only considered as an abducible if the side track exists. The other clauses represent the chain of actions and consequences for every abducible.

The predicate "`end(die(5))`" represents the final consequence if `watching` is abduced, that is, it will result in five people dying. On the other hand, the predicate "`end(save_men, ni_kill(N))`" represents the final consequence whenever "`throwing_switch`" is abduced, that is, it will save the five people without intentionally killing someone. The way of representing these two consequences is chosen differently because of the different nature of these

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

two abducibles. Merely watching the trolley go straight is an omission of action that just has negative consequence, whereas throwing the switch is an action that is performed to achieve a goal and additionally has negative consequence. Because abducibles in other cases of the trolley problem also share this property, this way of representation will be used throughout them. The predicate "`observed_end`" is used to encapsulate these two different means of representation, and will be useful later to avoid floundering when we model the principle of double effect.

```
expect(watching).
train_straight <- consider(watching).
end(die(5)) <- train_straight.
observed_end <- end(X).

expect(throwing_switch) <- side_track.
turn_side <- consider(throwing_switch).
kill(1) <- human(X), on_side(X), turn_side.
end(save_men,ni_kill(N)) <- turn_side, kill(N).
observed_end <- end(X,Y).
```

We can model the exclusiveness of the two possible decisions, that is, Hank has to decide either to throw the switch or merely watch, by using the exclusive/2 predicate of ACORDA:

```
exclusive(throwing_switch,decide).
exclusive(watching,decide).
```

Note that the exclusiveness between two possible decisions also holds in other cases.

Modeling the Footbridge Case

We represent the fact of a heavy man (here, also named John) on the footbridge standing near to Ian similarly to the Bystander case:

```
stand_near(john).
human(john).
heavy(john).
```

We can make this case more interesting by additionally having another (inanimate) heavy object – a rock – on the footbridge near to Ian and see whether our model of the moral rule still allows the reasoning to deliver moral decisions as expected:

```
stand_near(rock).
inanimate_object(rock).
heavy(rock).
```

Alternatively, if we want only to have either a man or an inanimate object on the footbridge next to Ian, we can model it by using an even loop over default negation:

```
stand_near(john) <- not stand_near(rock).
stand_near(rock) <- not stand_near(john).
```

In the following we show how to model the action of shoving an object as an abducible, together with the chain of actions and consequences for this abducible. The model for the decision of merely watching is the same as in the case of Bystander. Indeed, because the decision of watching is always available for other cases, we use the same modeling in every case.

```
expect(shove(X)) <- stand_near(X).
on_track(X) <- consider(shove(X)).
stop_train(X) <- on_track(X), heavy(X).
kill(1) <- human(X), on_track(X).
kill(0) <- inanimate_object(X), on_track(X).
end(save_men,ni_kill(N)) <- inanimate_object(X),
                            stop_train(X),
                            kill(N).
end(save_men,i_kill(N)) <- human(X),
                           stop_train(X),
                           kill(N).
observed_end <- end(X,Y).
```

Note that the action of shoving an object is only possible if there is an object near Ian to shove, hence the clause "`expect(shove(X)) <- stand_near(X).`" We also have two clauses that describe two possible final consequences. The clause with the head "`end(save_men, ni_kill(N))`" deals with the consequence of reaching the goal, that is, saving five, but not intentionally killing someone (in particular, without killing anyone in this case). To the contrary, the clause with the head "`end(save_men, i_kill(N))`" expresses the consequence of reaching the goal but involving an intentional killing.

Modeling the Loop Track Case

We consider three variants for the Loop Track case. Two variants are similar to the case of footbridge. The other variant will be used later to discuss the difference between the double effect and the triple effect principles, as already discussed.

In the first variant, instead of having only one loop side track as in the original scenario, we consider two loop side tracks: the left and the right loop side tracks. John, a heavy man, is standing on the left side track, whereas on the right side

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

track there is an inanimate heavy object, a rock. These facts can be represented in ACORDA as follows:

```
side_track(left).
side_track(right).

on(john,left).
human(john).
heavy(john).

on(rock,right).
inanimate_object(rock).
heavy(rock).
```

The switch can be thrown to either one of the two loop side tracks. This action of throwing the switch can be modeled as an abducible. This abducible together with the chain of actions and consequences that follow can be modeled declaratively in ACORDA:

```
expect(throwing_switch(Z)) <- side_track(Z).
turn_side(Z) <- consider(throwing_switch(Z)).
slowdown_train(X) <- turn_side(Z), on(X,Z),
                       heavy(X).
kill(1) <- turn_side(Z), on(X,Z), human(X).
kill(0) <- turn_side(Z), on(X,Z), inanimate_object(X).
end(save_men,ni_kill(N)) <- inanimate_object(X),
                              slowdown_train(X),
                              kill(N).
end(save_men,i_kill(N)) <- human(X),
                             slowdown_train(X),
                             kill(N).
observed_end <- end(X,Y).
```

Note that the clause:

```
expect(throwing_switch(Z)) <- side_track(Z)
```

states that the action of throwing the switch to turn the trolley onto a side track Z does make sense if the side track Z, to which the switch is connected, is available.

For the second variant, we consider one loop side track with either a man or an inanimate object on the side track. This alternative can be modeled by using an even loop over default negation:

```
side_track(john) <- not side_track(rock).
side_track(rock) <- not side_track(john).
```

Note that the argument of the predicate "side_track/1" does not refer to some side track (left or right, as in the first variant), but refers to some object (rock or John) on the only side track. This leads to a slight difference in the model of the chain of consequences for the throwing switch action:

```
expect(flipping_switch(X)) <- side_track(X).
turn_side(X) <- consider(flipping_switch(X)).
slowdown_train(X) <- turn_side(X), heavy(X).
kill(1) <- turn_side(X), human(X).
kill(0) <- turn_side(X), inanimate_object(X).
end(save(5),ni_kill(N)) <- inanimate_object(X),
                           slowdown_train(X),
                           kill(N).
end(save(5),i_kill(N)) <- human(X),
                          slowdown_train(X),
                          kill(N).
observed_end <- end(X,Y).
```

We may observe that from the first two variants there is no need to change the whole model; only the part of the model that represents the different facts need to be adapted. Moreover, the second variant shows that we do not need to have separate programs to model the fact that a man or an inanimate object is on the side track. Instead, we can have only one program and take the benefit from using an even loop over default negation to capture alternatives between objects.

Throwing the Switch and Shoving a Man

This variant will be useful later, when we discuss the difference between the double effect and the triple effect principles.

In this scenario, the action of throwing the switch to divert the trolley can be followed by the action of shoving a man to place the man onto the empty loop side track. There is only one loop side track, which we can represent by using a simple fact "`side_track.`" To represent either that there has already been a man standing on the loop side track or the loop side track is initially empty, we can use the following even loop over default negation:

```
man_stand_sidetrack <- not empty_sidetrack.
empty_sidetrack <- not man_stand_sidetrack.
```

We have three actions available, represented as abducibles, with the chain of consequences modeled as follows. For simplicity, without loss of generality, we assume that the man standing on the side track or the man shoved is heavy.

```
expect(watching).
train_straight <- consider(watching).
end(die(5)) <- train_straight.
observed_end <- end(X).

expect(throwing_switch) <- side_track.
turn_side <- consider(throwing_switch).
end(save_men,standing_hitting) <- man_stand_sidetrack,
                                  turn_side.

expect(shoving) <- empty_sidetrack,
                   consider(throwing_switch).

freely_goto_maintrack <- empty_sidetrack,
                         turn_side,
                         not consider(shoving).
end(die(5)) <- freely_goto_maintrack.

place_man_sidetrack <- consider(shoving).
end(save_men,placing_hitting) <- place_man_sidetrack,
                                 turn_side.

observed_end <- end(X,Y).
```

This model can be read declaratively. The action `watching`, together with its consequence, is similar as in other cases.

The action "`throwing_switch`" is possible if there is a side track to which the switch is connected. Throwing the switch will turn the trolley to the side track. If a (heavy) man has already been standing on the side track and the trolley also turns to the side track, then the five people are saved but the man standing on the side track is killed.

Following our scenario, the action of `shoving` is available if there is nothing on the side track ("`empty_sidetrack`") and the action of throwing the switch is performed. This is modeled by the clause:

```
expect(shoving) <- empty_sidetrack,
                   consider(throwing_switch).
```

This means that the action of shoving is a further action following the action of throwing the switch.

If the side track is empty, the trolley has turned to the side track; but if the action of shoving is not abduced, then the trolley will freely go to the main track where the five people are walking. This results in the death of the five people.

On the other hand, if shoving is abduced (as an ancillary action of throwing the switch), then this will result in placing a (heavy) man on the side track. If the trolley has turned to the side track and the man has been placed on the side track, then the five people are saved with the cost of killing the man placed on the side track as the consequence of shoving.

Modeling the Man-in-Front Case

Recall that in the case of Man-in-Front, there is a side track onto which the trolley can turn. On this side track, there is a heavy (inanimate) object, a rock. There is also a man standing in front of the heavy object. We can model these facts in ACORDA as follows:

```
side_track.
on_side(rock).
inanimate_object(rock).
heavy(rock).
in_front_of(rock,jack).
human(jack).
```

In order to prevent the trolley from killing the five people, a switch can be thrown to turn the trolley onto the side track. The following clauses are used to model the "throwing_switch" action as an abducible together with the consequences that follow:

```
expect(throwing_switch) <- side_track.
turn_side <- consider(throwing_switch).
kill(1) <- turn_side, on_side(X),
           in_front_of(X,Y), human(Y).
slowdown_train(X) <- turn_side, on_side(X),
                     heavy(X).

end(save_men,ni_kill(N)) <- inanimate_object(X),
                            slowdown_train(X),
                            kill(N).
end(save_men,i_kill(N)) <- human(X),
                           slowdown_train(X),
                           kill(N).

observed_end <- end(X,Y).
```

Modeling the Drop Man Case

The following facts model the scenario of the Drop Man case:

```
switch_connected(bridge).
human(john).
heavy(john).
on(bridge,john).
```

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

These facts state that there is a switch connected to the bridge where there is a heavy man, John, standing on the bridge. As in other cases, it is straightforward to model in ACORDA the "throwing_switch" actions along with its consequences:

```
expect(throwing_switch(Z)) <- switch_connected(Z).
drop(X) <- consider(throwing_switch(Z)), on(Z,X).
kill(1) <- human(X), drop(X).
stop_train(X) <- heavy(X), drop(X).
end(save_men,ni_kill(N)) <- inanimate_object(X),
                            stop_train(X),
                            kill(N).
end(save_men,i_kill(N)) <- human(X),
                           stop_train(X),
                           kill(N).
observed_end <- end(X,Y).
```

In this model, the fact that the switch is connected to the bridge is the condition for the action of throwing the switch to be reasonable to perform.

Modeling the Collapse Bridge Case

The Collapse Bridge case is a variation from the Drop Man case. In this case, the footbridge itself is the heavy object that may stop the train when it collapses and prevent the train from killing the five people. There is also a man, John, standing on the footbridge, as in the Drop Man case. These facts can be modeled in ACORDA as follows:

```
switch_connected(bridge).
heavy(bridge).
inanimate_object(bridge).
human(john).
on(bridge,john).
```

With slight variation of consequences from the Drop Man case, the following clauses model the "throwing_switch" action as an abducible together with the consequences that follow:

```
expect(throwing_switch(X)) <-
  switch_connected(X).

collapse(X) <- consider(throwing_switch(X)).
stop_train(X) <- heavy(X), collapse(X).
kill(1) <- human(Y), on(X,Y), collapse(X).
end(save_men,ni_kill(N)) <- inanimate_object(X),
                            stop_train(X),
                            kill(N).
```

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

```
end(save_men,i_kill(N)) <- human(X),
                           stop_train(X),
                           kill(N).
observed_end <- end(X,Y).
```

Modeling the Principles of Double and Triple Effect

The principles of double effect and triple effect can be modeled by using a combination of integrity constraints and a posteriori preferences.

Integrity constraints are used for two purposes. First, we need to observe the final consequences or endings of each possible decision to enable us later to morally prefer decisions by considering the greater good between possible decisions. This can be achieved by specifying the following integrity constraint:

```
falsum <- not observed_end.
```

This integrity constraint enforces all available decisions to be abduced together with their consequences by computing all possible observable hypothetical endings using all possible abductions. Indeed, to be able to reach a moral decision, all hypothetical scenarios afforded by the abducibles must lead to an observable ending. Second, we also need to rule out impermissible actions, that is, actions that involve intentional killing in the process of reaching the goal. This can be enforced by specifying the integrity constraint

```
falsum <- intentional_killing.
```

Depending on whether we employ the double effect or the triple effect principle, intentional killing can be easily defined. For the double effect principle, it can be modeled as follows:

```
intentional_killing <- end(save_men,i_kill(Y)).
```

Similarly, intentional killing for the triple effect principle can be modeled as follows:

```
intentional_killing <-
  end(save_men,placing_hitting).
```

These integrity constraints serve as the first filtering function of our abductive stable models by ruling out impermissible actions (the latter being coded by abducibles). In other words, integrity constraints already afford us with just those abductive stable models that contain only permissible actions.

Additionally, one can prefer among permissible actions those resulting in greater good. This can be realized by a posteriori preferences that evaluate the consequences of permissible actions and then prefer the one with greater good. The following definition of "`select/2`" achieves this purpose. The first argument of this predicate refers to the set of initial abductive stable models to prefer, whereas the second argument refers to the preferred ones. The auxiliary predicate "`select/3`" only keeps abductive stable models that contain decisions with greater good of consequences. In the trolley problem, the greater good is evaluated by a utility function concerning the number of people that die as a result of possible decisions. This is realized in the definition of predicate "`select/3`" by comparing final consequences that appear in the initial abductive stable models. The first clause of "`select/3`" is the base case. The second clause and the third clause together eliminate abductive stable models containing decisions with worse consequences, whereas the fourth clause will keep those models that contain decisions with greater good of consequences.

```
select(Xs,Ys) :- select(Xs,Xs,Ys).

select([],_,[]).
select([X|Xs],Zs,Ys) :-
      member(end(die(N)),X),
      member(Z,Zs),
      member(end(save_men,ni_kill(K)),Z), N > K,
      select(Xs,Zs,Ys).
select([X|Xs],Zs,Ys) :-
      member(end(save_men,ni_kill(K)),X),
      member(Z,Zs),
      member(end(die(N)),Z), N =< K,
      select(Xs,Zs,Ys).
select([X|Xs],Zs,[X|Ys]) :- select(Xs,Zs,Ys).
```

Recall the variant of the Footbridge case, in which either a man or an inanimate object is on the footbridge next to Ian. This exclusive alternative is specified by an even loop over default negation and we have an abductive stable model that contains the consequence of letting the five people die when a rock is next to Ian. This model is certainly *not* the one we would like our moral reasoner to prefer. The following replacement definition of "`select/2`" accomplishes this case.

```
select([],[]).
select([X|Xs],Ys) :-
      member(end(die(N)),X),
      member(stand_near(rock),X),
      select(Xs,Ys).
select([X|Xs],[X|Ys]) :- select(Xs,Ys).
```

It is important to note that in this case, because either a man or a rock is near to Ian and the model with shoving a man is already ruled out by our integrity constraint, there is no need to consider greater good in terms of the number of people that die. This means, as shown subsequently, that only two abductive stable models are preferred: the model with watching as the abducible whenever a man is standing near to Ian; and the other being the model with shoving the rock as the abducible.

There is an interest to be able to specify a posteriori preferences more declaratively, that is, by encapsulating the details of predicate "`select/2`" from the viewpoint of users. Although this may depend on the domain where moral reasoning is applied, some kind of generic macros can be defined. Our subsequent work has evidenced preliminary results (Pereira and Saptawijaya 2007). In those results, we extend the syntax of ACORDA by introducing the predicates "`elim/1`" and "`exists/1.`" We provide two types of generic macros for the purpose of specifying a posteriori preferences.

The first type of macros allows us to realize a posteriori preferences by eliminating abductive stable models containing abducibles with worse consequences. This can be done by comparing some consequences among initial abductive stable models. The aforementioned definition of predicate "`select/3`" is an example of this type. Instead of using this definition of "`select/3`," the same a posteriori preferences can now be expressed more declaratively as follows:

```
elim([end(die(N))]) <-
   exists([end(save_men,ni_kill(K))]), N > K.

elim([end(save_men,ni_kill(K))]) <-
   exists([end(die(N))]), N =< K.
```

The first clause is used to eliminate abductive stable models containing the literal "`end(die(N))`" if there exists other abductive stable models containing the literal "`end(save_men,ni_kill(K))`" and "`N>K`." The second clause can be read similarly.

The second type of macros deals with a posteriori preferences where an abductive stable model is eliminated based only on some literals it contains. This is in contrast with the first type of macros, where the elimination of an abductive stable model is performed through comparison with other abductive stable models. The latter definition of predicate "`select/2`" (in dealing with the variant of the Footbridge case) falls into this type. This definition of "`select/2`" can be replaced with the following line:

```
elim([end(die(N)),stand_near(rock)]).
```

This simple clause eliminates abductive stable models that contain both literals "`end(die(N))`" and "`stand_near(rock).`" Recall that the abductive stable model containing these two literals together is not the one a moral reasoner should prefer, because it may shove the rock to avoid the consequence of letting the five people die.

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

Table 22.2. *Summary of experiments in ACORDA*

Case	Initial models	Final models
Bystander	[throwing_switch],[watching]	[throwing_switch]
Footbridge(a)	[watching],[shove(rock)]	[shove(rock)]
Footbridge(b)	[watching,stand_near(john)], [watching,stand_near(rock)], [shove(rock)]	[watching,stand_near(john)], [shove(rock)]
Loop Track(a)	[throwing_switch(right)], [watching]	[throwing_switch(right)]
Loop Track(b)	[watching,side_track(john)], [watching,side_track(rock)], [throwing_switch(rock)]	[watching,side_track(john)], [throwing_switch(rock)]
Loop Track(c)	[watching,empty_sidetrack], [watching,man_stand_sidetrack], [throwing_switch, empty_sidetrack], [throwing_switch,man_stand_sidetrack]	[watching, empty_sidetrack], [throwing_switch, man_stand_side-track]
Man-in-front	[watching], [throwing_switch(rock)]	[throwing_switch(rock)]
Drop Man	[watching]	[watching]
Collapse Bridge	[watching], [throwing_switch(bridge)]	[throwing_switch(bridge)]

Running the Models in ACORDA

We report now on the experiments of running our models in ACORDA. Table 22.2 gives a summary of all cases of the trolley problem. Column Initial Models contains info about the abductive stable models obtained before a posteriori preferences are applied, whereas column Final Models contains those after a posteriori preferences are applied. Here, only relevant literals are shown. These results comply with the results found for most people in morality laboratory experiments.

Note that entry Footbridge(a) refers to the variant of Footbridge where both a man and a rock are near to Ian, and Footbridge(b) where either a man or a rock is near to Ian. Loop Track(a) refers to the variant of Loop Track where there are two loop tracks, with a man on the left loop track and a rock on the right loop track. Loop Track(b) only considers one loop track where either a man or a rock

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

is on the single loop track. Loop Track(c) is another variant considering the triple effect principle, where the ancillary act of shoving a man is available, following the action of throwing the switch. Note that both cases Loop Track(a) and Loop Track(b) employ the double effect principle. Consequently, there is no initial model (of these two cases) that contains the abducible throwing the switch whenever a man is on the side track. This model has been ruled out by the integrity constraint, because this is considered impermissible in the double effect principle. To the contrary, Loop Track(c), which employs the triple effect principle, does have an initial model with a man on the side track and throwing the switch as an abducible. This model is not ruled out by the integrity constraint, because it is deemed permissible by the triple effect principle.

Selective literals in an abductive stable model (e.g. the literals shown in Table 22.2) can be inspected in ACORDA. The feature of inspection enables us to examine whether some literals are true in the context of adopted abducibles without further abduction. It is useful, in particular, to explain computationally the moral reasoning in scenario Loop Track(c), where the triple effect principle is employed. We simply inspect that the man, an obstacle to the trolley, has already been standing on the side track, because only the action of throwing the switch is abduced. No additional action needs to be abduced to prevent the trolley from hitting the five people. On the other hand, in the models containing "`empty_sidetrack`," mere inspection is not enough because an extra abducible is required to shove the man onto the track in order to deliberately make him an obstacle where there was none. Hence, it prevents the trolley from hitting the five people. However, the abductive stable model containing this further action is already ruled out by our integrity constraint. This conforms with the triple effect principle, as it is impermissible to intentionally shove the man in addition to the act of throwing the switch, in order for the trolley to hit the man for the purpose of stopping the trolley. Indeed, simple inspection does not allow further abduction in order to make it actively true.

Conclusions and Future Work

We have shown how to model moral reasoning using prospective logic programming. We use various dilemmas of the trolley problem and the principle of double effect and triple effect as the moral rules. Possible decisions in a dilemma are modeled as abducibles. Abductive stable models are then computed, which capture abduced decisions and their consequences. Models violating integrity constraints, that is, models that contain actions involving intentional killing, are ruled out. Finally, a posteriori preferences are used to prefer models that characterize more preferred moral decisions, including the use of utility functions. These experiments show that preferred moral decisions, namely the ones that follow the principle of double effect, are successfully delivered. They conform to the results of empirical experiments conducted in cognitive science and law. Regarding the triple effect principle, the inspection feature of ACORDA can be employed to detect

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

mere consequences of abducibles. Hence, we can distinguish computationally two moral judgments in line with the triple effect principle, that is, whether an action is performed in order to bring about an evil or just because an evil will occur.

Much research has emphasized using machine-learning techniques such as statistical analysis (Rzepka and Araki 2005), neural networks (Guarini 2005), case-based reasoning (McLearen 2006), and inductive logic programming (Anderson et al. 2006) to model moral reasoning from examples of particular moral dilemmas. Our approach differs from them as we do not employ machine-learning techniques to deliver moral decisions.

Powers (2006) proposes to use nonmonotonic logic to specifically model Kant's categorical imperatives, but it is unclear whether his approach has ever been realized in a working implementation. On the other hand, Bringsjord et al. (2006) propose the use of deontic logic to formalize moral codes. The objective of their research is to arrive at a methodology that allows an agent to behave ethically as much as possible in an environment that demands such behavior. We share our objective with them to some extent as we also would like to come up with a general framework to model morality computationally. In contrast to our work, they use an axiomatized deontic logic to decide which moral code is operative to arrive at an expected moral outcome. This is achieved by seeking a proof for the expected moral outcome to follow from candidates of operative moral codes.

To arrive at our ultimate research goal, we envision several possible future directions. We would like to explore how to express metarule and metamoral injunctions. By "metarule," we mean a rule to resolve two existing conflicting moral rules in deriving moral decisions. "Metamorality," on the other hand, is used to provide protocols for moral rules, to regulate how moral rules interact with one another. Another possible direction is to have a framework for generating precompiled moral rules. This will benefit fast and frugal moral decision making, which is sometimes needed instead of full deliberative moral reasoning every time (cf. heuristics for decision making in law [Gigerenzer and Engel 2006]).

We envision a final system that can be employed to test moral theories and can also be used for moral-reasoning training, including the automated generation of example tests and their explanation. Finally, we hope our research will help in imparting moral behavior to autonomous agents.

References

Alferes, J. J., Brogi, A., Leite, J. A., and Pereira, L. M. 2002. Evolving Logic Programs. Pages 50–61 of: Flesca, S., Greco, S., Leone, N., and Ianni, G. (eds), *Procs. 8th European Conf. on Logics in Artificial Intelligence (JELIA'02)*. LNCS 2424. Springer.

Anderson, M., Anderson, S., and Armen, C. 2006. MedEthEx: A Prototype Medical Ethics Advisor. In: *Procs. 18th Conf. on Innovative Applications of Artificial Intelligence (IAAI-06)*.

Bringsjord, S., Arkoudas, K., and Bello, P. 2006. Toward a General Logicist Methodology for Engineering Ethically Correct Robots. *IEEE Intelligent Systems*, 21(4), 38–44.

de Waal, F. 2006. *Primates and Philosophers, How Morality Evolved*. Princeton U.P.

https://doi.org/10.1017/CBO9780511978036.027 Published online by Cambridge University Press

Dell'Acqua, P., and Pereira, L. M. 2005. Preferential Theory Revision. Pages 69–84 of: Pereira, L. M., and Wheeler, G. (eds), *Procs. Computational Models of Scientific Reasoning and Applications*.

Dell'Acqua, P., and Pereira, L. M. 2007. Preferential Theory Revision (*extended version*). *Journal of Applied Logic*, 5(4):586–601

Foot, P. 1967. The Problem of Abortion and the Doctrine of Double Effect. *Oxford Review*, 5, 5–15.

Gelfond, M., and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In: Kowalski, R., and Bowen, K. A. (eds), 5th Intl. Logic Programming Conf. MIT Press.

Gigerenzer, G ., and Engel, C . (eds). 2006. *Heuristics and the Law*. MIT Press.

Guarini, M. 2005. Particularism and Generalism: How AI Can Help Us to Better Understand Moral Cognition. In: Anderson, M., Anderson, S., and Armen, C. (eds), *Machine ethics: Papers from the AAAI Fall Symposium*. AAAI Press.

Hauser, M. D. 2007. *Moral Minds: How Nature Designed Our Universal Sense of Right and Wrong*. Little Brown.

Joyce, R. 2006. *The Evolution of Morality*. The MIT Press.

Kakas, A., Kowalski, R., and Toni, F. 1998. The Role of Abduction in Logic Programming. Pages 235–324 of: Gabbay, D., Hogger, C., and Robinson, J. (eds), *Handbook of Logic in Artificial Intelligence and Logic Programming*, vol. 5. Oxford U. P.

Kamm, F. M. 2006. *Intricate Ethics: Rights, Responsibilities, and Permissible Harm*. Oxford U. P.

Katz, L. D . (ed). 2002. *Evolutionary Origins of Morality, Cross-Disciplinary Perspectives*. Imprint Academic.

Kowalski, R. 2006. The Logical Way to be Artificially Intelligent. Page 122 of: Toni, F., and Torroni, P. (eds), *Procs. of CLIMA VI, LNAI*. Springer.

Lopes, G., and Pereira, L. M. 2006. Prospective Logic Programming with ACORDA. In: *Procs. of the FLoC'06, Workshop on Empirically Successful Computerized Reasoning*, 3rd Intl. J. Conf. on Automated Reasoning.

McLaren, B. M. 2006. Computational Models of Ethical Reasoning: Challenges, Initial Steps, and Future Directions. *IEEE Intelligent Systems*, 21(4), 29–37.

Mikhail, J. 2007. Universal Moral Grammar: Theory, Evidence, and The Future. *Trends in Cognitive Sciences*, 11(4), 143–152.

Otsuka, M. 2008. Double Effect, Triple Effect and the Trolley Problem: Squaring the Circle in Looping Cases. *Utilitas*, 20(1), 92–110.

Pereira, L. M., and Lopes, G. 2007. Prospective Logic Agents. In: Neves, J. M., Santos, M. F., and Machado, J. M. (eds), *Procs. 13th Portuguese Intl.Conf. on Artificial Intelligence (EPIA'07)*. Springer LNAI.

Pereira, L. M., and Saptawijaya, A. 2007. Moral Decision Making with ACORDA. In: Dershowitz, N., and Voronkov, A. (eds), *Short papers call, Local Procs. 14th Intl. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR'07)*.

Powers, T. M. 2006. Prospects for a Kantian Machine. *IEEE Intelligent Systems*, 21(4), 46–51.

Rzepka, R., and Araki, K. 2005. What Could Statistics Do for Ethics? The Idea of a Commonsense-Processing-Based Safety Valve. In: Anderson, M., Anderson, S., and Armen, C. (eds), *Machine ethics: Papers from the AAAI Fall Symposium*. AAAI Press.

Tancredi, L. 2005. *Hardwired Behavior: What Neuroscience Reveals about Morality*. Cambridge U. P.