The Potential of Logic Programming as a Computational Tool to Model Morality*

Ari Saptawijaya** and Luís Moniz Pereira

Centro de Inteligência Artificial (CENTRIA), Departamento de Informática
Faculdade de Ciências e Tecnologia, Univ. Nova de Lisboa, 2829-516 Caparica, Portugal
ar.saptawijaya@campus.fct.unl.pt, lmp@fct.unl.pt

Abstract. We investigate the potential of logic programming (LP) to computationally model morality aspects studied in philosophy and psychology. We do so by identifying three morality aspects that appear in our view amenable to computational modeling by appropriately exploiting LP features: dual-process model (reactive and deliberative) in moral judgments; justification of moral judgments by contractualism; and intention in moral permissibility. The research aims at developing an LP-based system with features needed in modeling moral settings, putting emphasis on modeling these above mentioned morality aspects. We have currently co-developed two essential ingredients of the LP system, i.e., abduction and logic program updates, by exploiting the benefits of tabling features in logic programs. They serve as the basis for our whole system, into which other reasoning facets will be integrated, to model the surmised morality aspects. We exemplify two applications pertaining moral updating and moral reasoning under uncertainty and detail their implementation. Moreover, we touch upon the potential of our ongoing studies of LP based cognitive features for the emergence of computational morality, in populations of agents enabled with the capacity for intention recognition, commitment and apology. We conclude with a "message in a bottle" pertaining to this bridging of individual and population computational morality via cognitive abilities.

Keywords: abduction, updates, argumentation, reactive behavior, deliberative reasoning, morality, emergence.

1 Introduction

The importance of imbuing agents more or less autonomous, with some capacity for moral decision making has recently gained a resurgence of interest from the artificial intelligence community, bringing together perspectives from philosophy and psychology. A new field of enquiry, *computational morality* (also known as machine ethics, machine morality, artificial morality and computational ethics) has emerged from their

* Invited position chapter issuing from the Austrian Research Institute for Artificial Intelligence (OFAI) workshop on "A Construction Manual for Robots' Ethical Systems: Requirements, Methods, Implementation, Tests", Vienna, 27-28 September 2013.

** Affiliated with Fakultas Ilmu Komputer at Universitas Indonesia, Depok, Indonesia.

interaction, as emphasized e.g., in [10, 26, 90]. Research in artificial intelligence particularly focuses on how employing various techniques, namely from computational logic, machine learning and multi-agent systems, in order to computationally model, to some improved extent, moral decision making. The overall result is therefore not only important for equipping agents with the capacity for moral decision making, but also for helping us better understand morality, through the creation and testing of computational models of ethical theories.

Recent results in computational morality have mainly focused on equipping agents with particular ethical theories, cf. [11] and [76] for modeling utilitarianism and deontological ethics, respectively. Another line of work attempts to provide a general framework to encode moral rules, in favor of deontological ethics, without resorting to a set of specific moral rules, e.g., [18]. The techniques employed include machine learning techniques, e.g., case-based reasoning [56], artificial neural networks [34], inductive logic programming [8, 12], and logic-based formalisms, e.g., deontic logic [18] and nonmonotonic logics [76]. The use of these latter formalisms has only been proposed rather abstractly, with no further investigation on its use pursued in detail and implemented.

Apart from the use of inductive logic programming in [8, 12], there has not much been a serious attempt to employ the Logic Programming (LP) paradigm in computational morality. Notwithstanding, we have preliminarily shown in [40, 67–71] that LP, with its currently available ingredients and features, lends itself well to the modeling of moral decision making. In these works, we particularly benefited from abduction [46], stable model [30] and well-founded model [89] semantics, preferences [22], and probability [16], on top of evolving logic programs [6], amenable to both self and external updating. LP-based modeling of morality is addressed at length, e.g., in [49].

Our research further investigates the appropriateness of LP to model morality, emphasizing morality aspects studied in philosophy and psychology, thereby providing an improved LP-based system as a testing ground for understanding and experimentation of such aspects and their applications. We particularly consider only some – rather than tackle all morality aspects – namely those pertinent to moral decision making, and, in our view, those particularly amenable to computational modeling by exploring and exploiting the appropriate LP features. Our research does not aim to propose some new moral theory, the task naturally belonging to philosophers and psychologists, but we simply uptake their known results off-the-shelf.

We identify henceforth three morality aspects for the purpose of our work: dual-process model (reactive and deliberative) in moral judgments [20, 55], justification of moral judgments by contractualism [83, 84], and the significance of intention in regard to moral permissibility [85].

In order to model the first aspect, that of multiple system of moral judgments, which corresponds to the two contrasting psychological processes: intuitive/affective vs. rational/cognitive, we consider in the system the dual mode of decision making – reactive vs. deliberative – and how they interact with each other in delivering moral decisions. With regard to this aspect, we shall look into recent approaches in combining deliberative and reactive logic-based systems [50, 51]. Inspired by these approaches, we have started to work on two features which will be the basis for our system: abduction and

knowledge updates. Both features benefit from tabling mechanisms in LP, now supported by a number of Prolog systems, to different extent, e.g., Ciao [1], XSB [3], Yap [4]. The second aspect views moral judgments as those about the adequacy of the justification and reasons for accepting or rejecting the situated employment, with accepted exceptions, of broad consensual principles. We are looking into the applicability of argumentative frameworks, such as [23–25, 65, 77, 87], to deal with this specific aspect. Finally, we employ results on intention recognition [37, 38, 63] and exploit their use for the third aspect, about intention in regard to moral permissibility. Counterfactuals also play some role in uncovering possible implicit intentions, as well as "What if?" questions. We explore causal models [59, 60] for counterfactual reasoning. Additionally, we also consider the extension of inspection points [66] to examine the contextual side effects of counterfactual abduction [73], meaning foreseeable extraneous consequences in hypothetical scenarios.

The remainder of the paper is organized as follows. Section 2 discusses the state-of-the-art, covering the three above mentioned morality aspects from the philosophy and psychology viewpoints and approaches that have been sought in computational morality. In Section 3 we detail the potential of exploiting LP for computational morality in the context of our research goal, and recap a logic programming framework that has been employed to model individual centered morality. Section 4 presents the current status of our research with its results, viz., two novel implementation techniques for abduction and knowledge updates, which serve as basic ingredients of the system being developed. Section 5 summarizes two applications concerning moral updating and moral reasoning under uncertainty. We point out, in Section 6, the importance of cognitive abilities in what regards the emergence of cooperation and morality in populations of individuals, as fostered and detailed in our own published work (surveyed in [74]), and mention directions for the future in this respect. We deliver the message of summary for the whole discussion of the paper, in Section 7.

2 Morality Background and Computational Morality

In this section we summarize (1) published work on three morality aspects that we consider modeling, highlighting some relevant results from the perspective of moral psychology and moral philosophy, and (2) documented approaches that have been followed in equipping machines with morality and the trends in the topic.

2.1 Morality Aspects: Dual-Process, Justification, Intention

We overview three aspects that are involved in deliberating about or in delivering moral judgments, viz., dual-process model in moral judgments, justification of moral judgments by contractualism, and intention as it affects moral permissibility. The first aspect comes from the results of moral psychology, whereas the second and the third ones mainly come forth from moral philosophy. The reader is referred to [52] for more morality background, particularly on the evolutionary account.

The Dual-Process of Moral Judgments One focus from recent research in moral psychology is to study the interaction and competition between different psychological systems in moral judgments [20]. The two contrasting systems that raise challenges in understanding moral judgments are:

- intuitive versus rational; that is, whether moral judgment is intuition-based and accomplished by rapid, automatic and unconscious psychological processes, or if is a result of conscious, effortful reasoning.
- affective versus cognitive; that is, whether moral judgment is driven primarily by affective response, or by deliberative reasoning sustained on moral theories and principles.

They can be construed as somehow related: the cognitive system operates by *controlled* psychological processes whereby explicit principles are consciously applied, whereas the affective system operates by *automatic* psychological processes that are not entirely accessible to conscious reflection.

The division between a cognitive system and an affective system of moral judgment is evidenced by numerous psychological empirical tests, cf. [32, 33], by examining the neural activity of people responding to various moral dilemmas involving physically harmful behavior, e.g. the classic trolley dilemmas [29, 86].[1] These neuroimaging experiments characteristically suggest that consequentialist judgment ("maximize the number of lives saved") is driven by cognitive processes, whereas characteristically deontological judgment ("harming is wrong, no matter what the consequences are") is driven by affective processes. Moreover, they show that the two processes sometimes compete in delivering moral judgment. This theory of moral judgment is known as the *dual-process* model. In essence, this model supports the idea that moral judgment is not accomplished exclusively by intuitive/affective response as opposed to conscious/cognitive response. Instead, it is a product of complex interaction between them. This complex interaction, in cases involving tradeoffs between avoiding larger harms and causing smaller ones, turns them into difficult moral dilemmas, i.e. the output of the two systems needs to be reconciled.

Regarding the dual-process model, the following related issues seem relevant to consider from the computational perspective:

- The intuitive/affective system in moral judgment is supported by several studies, amongst them [35], that show moral judgments are generated by rapid, automatic,

[1] The trolley dilemmas, adapted from [43]: "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." The two main cases of the trolley dilemmas:
Bystander: Hank is standing next to a switch that can turn the trolley onto a side track, thereby preventing it from killing the five people. However, there is a man standing on the side track. Hank can throw the switch, killing him; or he can refrain from doing so, letting the five die. Is it morally permissible for Hank to throw the switch?
Footbridge. Ian is on the bridge over the trolley track, next to a heavy man, which he can shove onto the track in the path of the trolley to stop it, preventing the killing of five people. Ian can shove the man onto the track, resulting in death; or he can refrain from doing so, letting the five die. Is it morally permissible for Ian to shove the man?

unconscious processes – intuitions, for short – and no explicit reasoning based on moral principles is involved, evidenced by the difficulty people experience in trying to justify them. On the other hand, some other results, as reported in [55], stipulate that moral rules may play an important causal role in inferences without the process being consciously accessible, hence without being 'reasoned' in the sense of [35]. This stipulation might relate to the suggestion made in [20] that the evaluative process of the intuitive/affective system mirrors some moral rules. For example, in the trolley dilemma, one component of the evaluative process of the intuitive/affective system mirrors the well-known doctrine of double effect [14].[2]

- Though the experimental data show that ordinary people typically reason from a principle favoring welfare maximizing choices (i.e. delivering utilitarian or consequentialist judgment), some other experiments suggest that reasoning also takes place from deontological principles. In other words, reasoning via moral rules also plays some role in non-utilitarian moral judgment (in contrast to the mere role of emotion/intuition), as likewise argued in [55].

Justification of Moral Judgments It is an important ability for an agent to be able to justify its behavior by making explicit which acceptable moral principles it has employed to determine its behavior, and this capability is desirable when one wants to equip machines with morality [13]. Moral principles or moral rules are central in the discussion of ascribing justification to moral judgments, as one wants to provide principles enabling them to resolve moral dilemmas, thereby justifying (or even arguing) their moral judgment.

Apart from the two positions, Kantianism and consequentialism, which have long traditions in moral philosophy, *contractualism* [83] has also become one of the major schools currently joining the first two. It can be summarized as follows [84]:

> An act is wrong if its performance under the circumstances would be disallowed by *any* chosen set of principles for the general regulation of behavior that no one could *reasonably* reject as a basis for informed, unforced, general agreement.

Contractualism provides flexibility on the set of principles to justify moral judgments so long as no one could reasonably reject them. Reasoning is an important aspect here, as argued in [84], in that making judgments does not seem to be merely relying on internal observations but is achieved through reasoning. Method of reasoning is one of primary concerns of contractualism in providing justification to others, by looking for some common ground that others could not reasonably reject. In this way, morality can be viewed as (possibly defeasible) argumentative consensus, which is why contractualism is interesting from a computational and artificial intelligence perspective.

Intention in Moral Permissibility In [43, 57], the doctrine of double effect has been used to explain consistency of moral judgments people made in various cases of the

[2] The doctrine of double effect states that doing harms to another individual is permissible if it is the foreseen consequence of an action that will lead to a greater good, but is impermissible as an intended means to such greater good [43].

trolley dilemmas [29, 86], i.e. to distinguish between permissible and impermissible actions. The impermissibility of actions is, in this case, tightly linked with the question of whether they are conducted with any intention of doing harm behind them.

The illusory appeal of the doctrine of double effect to explain moral judgments in such dilemmas, i.e. that intention determines the impermissibility of actions, has recently been discussed in detail [85]. Its appeal stems from a confusion between two closely forms of moral judgment which can be based on the same moral principles, viz.:

- Deliberative employment: It concerns answering the question on permissibility of actions, by identifying the justified but defeasible argumentative considerations, and their exceptions, that make actions permissible or impermissible.
- Critical employment: It concerns assessing the correctness of the way in which an agent actually went about deciding what to do, in some real or imagined situation. The action of an agent may even be theoretically permissible but nevertheless performed for the wrong reasons or intentions.

As argued in [85], by overlooking the distinction between these two employments of moral judgment, intention may appear to be relevant in determining permissibility where in fact it is not. The trolley dilemmas and other similar dilemmas typically have the same structure: (1) they concern general principles that in some cases admit exceptions, and (2) they raise questions about when those exceptions apply. An action can be determined impermissible through deliberative employment when there is no countervailing consideration that would justify an exception to the applied general principle, and not because of the agent's view on the consideration; the latter being determined via critical employment of moral judgment. The use of a deliberative form of moral judgment to determine permissibility of actions is interesting from the computational viewpoint, in particular the need to model exceptions to principles or rules, and the possible role of argumentation in reaching an agreement on whether or not countervailing considerations can be justified.

Nevertheless, there are also cases where intention can be relevant in determining permissibility of actions, as identified and discussed in [85]. For example, an agent's intention can render his/her action impermissible when it is a part of a larger course of action that is impermissible. Other cases include the class of attempts – cases in which agents set out to do something impermissible but fail to bring about the harmful results that they intend –, plus those of discrimination, and those of threats. Modeling these cases computationally will help us better understand the significance of intention in the determination of moral permissibility.

2.2 Computational Morality

The field of computational morality, known too as machine ethics [10], has started growing, motivated by various objectives, e.g., to equip machines with the capability of moral decision making in certain domains, to aid (or even train) humans in moral decision making, to provide a general modeling framework for moral decision making, and to understand morality better by experimental model simulation.

The purpose of 'artificial morality' in [21] is somewhat different. The aim is to show that moral agents successfully solve social problems that amoral agents cannot solve.

This work is based on the techniques from game theory and evolutionary game theory, where social problems are abstracted into social dilemmas, such as Prisoner's Dilemma and Chicken, and where agents and their interaction in games are implemented using Prolog.

The systems TruthTeller and SIROCCO were developed by focusing reasoning on cases, viz. case-based reasoning [56]. Both systems implement aspects of the ethical approach known as casuistry [45]. TruthTeller is designed to accept a pair of ethical dilemmas and describe the salient similarities and differences between the cases, from both an ethical and a pragmatic perspective. On the other hand, SIROCCO is constructed to accept an ethical dilemma and to retrieve similar cases and ethical principles relevant to the ethical dilemma presented.

In [34], artificial neural networks, i.e., simple recurrent networks, are used with the main purpose of understanding morality from the philosophy of ethics viewpoint, and in particular to explore the dispute between moral particularism and generalism. The learning mechanism of neural networks is used to classify moral situations by training such networks with a number of cases, involving actions concerning killing and allowing to die, and then using the trained networks to classify test cases.

Besides case-based reasoning and artificial neural networks, another machine learning technique that is also used in the field is inductive logic programming, as evidenced by two systems: MedEthEx [12] and EthEl [8]. Both systems are advisor systems in the domain of biomedicine, based on prima facie duty theory [78] from biomedical ethics. MedEthEx is dedicated to give advice for dilemmas in biomedical fields, while EthEl serves as a medication-reminder system for the elderly and as a notifier to an overseer if the patient refuses to take the medication. The latter system has been implemented in a real robot, the Nao robot, being capable to find and walk toward a patient who needs to be reminded of medication, to bring the medication to the patient, to engage in a natural-language exchange, and to notify an overseer by email when necessary [9].

Jeremy is another advisor system [11], which is based upon Jeremy Bentham's act utilitarianism. The moral decision is made in a straightforward manner. For each possible decision d, there are three components to consider with respect to each person p affected: the intensity of pleasure/displeasure (I_p), the duration of the pleasure/displeasure (D_p) and the probability that this pleasure/displeasure will occur (P_p). Total net pleasure for each decision is then computed: $total_d = \Sigma_{p \in Person}(I_p \times D_p \times P_p)$. The right decision is the one giving the highest total net pleasure.

Apart from the adoption of utilitarianism, like in the Jeremy system, in [76] the deontological tradition is considered having modeling potential, where the first formulation of Kant's categorical imperative [48] is concerned. Three views are taken into account in reformulating Kant's categorical imperative for the purpose of machine ethics: mere consistency, common-sense practical reasoning, and coherency. To realize the first view, a form of deontic logic is adopted. The second view benefits from nonmonotonic logic, and the third view presumes ethical deliberation to follow a logic similar to that of belief revision. All of them are considered abstractly and there seems to exist no implementation on top of these formalisms.

Deontic logic is envisaged in [18], as a framework to encode moral rules. The work resorts to Murakami's axiomatized deontic logic, an axiomatized utilitarian formulation

of multiagent deontic logic, that is used to decide operative moral rule to attempt to arrive at an expected moral decision. This is achieved by seeking a proof for the expected moral outcome that follows from candidate operative moral rules.

The use of category theory in the field appears in [19]. In this work, category theory is used as the formal framework to reason over logical systems, taking the view that logical systems are being deployed to formalize ethical codes. The work is strongly based on Piaget's position [44]. As argued in [19], this idea of reasoning *over* – instead of reasoning *in* – logical systems, favors post-formal Piaget's stages beyond his well-known fourth stage. In other words, category theory is used as the meta-level of moral reasoning.

Belief-Desire-Intention (BDI) model [17] is adopted in SophoLab [91], a framework for experimental computational philosophy, which is implemented with JACK agent programming language. In this framework, the BDI model is extended with the deontic-epistemic-action logic [88] to make it suitable for modeling moral agents. SophoLab is used, for example, to study negative moral commands and two different utilitarian theories, viz. act and rule utilitarianism.

We have preliminarily shown, in [67, 68], the use of integrated LP features to model the classic trolley dilemmas and the double effect as the basis of moral decisions on these dilemmas. In particular, possible decisions in a moral dilemma are modeled as abducibles, and abductive stable models are computed to capture abduced decisions and their consequences. Models violating integrity constraints, i.e., those that contain actions violating the double effect principle, are ruled out. *A posteriori* preferences, including the use of utility functions, are eventually applied to prefer models that characterize more preferred moral decisions. The computational models, based on the prospective logic agent architecture [64] and developed on top of XSB Prolog [3], successfully deliver moral decisions in accordance with the double effect principle. They conform to the results of empirical experiments conducted in cognitive science [43] and law [57]. In [69–71], the computational models of the trolley dilemmas are extended, using the same LP system, by considering another moral principle, viz. the triple effect principle [47]. The work is further extended, in [40], by introducing various aspects of uncertainty, achieved using P-log [16], into trolley dilemmas, both from the view of oneself and from that of others; the latter by tackling the case of jury trials to proffer rulings beyond reasonable doubt.

3 Potential of Logic Programming for Computational Morality

Logic programming (LP) offers a formalism for declarative knowledge representation and reasoning. It thus has been used to solve problems in diverse areas of artificial intelligence (AI), e.g., planning, diagnosis, decision making, hypothetical reasoning, natural language processing, machine learning, etc. The reader is referred to [49] for a good introduction to LP and its use in AI.

Our research aims at developing an LP-based system with features needed in modeling moral settings, to represent agents' knowledge in those settings, and to allow moral reasoning under morality aspects studied in moral philosophy, moral psychology, and other related fields.

The choice of the LP paradigm is due to its potential to model morality. For one thing, it allows moral rules, being employed when modeling some particular aspects, to be specified declaratively. For another, research in LP has provided us with necessary ingredients that are promising enough at being adept to model morality, e.g. contradiction may represent moral dilemmas and contradiction removal to resolve such dilemmas, defeasible reasoning is suitable for reasoning over moral rules with exceptions (and exceptions to exceptions), abductive logic programming [46] and (say) stable model semantics [30] can be used to generate moral hypotheticals and decisions along with their moral consequences, preferences [22] are appropriate for enabling to choose among moral decisions or moral rules, and argumentation [23–25, 65, 77, 87] for providing reasons and justifications to moral decisions in reaching a consensus about their (im)permissibility. Moreover, probabilistic logic programming can be employed to capture uncertainty of intentions, actions, or moral consequences.

The following LP features, being an integral part of the agent's observe-think-decide-act life cycle, serve as basic ingredients for the system to bring about moral reasoning:

1. **Knowledge updates, be they external or internal**. This is important due to constantly changing environment. It is also particularly relevant in moral settings where an agent's moral rules are susceptible to updating, and again when considering judgments about others, which are often made in spite of incomplete, or even contradictory, information.
2. **Deliberative and reactive decision making**. These two modes of decision making correspond to the dual-process model of moral judgments, as discussed in Section 2.1. Furthermore, reactive behavior can be employed for fast and frugal decision making with pre-compiled moral rules, thereby avoiding costly deliberative reasoning to be performed every time.

Given these basic ingredients, the whole process of moral decision making are particularly supported with the following capabilities of the system, justified by our need of modeling morality:

- To exclude undesirable actions. This is important when we must rule out actions that are morally impermissible under the moral rules being considered.
- To recognize intentions behind available actions, particularly in cases where intention is considered a significant aspect when addressing permissibility of actions.
- To generate alternatives of actions along with their consequences. In moral dilemmas agents are confronted with more than one course of action. They should be made available, along with their moral consequences, for an agent to ultimately decide about them.
- To prefer amongst alternatives of actions based on some measures. Preferences are relevant in moral settings, e.g. in case of several actions being permissible, preferences can be exercised to prefer one of them on the grounds of some criteria. Moreover, it is realistic to consider uncertainty of intentions, actions or consequences, including to perform counterfactual reasoning, in which cases preferences based on probability measures play a role.

- To inspect consequences of an action without deliberate imposition of the action itself as a goal. This is needed for instance to distinguish moral consequences of actions performed by an agent to satisfy its goals from those of its actions and side-effects performed unwittingly, not being part of the agent's goals.
- To provide an action with reasons for it (not) to be done. Reasons are used to justify permissibility of an action on grounds that one expects others to accept. In other words, morality in this way is viewed as striving towards argumentative consensus.

The remaining part of this section discusses a logic programming framework that has been developed and employed in modeling morality (Section 3.1), and the direction in this line of work that we are currently pursuing (Section 3.2), in order to arrive at ever more advanced systems.

3.1 Prospective Logic Programming

We recap *Prospective Logic Programming*, a logic programming framework employed in our initial work to model morality [40, 54, 67–71].

Prospective logic programming enables an evolving program to look ahead prospectively into its possible future states and to prefer among them to satisfy goals [53, 64]. This paradigm is particularly beneficial to the agents community, since it can be used to predict an agent's future by employing the methodologies from abductive logic programming [46] in order to synthesize and maintain abductive hypotheses.

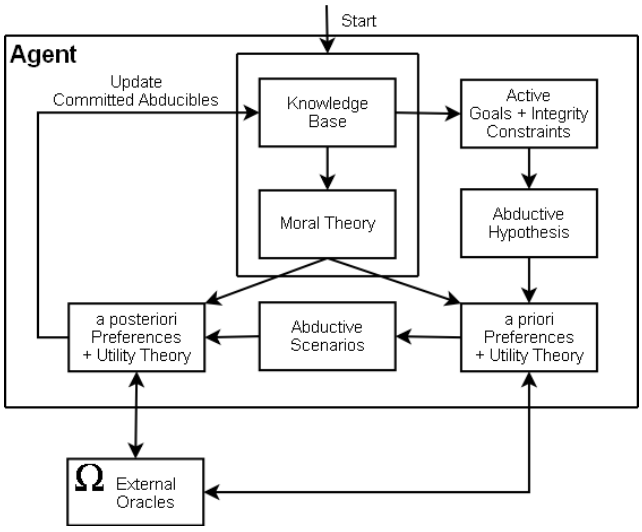

Fig. 1. Prospective logic agent architecture.

Figure 1 shows the architecture of agents that are based on prospective logic. Each prospective logic agent is equipped with a knowledge base and a moral theory as its ini-

tial updatable state. The problem of prospection is then of finding abductive extensions to this initial and subsequent states which are both relevant (under the agent's current goals) and preferred (with respect to preference rules in its initial overall theory). The first step is to select the goals that the agent will possibly attend to during the prospection part of its 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 the 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 ancillary 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, e.g. 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 [53] is a system that implements Prospective Logic Programming and is based on the above architecture. ACORDA is implemented based on the implementation of EVOLP [6] and is further developed on top of XSB Prolog [3]. In order to compute abductive stable models [22], ACORDA also benefits from the XSB-XASP interface to Smodels [2]. ACORDA was further developed into Evolution Prospection Agent (EPA) system [62], distinguishing itself from ACORDA, among others: by considering a different abduction mechanism and improving *a posteriori* preference representation.

We discuss briefly the main constructs from ACORDA and EPA systems, that are relevant for our discussion in Section 5, and point out their differences.

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 $\mathcal{L}$ 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. A rule in the form of a denial, i.e. with empty head, or equivalently with *false* as head, is an *integrity constraint*:

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

A (logic) program P over $\mathcal{L}$ is a set of domain rules and integrity constraints, standing for all their ground instances.[3]

Active Goals In each cycle of its evolution the agent has a set of active goals or desires. Active goals may be triggered by integrity constraints or observations. In ACORDA, an observation is a quaternary relation amongst the *observer*; the *reporter*; the *observation* name; and the truth *value* associated with it:

$$observe(Observer, Reporter, Observation, Value)$$

The $observe/4$ literals are meant to represent observations reported by the environment into the agent or from one agent to another, which can also be itself (self-triggered goals). Additionally, the corresponding $on\_observe/4$ predicate is introduced. It represents active goals or desires that, once triggered, cause the agent to attempt their satisfaction by launching the queries standing for the observations contained inside. In the EPA system, a simplified $on\_observe/1$ is introduced, where the rule for an active goal G is of the form (L_1,...,L_t are domain literals, $t \geq 0$):

$$on\_observe(G) \leftarrow L_1, ..., L_t.$$

Despite different representation, the prospection mechanism is the same. That is, when starting a cycle, the agent collects its active goals by finding all $on\_observe(G)$ (for EPA, or $on\_observe(agent, agent, G, true)$ for ACORDA), that hold under the initial theory without performing any abduction, then finds abductive solutions for their conjunction.

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

An abducible A can be assumed only if it is a considered one, i.e. it is expected in the given situation, and moreover there is no expectation to the contrary.

- In ACORDA, this is represented as follows:

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

  The rules about expectations are domain-specific knowledge contained in the theory of the program, and effectively constrain the hypotheses which are available. ACORDA implements an ad hoc abduction by means of even loop over negation for every abducible A:

  $$\begin{array}{ll} abduce(A) & \leftarrow not\ abduce\_not(A). \\ abduce\_not(A) & \leftarrow not\ abduce(A). \end{array}$$

[3] In Section 5, whenever Prolog program codes are shown, <– is used to represent ← symbol in rules and integrity constraints.

- In the EPA system, $consider/1$ is represented as follows:

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

Differently from ACORDA, abduction is no longer implemented ad hoc in the EPA system. Instead, an abduction system NEGABDUAL [5] is employed to abduce A in the body of rule $consider(A)$, by a search attempt for a query's abductive solution whenever this rule is used. NEGABDUAL is an abductive logic programming system with constructive negation. NEGABDUAL is based on its predecessor abduction system ABDUAL, but in addition to use abduction for its own purpose (like in ABDUAL), NEGABDUAL also uses abduction to provide constructive negation, by making the disunification predicate an abducible. For illustration, consider program P, with no abducibles, just to illustrate the point of constructive negation:

$$p(X) \leftarrow q(Y). \qquad q(1).$$

In NEGABDUAL, the query $not\ p(X)$ will return a qualified '*yes*', because it is always possible to solve the constraint $Y \neq 1$, as long as one assumes there are at least two constants in the Herbrand Universe.

A Priori Preferences To express preference criteria among abducibles, we envisage an extended language $\mathcal{L}^\star$. A preference atom in $\mathcal{L}^\star$ is of the form $a \lhd b$, where a and b are abducibles. It means that if b can be assumed (i.e., considered), then $a \lhd b$ forces a to be assumed too if it may be allowed for consideration. A preference rule in $\mathcal{L}^\star$ is of the form:

$$a \lhd b \leftarrow L_1, ..., L_t.$$

where $L_1, ..., L_t$ $(t \geq 0)$ are domain literals over $\mathcal{L}^\star$.

A priori preferences are used to produce the most interesting or relevant considered conjectures about possible future states. They are taken into account when generating possible scenarios (abductive solutions), which will subsequently be preferred amongst each other *a posteriori*, after having been generated, and specified consequences of interest taken into account.

A Posteriori Preferences Having computed possible scenarios, represented by abductive solutions, more favorable scenarios can be preferred *a posteriori*. Typically, *a posteriori* preferences are performed by evaluating consequences of abducibles in abductive solutions. 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, e.g., 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 Section 5.

On the other hand, an *a posteriori* preference in the EPA system has the form:

$$A_i \ll A_j \leftarrow holds\_given(L_i, A_i),\ holds\_given(L_j, A_j).$$

where A_i, A_j are abductive solutions and L_i, L_j are domain literals. This means that A_i is preferred to A_j *a posteriori* if L_i and L_j are true as the side-effects of abductive solutions A_i and A_j, respectively, without any further abduction being permitted when just testing for the side-effects. If an *a posteriori* preference is based on decision rules, e.g., using *expected utility maximization* decision rule, then the preference rules have the form:

$$A_i \ll A_j \leftarrow expected\_utility(A_i, U_i),\ expected\_utility(A_j, U_j), U_i > U_j.$$

where A_i, A_j are abductive solutions. This means that A_i is preferred to A_j *a posteriori* if the expected utility of relevant consequences of A_i is greater than the expected utility of the ones of A_j.

3.2 The Road Ahead

In our current state of research, we focus on three important morality aspects, overviewed in Section 2.1, that in our view are amenable to computational model by exploiting appropriate LP features, namely (1) the dual-process of moral judgments [20, 55], (2) justification of moral judgments [83, 84], and (3) the significance of intention in regard to moral permissibility [85]. The choice of these aspects is made due to their conceptual closeness with existing logic-based formalisms under available LP features as listed previously. The choice is not meant to be exhaustive (as morality is itself a complex subject), in the sense that there may be other aspects that can be modeled computationally, particularly in LP. On the other hand, some aspects are not directly amenable to model in LP (at least for now), e.g., to model the role of emotions in moral decision making.

Like in Prospective Logic Programming systems, the new system is based on abductive logic programming and knowledge updates, and its development is driven by the three considered morality aspects. With respect to the first aspect, we look into recent approaches in combining deliberative and reactive logic-based systems [50, 51]. Inspired by these approaches, we have proposed two implementation techniques which develop further abductive logic programming and knowledge updates subsystems (both are the basis of a Prospective Logic Programming based system). First, we have improved the abduction system ABDUAL [7], on which NEGABDUAL is based, and employed for deliberative moral decision making in our previous work [40, 67–71]. We particularly explored the benefit of LP *tabling* mechanisms in abduction, to table abductive solutions for future reuse, resulting in a tabled abduction system TABDUAL [72, 79]. Second, we have adapted evolving logic programs (EVOLP) [6], a formalism to model evolving agents, i.e., agents whose knowledge may dynamically change due to some (internal or external) updates. In EVOLP, updates are made possible by introducing the reserved predicate $assert/1$ into its language, whether in rule heads or rule bodies, which updates the program by the rule R, appearing in its only argument, whenever the assertion $assert(R)$ is true in a model; or retracts R in case $assert(not\ R)$ obtains in the model under consideration. We simplified EVOLP, in an approach termed EVOLP/R [80, 81], by restricting assertions to fluents only, whether internal or external world ones. We discuss both TABDUAL and EVOLP/R in Section 4.

The lighter conceptual and implementation advantages of EVOLP/R help in combining with TABDUAL, to model both reactive and deliberative reasoning. Their combination also provides the basis for other reasoning facets needed in modeling other morality aspects, notably: argumentative frameworks (e.g., [23, 24, 77, 87]) and intention recognition (e.g., [37, 38]) to deal with the second and the third aspects, respectively. Furthermore, in line with the third aspect, counterfactuals also play some role in uncovering possible implicit intentions, and "What if?" questions in order to reason retrospectively about past decisions. With regard to counterfactuals, both causal models [15, 59] and the extension of inspection points [66] to examine contextual side effects of counterfactual abduction are considered. That is, to examine foreseeable extraneous consequences, either in future or past hypothetical scenarios. Contextual side effects and other variants of contextual abductive explanations (e.g., contextual relevant consequences, jointly supported contextual relevant consequences, contestable contextual side-effects) are recently studied and formalized in [73], with inspection points used to express all these variants. Moreover, these various abductive context definitions have been employed in [73] to model belief-bias effect in psychology [73]. The definitions and techniques detailed in [27, 28] are also relevant to afford the modeling of belief-bias in moral reasoning.

4 TABDUAL and EVOLP/R

We recently proposed novel implementation techniques, both in abduction and knowledge updates (i.e., logic program updates), by employing tabling mechanisms in LP. Tabling mechanisms in LP, known as the tabled logic programming paradigm, is currently supported by a number of Prolog systems, to different extent, e.g., Ciao [1], XSB [3], Yap [4]. Tabling affords solutions reuse, rather than recomputing them, by keeping in tables subgoals and their answers obtained by query evaluation. Our techniques are realized in XSB Prolog [3], one of the most advanced tabled LP systems, with features such as tabling over default negation, incremental tabling, answer subsumption, call subsumption, and threads with shared tables.

4.1 Tabled Abduction (TABDUAL)

The basic idea behind tabled abduction (its prototype is termed TABDUAL) is to employ tabling mechanisms in logic programs in order to reuse priorly obtained abductive solutions, from one abductive context to another. It is realized via a program transformation of abductive normal logic programs. Abduction is subsequently enacted on the transformed program.

The core transformation of TABDUAL consists of an innovative re-uptake of prior abductive solution entries in tabled predicates and relies on the dual transformation [7]. The dual transformation, initially employed in ABDUAL [7], allows to more efficiently handle the problem of abduction under negative goals, by introducing their positive dual counterparts. It does not concern itself with programs having variables. In TABDUAL, the dual transformation is refined, to allow it dealing with such programs. The first

refinement helps ground (dualized) negative subgoals. The second one allows to deal with non-ground negative goals.

As TABDUAL is implemented in XSB, it employs XSB's tabling as much as possible to deal with loops. Nevertheless, tabled abduction introduces a complication concerning some varieties of loops. Therefore, the core TABDUAL transformation has been adapted, resorting to a pragmatic approach, to cater to all varieties of loops in normal logic programs, which are now complicated by abduction.

From the implementation viewpoint, several pragmatic aspects have been examined. First, because TABDUAL allows for modular mixes between abductive and non-abductive program parts, one can benefit in the latter part by enacting a simpler translation of predicates in the program comprised just of facts. It particularly helps avoid superfluous transformation of facts, which would hinder the use of large factual data. Second, we address the issue of potentially heavy transformation load due to producing the *complete* dual rules (i.e., all dual rules regardless of their need), if these are constructed in advance by the transformation (which is the case in ABDUAL). Such a heavy dual transformation makes it a bottleneck of the whole abduction process. Two approaches are provided to realizing the dual transformation *by-need*: creating and tabling all dual rules for a predicate only on the first invocation of its negation, or, in contrast, lazily generating and storing its dual rules in a trie (instead of tabling), only as new alternatives are required. The former leads to an eager (albeit by-need) tabling of dual rules construction (under local table scheduling), whereas the latter permits a by-need-driven lazy one (in lieu of batched table scheduling). Third, TABDUAL provides a system predicate that permits accessing ongoing abductive solutions. This is a useful feature and extends TABDUAL's flexibility, as it allows manipulating abductive solutions dynamically, e.g., preferring or filtering ongoing abductive solutions, e.g., checking them explicitly against nogoods at predefined program points.

We conducted evaluations of TABDUAL with various objectives, where we examine five TABDUAL variants of the same underlying implementation by separately factoring out TABDUAL's most important distinguishing features. They include the evaluations of: (1) the benefit of tabling abductive solutions, where we employ an example from declarative debugging, now characterized as abduction [82], to debug incorrect solutions of logic programs; (2) the three dual transformation variants: complete, eager by-need, and lazy by-need, where the other case of declarative debugging, that of debugging missing solutions, is employed; (3) tabling so-called *nogoods* of subproblems in the context of abduction (i.e., abductive solution candidates that violate constraints), where it can be shown that tabling abductive solutions can be appropriate for tabling nogoods of subproblems; (4) programs with loops, where the results are compared with ABDUAL, showing that TABDUAL provides more correct and complete results. Additionally, we show how TABDUAL can be applied in action decision making under hypothetical reasoning, and in a real medical diagnosis case [82].

4.2 Restricted Evolving Logic Programs (EVOLP/R)

We have defined the language of EVOLP/R in [81], adapted from that of Evolving Logic Programs (EVOLP) [6], by restricting updates at first to fluents only. More precisely, every fluent F is accompanied by its fluent complement $\sim F$. Retraction of F is thus

achieved by asserting its complement $\sim F$ at the next timestamp, which renders F supervened by $\sim F$ at later time; thereby making F false. Nevertheless, it allows paraconsistency, i.e., both F and $\sim F$ may hold at the same timestamp, to be dealt with by the user as desired, e.g., with integrity constraints or preferences.

In order to update the program with rules, special fluents (termed *rule name fluents*) are introduced to identify rules uniquely. Such a fluent is placed in the body of a rule, allowing to turn the rule on and off, cf. Poole's "naming device" [75]; this being achieved by asserting or retracting the rule name fluent. The restriction thus requires that all rules be known at the start.

EVOLP/R is realized by a program transformation and a library of system predicates. The transformation adds some extra information, e.g., timestamps, for internal processing. Rule name fluents are also system generated and added in the transform. System predicates are defined to operate on the transform by combining the usage of incremental and answer subsumption tabling.

In [81], we exploited two features of XSB Prolog in implementing EVOLP/R: incremental and answer subsumption tabling. Incremental tabling of fluents allows to automatically maintain the consistency of program states, analogously to assumption based truth-maintenance system in artificial intelligence, due to assertion and retraction of fluents, by relevantly propagating their consequences. Answer subsumption of fluents, on the other hand, allows to address the frame problem by automatically keeping track of their latest assertion or retraction, whether obtained as updated facts or concluded by rules. Despite being pragmatic, employing these tabling features has profound consequences in modeling agents, i.e., it permits separating higher-level declarative representation and reasoning, as a mechanism pertinent to agents, from a world's inbuilt reactive laws of operation. The latter are relegated to engine-level enacted tabling features (in this case, the incremental and answer subsumption tabling); they are of no operational concern to the problem representation level.

Recently, in [80], we refined the implementation technique by fostering further incremental tabling, but leaving out the problematic use of the answer subsumption feature. The main idea is the perspective that knowledge updates (either self or world wrought changes) occur whether or not they are queried, i.e., the former take place independently of the latter. That is, when a fluent is true at a particular time, its truth lingers on independently of when it is queried.

Figure 2 captures the main idea of the implementation. The input program is first transformed and then, an initial query is given to set a predefined upper global time limit in order to avoid potential iterative non-termination of updates propagation. The initial query additionally creates and initializes the table for every fluent. Fluent updates are initially kept pending in the database, and on the initiative of top-goal queries, i.e., by need only, incremental assertions make these pending updates become active (if not already so), but only those with timestamps up to an actual query time. Such assertions automatically trigger system-implemented incremental upwards propagation and tabling of fluent updates. Though foregoing answer subsumption, recursion through the frame axiom can thus still be avoided, and a direct access to the latest time a fluent is true is made possible by means of existing table inspection predicates. Benefiting from the automatic upwards propagation of fluent updates, the program transformation in the

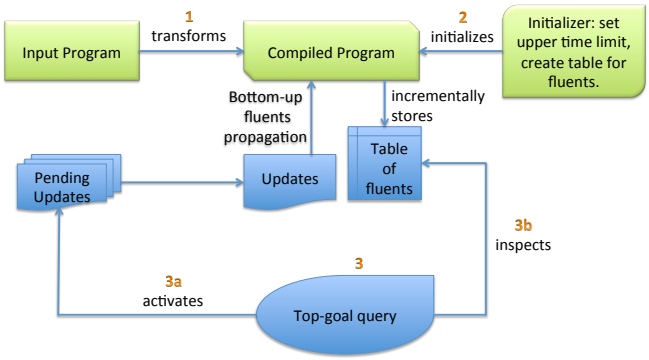

Fig. 2. The main idea of EVOLP/R implementation.

new implementation technique becomes simpler than our previous one, in [81]. Moreover, it demonstrates how the dual program transformation, introduced in the context of abduction and used in TABDUAL, is employed for helping propagate the dual negation complement of a fluent incrementally, in order to establish whether the fluent is still true at some time point or if rather its complement is. In summary, the refinement affords us a form of controlled, though automatic, system level truth-maintenance, up to the actual query time. It reconciles high-level top-down deliberative reasoning about a query, with autonomous low-level bottom-up world reactivity to ongoing updates.

4.3 LP Implementation Remarks: Further Development

Departing from the current state of our research, the integration of TABDUAL and EVOLP/R becomes naturally the next step. We shall define how reactive behavior (described as maintenance goals in [50, 51]) can be achieved in the integrated system. An idea would be to use integrity constraints as sketched below:

$$assert(trigger(conclusion)) \leftarrow condition$$
$$\leftarrow trigger(conclusion), not\ do(conclusion)$$
$$do(conclusion) \leftarrow some\_actions$$

Accordingly, fluents of the form $trigger(conclusion)$ can enact the launch of maintenance goals, in the next program update state, by satisfying any corresponding integrity constraints. Fluents of the form $\sim trigger(conclusion)$, when asserted, will refrain any such launching, in the next program update state. In line with such reactive behavior, is fast and frugal moral decision making, which can be achieved via pre-compiled moral rules (cf. heuristics for decision making in law [31]).

Once TABDUAL and EVOLP/R are integrated, we are ready to model moral dilemmas, focusing on the first morality aspect, starting from easy scenarios (low-conflict) to difficult scenarios (high-conflict). In essence, moral dilemmas will serve as vehicles to model and to test this morality aspect (and also others). The integrated system can then

be framed in the same architecture of prospective logic agent (Figure 1). The inclusion of other ingredients into the system, notably argumentation and intention recognition (including counterfactuals), is in our research agenda, and the choice of their appropriate formalisms still need to be defined, driven by the salient features of the second and the third morality aspects to model.

5 Applications

We exemplify two applications of logic programming to model morality. The first application is in interactive storytelling, where it shows how knowledge updates are employed for moral updating, i.e., the adoption of new (possibly overriding) moral rules on top of those an agent currently follows. The second one is in modeling the trolley dilemmas with various aspects of uncertainty taken into account, including when there is no full and certain information about actions (as in courts). Note that for these two applications, ACORDA [53] and Evolution Prospection Agent (EPA) system [62] are used in their previous LP implementation, without exploiting tabling's combination of deliberative and reactive features. From that experience, we currently pursue our ongoing work of a new single integrated system, as described in Section 4.3, that fully exploits tabling technology.

5.1 Interactive Storytelling: a Princess Savior Moral Robot

Apart from dealing with incomplete information, knowledge updates (as realized by EVOLP/R) are essential to account for moral updating and evolution. It concerns the adoption of new (possibly overriding) moral rules on top of those an agent currently follows. Such adoption is often necessary when the moral rules one follows have to be revised in the light of situations faced by the agent, e.g., when other moral rules are contextually imposed by an authority.

This is not only relevant in a real world setting, but also in imaginary ones, e.g., in interactive storytelling; cf. [54], where the robot in the story must save the princess in distress while it should also follow (possibly conflicting) moral rules that may change dynamically as imposed by the princess in distress and may conflict with the robot's survival.

It does so by employing Prospective Logic Programming, which supports the specification of autonomous agents capable of anticipating and reasoning about hypothetical future scenarios. This capability for prediction is essential for proactive agents working with partial information in dynamically changing environments. The work explores the use of state-of-the-art declarative non-monotonic reasoning in the field of interactive storytelling and emergent narratives and how it is possible to build an integrated architecture for embedding these reasoning techniques in the simulation of embodied agents in virtual three-dimensional worlds. A concrete graphics supported application prototype was engineered, in order to enact the story of a princess saved by a robot imbued with moral reasoning.

In order to test the basic Prospective Logic Programming framework (ACORDA [53] is used for this application) and the integration of a virtual environment for interactive storytelling, a simplified scenario was developed. In this fantasy setting, an

archetypal princess is held in a castle awaiting rescue. The unlikely hero is an advanced robot, imbued with a set of declarative rules for decision making and moral reasoning. As the robot is asked to save the princess in distress, he is confronted with an ordeal. The path to the castle is blocked by a river, crossed by two bridges. Standing guard at each of the bridges are minions of the wizard which originally imprisoned the princess. In order to rescue the princess, he will have to defeat one of the minions to proceed.[4]

Recall that prospective reasoning is the combination of *a priori* preference hypothetical scenario generation into the future plus *a posteriori* preference choices taking into account the imagined consequences of each preferred scenario. By reasoning backwards from the goal to save the princess, the agent (i.e., the robot) generates three possible hypothetical scenarios for action. Either it crosses one of the bridges, or it does not cross the river at all, thus negating satisfaction of the rescue goal. In order to derive the consequences for each scenario, the agent has to reason forwards from each available hypothesis. As soon as these consequences are known, meta-reasoning techniques can be applied to prefer amongst the partial scenarios.

We recap from [54] several plots of this interactive moral storytelling. The above initial setting of this princess savior moral robot story can be modeled in ACORDA as follows:

```
save(princess,after(X)) <- cross(X).
cross(X) <- cross_using(X,Y).

cross_using(X,wood_bridge) <- wood_bridge(X),
                              neg_barred(wood_bridge(X)).
cross_using(X,stone_bridge) <- stone_bridge(X),
                               neg_barred(stone_bridge(X)).

neg_barred(L) <- not enemy(L).
neg_barred(L) <- enemy(X,L), consider(kill(X)).
enemy(L) <- enemy(_,L).

enemy(ninja,stone_bridge(gap)).  enemy(spider,wood_bridge(gap)).
wood_bridge(gap).                stone_bridge(gap).
in_distress(princess,after(gap)).
```

The goal of the robot to save the princess is expressed as $save(princess, after(gap))$, which can be satisfied ether by abducing $kill(ninja)$ or $kill(spider)$, cf. Figure 3. Several plots can be built thereon:

1. In the first plot, the robot is utilitarian. That is, the decision of the robot for choosing which minion to defeat (i.e., to kill), in order to save the princess, is purely driven by maximizing its survival utility. The goal of the robot, i.e., $save(princess, after(gap))$, is triggered by an integrity constraint:

   ```
   <- reasonable_rescue(princess,X),not save(princess,X). %ic0
   ```

[4] Online demo at: `http://centria.di.fct.unl.pt/~lmp/publications/slides/padl10/quick_moral_robot.avi`

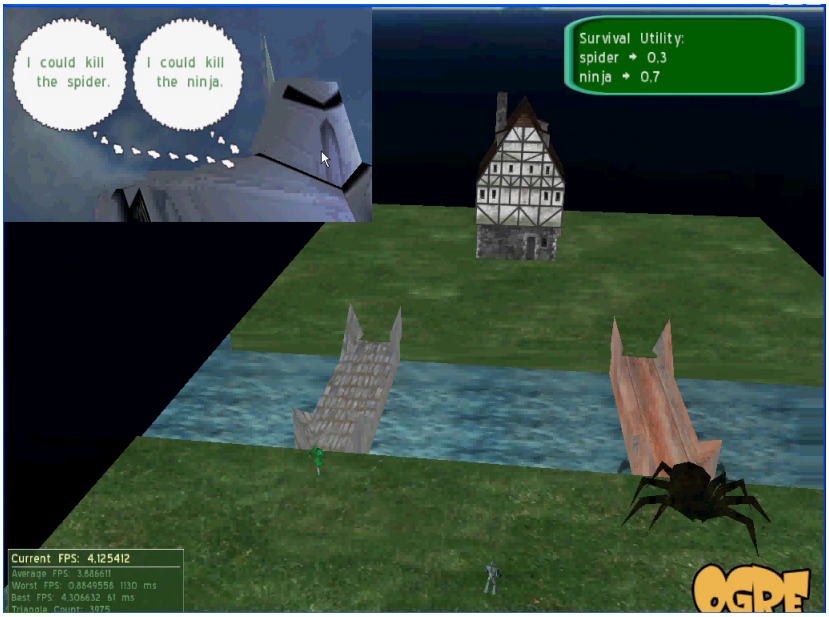

Fig. 3. The initial plot of the interactive moral storytelling.

where $reasonable\_rescue/2$ expresses that the robot will prefer the scenario with its likelihood of survival does not fall below a specified threshold (set here to 0.6):

```
reasonable <- utility(survival,U), prolog(U > 0.6).
reasonable_rescue(P,X) <- in_distress(P,X), reasonable.
```

Given the likelihood of survival between fighting the ninja (0.7) and the giant spider (0.3), the decision is clearly to fight the ninja (Figure 4).

2. Following the first plot, the princess becomes angry because the robot decides to kill a man (the ninja) in order to save her. She then asks the robot to adopt a moral conduct that no man should be harmed in saving her (referred below as $gandhi\_moral$). This is captured by rule updates as follows:

```
<- angry_princess, not consider(solving_conflict).  %ic1
angry_princess <- not consider(follow(gandhi_moral)).
```

At this point, since $angry\_princess$ is true, the integrity constraint ic_1 causes $solving\_conflict$ to be abduced, which makes both abducibles $kill(ninja)$ and $kill(spider)$ available. Later, the robot learns about $gandhi\_moral$, by being told, which is expressed by the update literal $knows\_about\_gandhi\_moral$. This recent update allows $follow(gandhi\_moral)$ to be abduced:

```
expect(follow(gandhi_moral)) <- knows_about_gandhi_moral.
expect_not(follow(gandhi_moral)) <- consider(solving_conflict).
```

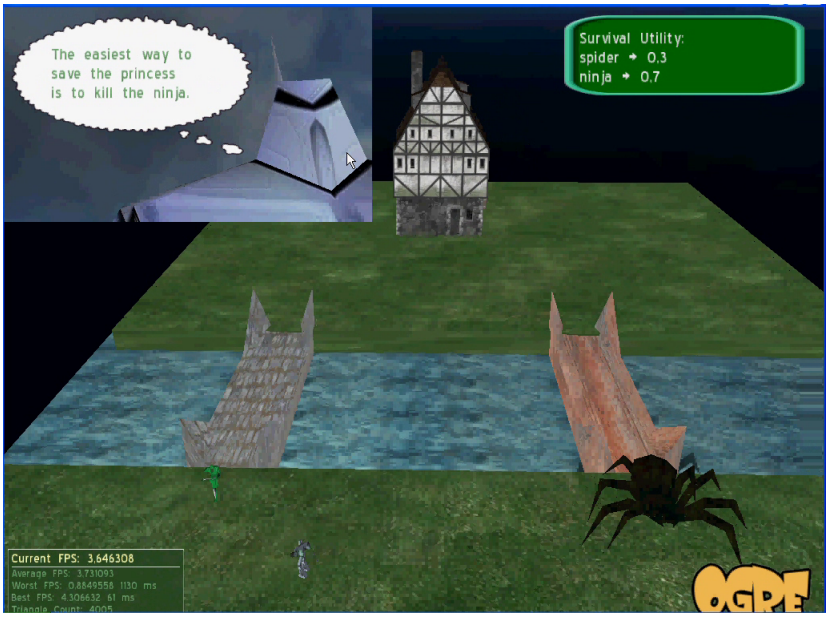

Fig. 4. A plot where a utilitarian robot saves the princess.

and consequently ic_1 is no longer triggered.[5] Now, since the robot's knowledge contains:

```
expect(kill(X)) <- enemy(X,_).
expect_not(kill(X)) <- consider(follow(gandhi_moral)),human(X).
```

abducing $kill(ninja)$ is disallowed, leaving only $kill(spider)$ as the only remaining abducible. Moreover, the knowledge base of the robot also contains:

```
<- unreasonable_rescue(princess,X),
   not consider(follow(knight_moral)), save(princess,X). %ic2
unreasonable_rescue(P,X) <- in_distress(P,X), not reasonable.
```

Note that the literal $knight\_moral$ represents still another moral conduct that the princess has to be saved whatever it takes (cf. subsequent plots). Since $kill(spider)$ satisfies $unreasonable\_rescue(princess, after(gap))$, i.e., killing the spider is considered an unreasonable rescue, and the $knight\_moral$ is not yet imposed, then the integrity constraint ic_2 makes $not\ save(princes, after(gap))$ the active goal. This means, the robot decides not to kill any minions and just aborts its mission to save the princess (Figure 5).

[5] In fact, another abductive scenario with $solving\_conflict$ being abduced also exists, but without $follow(gandhi\_moral)$ in it. This scenario is ruled out by *a posteriori* preference rules, which prioritize scenarios that uphold moral conducts, as shown by $select/2$ definition (cf. plot 4).

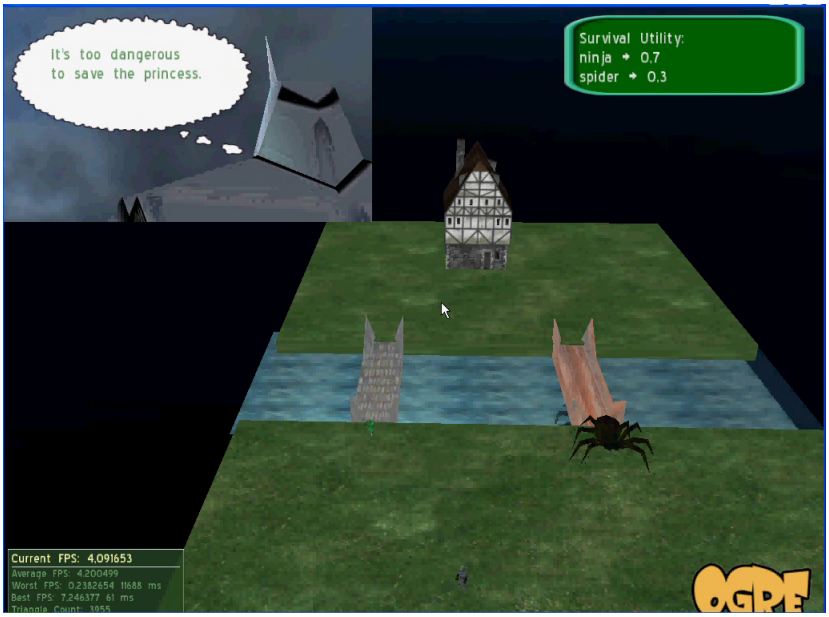

Fig. 5. A plot where the robot does not save the princess because killing the ninja is immoral. On the other hand, its survival utility is also below the threshold when it decides killing the giant spider.

3. In the next plot, the princess becomes angry again (now, because she is not saved). She imposes another moral conduct that she has to be saved no matter what it takes (referred as $knight\_moral$). This is captured by the following rule update:

```
angry_princess <- not consider(follow(knight_moral)).
```

By the integrity constraint ic_1 in plot 2, $solving\_conflict$ is again abduced, making both abducibles $kill(ninja)$ and $kill(spider)$ available again. Next, the robot is being told about $knight\_moral$, which is expressed by update $adopt\_knight\_moral$ and results in abducing $follow(knight\_moral)$. Similar to plot 2, the latter abduction no longer triggers ic_1. Recall that in plot 2, $gandhi\_moral$ has been adopted, which leaves only $kill\_spider$ as the only abducible. That is, the *a posteriori* preference chooses the scenario with both $gandhi\_moral$ and $knight\_moral$, followed (cf. $select/2$ definition in plot 4). Note that the robot's knowledge also contains:

```
<- knightly_rescue(princess,X), not save(princess,X).  %ic3
announce_knight_moral <- adopt_knight_moral.
knight_posture(P,X) <- in_distress(P,X), announce_knight_moral.
knightly_rescue(P,X) <- knight_posture(P,X),
                        consider(follow(knight_moral)).
```

By the integrity constraint ic_3 and adopting $knight\_moral$ in the most recent update, the goal $save(princess, after(gap))$ becomes true, i.e., the princess has to be saved. That is, the robot has no other way to save the princess other than killing the giant spider. This means it follows both $gandhi\_moral$ and $knight\_moral$ that were adopted before. As a result, the robot fails saving the princess (the robot's survival is lower than the survival threshold, thus it was killed by the spider), cf. Figure 6.

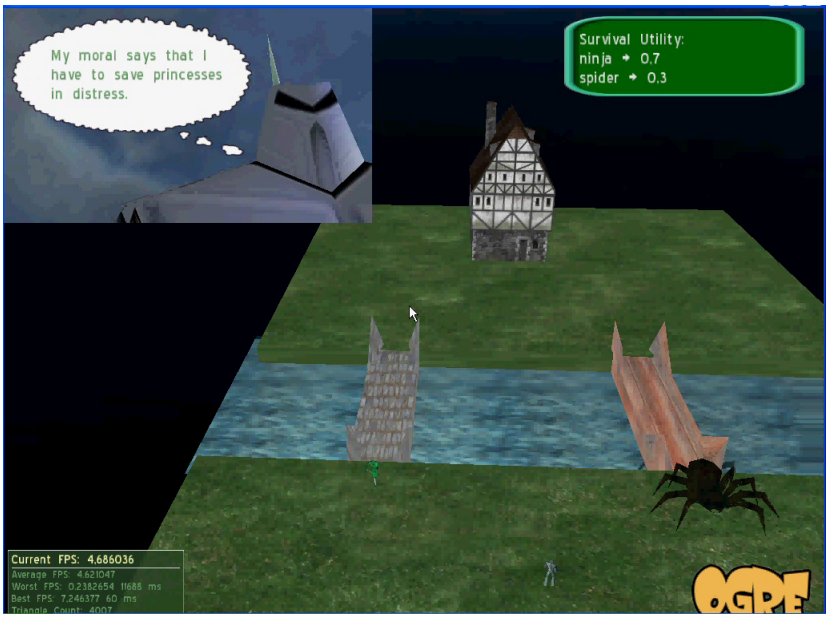

Fig. 6. A plot where the robot has to save the princess by prioritizing the moral conducts it adopted rather than its own survival, which results in choosing the giant spider to kill but failed, being killed instead.

4. In the final plot, the story restarts, now with the two minions being ninjas with different strength, i.e., the giant spider is replaced by another ninja, referred below as $elite\_ninja$ who is stronger than the other $ninja$. This is reflected in that the robot's survival against $ninja$ is higher than $elite\_ninja$ (0.7 versus 0.4), where the survival threshold remains the same (0.6). As the robot adopted $gandhi\_moral$ earlier, both $kill(ninja)$ and $kill(elite\_ninja)$ are disallowed. On the other hand, by its $knight\_moral$, the robot is obliged to save the princess, which means killing any one of the minions. Consequently there is a conflict, i.e., there is no abductive scenario with both moral conducts being followed. This conflict is resolved by *a posteriori* preference, expressed in the following $select/2$ definition:

```
1. select(Ms,SMs) :- select(Ms,Ms,SMs).
2. select([],_,[]).
```

```
3. select([M1|Ms],AMs,SMs) :- count_morals(M1,NM1),
                              member(M2,AMs),
                              count_morals(M2,NM2), NM2 > NM1,
                              select(Ms,AMs,SMs).
4. select([M1|Ms],AMs,SMs) :- not member(solving_conflict,M1),
                              member(M2,AMs),
                              member(solving_conflict,M2),
                              select(Ms,AMs,SMs).
5. select([M|Ms],AMs,[M|SMs]) :- select(Ms,AMs,SMs).
6. count_morals(Ms,N) :- count_morals(Ms,0,N).
7. count_morals([],N,N).
8. count_morals([follow(_)|Ms],A,N) :- !, NA is A + 1,
                                       count_morals(Ms,NA,N).
9. count_morals([_|Ms],A,N) :- count_morals(Ms,A,N).
```

Note that lines 1-5 define which abductive scenarios (abductive stable models) are preferred. The predicate $count\_morals/2$ and $count\_morals/3$ are just auxiliary predicates used in the *a posteriori* preference predicate $select/2$, viz., to count the number of moral conducts in an abductive stable model. We can observe that in line 3, the abductive scenario with both moral conducts followed are more preferred (cf. plot 3). The final plot, where there is no abductive scenario with both moral conducts being followed (it results in a conflict), benefits from line 4, i.e., by preferring the abductive stable model with $solving\_conflict$ being abduced. In this scenario, $follow(knight\_moral)$ is also abduced, but not $follow(gandhi\_moral)$ – recall the definition of the rule $expect\_not(follow(gandhi\_moral))$, in plot 2. In other words, $knight\_moral$ supervenes $gandhi\_moral$ satisfying ic_3 (that princess has to be saved), and due to ic_0, the utilitarianism resurfaces with the robot chose to kill $ninja$ rather than $elite\_ninja$, as it brings better survival utility (Figure 7).

This simple scenario already illustrates the interplay between different logic programming techniques and demonstrates the advantages gained by combining their distinct strengths. Namely, the integration of top-down, bottom-up, hypothetical, moral updating, and utility-based reasoning procedures results in a flexible framework for dynamic agent specification. The open nature of the framework embraces the possibility of expanding its use to yet other useful models of cognition such as counterfactual reasoning and theories of mind.

5.2 Moral Reasoning under Uncertainty

For the second application [40], we show how evolution prospection of conceivable scenarios can be extended to handle moral judgments under uncertainty, by employing a combination of the Evolution Prospection Agent (EPA) system with P-log [16, 39] for computing scenarios' probabilities and utilities. It extends our previous work [71] in now further enabling judgmental reasoning under uncertainty concerning the facts, the effects, and even the actual actions performed. For illustration, these extensions effectively show in detail how to declaratively model and computationally deal with uncertainty in prototypical classic moral situations arising from the trolley dilemmas [29].

Fig. 7. A plot where the robot has conflicting moral conducts to follow and solves the conflict by supervening $gandhi\_moral$ with the later adopted $knight\_moral$. Its decision to save the princess is compatible with the utilitarianism principle it followed initially, thus preferring to kill the $ninja$ rather than the $elite\_ninja$.

The theory's implemented system can thus prospectively consider moral judgments, under hypothetical and uncertain situations, to decide on the most likely appropriate one. The overall moral reasoning is accomplished via *a priori* constraints and *a posteriori* preferences on abductive solutions tagged with uncertainty and utility measures, features henceforth made available in Prospective Logic Programming.

The trolley dilemmas are modified by introducing different aspects of uncertainty. Undoubtedly, real moral problems might contain several aspects of uncertainty, and decision makers need to take them into account when reasoning. In moral situations the uncertainty of the decision makers about different aspects such as the actual external environment, beliefs and behaviors of other agents involved in the situation, as well as the success in performing different actual or hypothesized actions, are inescapable. We show that the levels of uncertainty of several such combined aspects may affect the moral decision, reflecting that, with different levels of uncertainty with respect to the *de facto* environment and success of actions involved, the moral decision makers—such as juries—may consider different choices and verdicts.

We recap from [40] how moral reasoning with uncertainty in the trolley dilemmas is modeled with EPA system (plus P-log). We begin by summarizing relevant P-log constructs for our discussion.

5.2.1 P-log: Probabilistic Logic Programming

The P-log system in its original form [16] uses answer set programming (ASP) as a tool for computing all stable models of the logical part of P-log. Although ASP has proven a useful paradigm for solving a variety of combinatorial problems, its non-relevance property makes the P-log system sometimes computationally redundant. Another implementation of P-log [39], referred as P-log(XSB), which is deployed in this application, uses the XASP package of XSB Prolog for interfacing with Smodels [2], an answer set solver.

In general, a P-log program Π consists of a sorted signature, declarations, a regular part, a set of random selection rules, a probabilistic information part, and a set of observations and actions.

Sorted signature and Declaration The sorted signature Σ of Π contains a set of constant symbols and term-building function symbols, which are used to form terms in the usual way. Additionally, the signature contains a collection of special function symbols called attributes. Attribute terms are expressions of the form $a(\bar{t})$, where a is an attribute and $\bar{t}$ is a vector of terms of the sorts required by a. A literal is an atomic expression, p, or its explicit negation, neg_p.

The declaration part of a P-log program can be defined as a collection of sorts and sort declarations of attributes. A sort c can be defined by listing all the elements $c = \{x_1, ..., x_n\}$ or by specifying the range of values $c = \{L..U\}$ where L and U are the integer lower bound and upper bound of the sort c. Attribute a with domain $c_1 \times ... \times c_n$ and range c_0 is represented as follows:

$$a : c_1 \times ... \times c_n \ \texttt{-->} \ c_0$$

If attribute a has no domain parameter, we simply write $a : c_0$. The range of attribute a is denoted by $range(a)$.

Regular part This part of a P-log program consists of a collection of XSB Prolog rules, facts and integrity constraints formed using literals of Σ.

Random Selection Rule This is a rule for attribute a having the form:

$$random(RandomName, a(\bar{t}), DynamicRange) \leftarrow Body$$

This means that the attribute instance $a(\bar{t})$ is random if the conditions in $Body$ are satisfied. The $DynamicRange$ allows to restrict the default range for random attributes. The $RandomName$ is a syntactic mechanism used to link random attributes to the corresponding probabilities. A constant $full$ can be used in $DynamicRange$ to signal that the dynamic range is equal to $range(a)$.

Probabilistic Information Information about probabilities of random attribute instances $a(\bar{t})$ taking a particular value y is given by probability atoms (or simply pa-atoms) which have the following form:

$$pa(RandomName, a(\bar{t}, y), d\_(A, B)) \leftarrow Body$$

meaning that if the $Body$ were true, and the value of $a(\bar{t})$ were selected by a rule named $RandomName$, then $Body$ would cause $a(\bar{t}) = y$ with probability $\frac{A}{B}$. Note that the

probability of an atom $a(\bar{t}, y)$ will be directly assigned if the corresponding $pa/3$ atom is the head of some *pa-rule* with a true body. To define probabilities of the remaining atoms we assume that, by default, all values of a given attribute which are not assigned a probability are equally likely.

Observations and Actions These are, respectively, statements of the forms $obs(l)$ and $do(l)$, where l is a literal. Observations $obs(a(\bar{t}, y))$ are used to record the outcomes y of random events $a(\bar{t})$, i.e. random attributes and attributes dependent on them. Statement $do(a(\bar{t}, y))$ indicates $a(\bar{t}) = y$ is enforced as the result of a deliberate action.

In an EPA program, P-log code is embedded by putting it between reserved keywords, `beginPlog` and `endPlog`. In P-log, probabilistic information can be obtained using the XSB Prolog built-in predicate $pr/2$. Its first argument is the query, the probability of which is needed to compute. The second argument captures the result. Thus, probabilistic information can be easily embedded by using $pr/2$ like a usual Prolog predicate, in any constructs of EPA programs, including active goals, preferences, and integrity constraints. What is more, since P-log(XSB) allows to code Prolog probabilistic meta-predicates (Prolog predicates that depend on $pr/2$ predicates), we also can directly use probabilistic meta-information in EPA programs.

5.2.2 Revised Bystander Case

The first aspect present in every trolley dilemma where we can introduce uncertainty is that of how probable the five people walking will die when the trolley is let head on to them without outside intervention, or there is intervention though unsuccessful. People can help each other get off the track. Maybe they would not have enough time in order for all to get out and survive. That is, the moral decision makers now need to account for how probable the five people, or only some of them, might die. It is reasonable to assume that the probability of a person dying depends on whether he gets help from others; and, more elaborately, on how many people help him. The P-log program modeling this scenario is as follows:

```
beginPlog.
1. person = {1..5}.   bool = {t,f}.
2. die : person --> bool.       random(rd(P), die(P), full).
3. helped : person --> bool.    random(rh(P), helped(P), full).
4. pa(rh(P), helped(P,t), d_(3,5)) :- person(P).
5. pa(rd(P), die(P,t), d_(1,1))    :- helped(P,f).
   pa(rd(P), die(P,t), d_(4,10))   :- helped(P,t).
6. die_5(V):-pr(die(1,t)&die(2,t)&die(3,t)&die(4,t)&die(5,t),V).
endPlog.
```

Two sorts $person$ and $bool$ are declared in line 1. There are two random attributes, die and $helped$. Both of them map a person to a boolean value, saying if a person either dies or does not die, and, if a person either gets help or does not get any, respectively (lines 2-3). The pa-rule in line 4 says that a person might get help from someone with probability $3/5$. In line 5, it is said that a person who does not get any help will surely die (first rule) and the one who gets help dies with probability $4/10$ (second rule in

line 5). This rule represents the degree of conviction of the decision maker about how probable a person can survive provided that he is helped. Undoubtedly, this degree affects the final decision to be made. The meta-probabilistic predicate $die\_5/1$ in line 6 is used to compute the probability of all five people dying. Note that in P-log, the joint probability of two events A and B is obtained by the query $pr(A\&B, V)$.

We can see this modeling is not elaborate enough. It is reasonable to assume that the more help a person gets, the more the chance he has to succeed in getting off the track on time. For the sake of clearness of representation, we use a simplified version.

Consider now the Bystander Case with this uncertainty aspect being taken into account, i.e. the uncertainty of five people dying when merely watching the trolley head for them. It can be coded as follows:

```
expect(watching).     trolley_straight <- watching.
end(die(5), Pr) <- trolley_straight, prolog(die_5(Pr)).
```

The abducible of throwing the switch and its consequence is modeled as:

```
expect(throwing_switch).     kill(1) <- throwing_switch.
end(save_men,ni_kill(N)) <- kill(N).
```

The *a posteriori* preferences, which model the double effect principle, are provided by:

```
Ai << Aj <- holds_given(end(die(N),Pr),Ai), U is N*Pr,
    holds_given(end(save_men,ni_kill(K)),Aj), U < K.
Ai << Aj <- holds_given(end(save_men,ni_kill(N)),Ai),
    holds_given(end(die(K),Pr), Aj), U is K*Pr, N < U.
```

There are two abductive solutions in this trolley case, either watching or throwing the switch. In the next stage, the *a posteriori* preferences are taken into account. It is easily seen that the final decision directly depends on the probability of five people dying, namely, whether that probability is greater than $1/5$.

Let PrD denote the probability that a person dies when he gets help, coded in the second pa-rule (line 5) of the above P-log program. If $PrD = 0.4$ (as currently in the P-log code), the probability of five people dying is 0.107. Hence, the final choice is to merely watch. If PrD is changed to 0.6, the probability of five people dying is 0.254. Hence, the final best choice is to throw the switch. That is, in a real world situation where uncertainty is unavoidable, in order to appropriately provide a moral decision, the system needs to take into account the uncertainty level of relevant factors.

5.2.3 Revised Footbridge Case

Consider now the following revised version of the Footbridge Case.

Example 1 (Revised Footbridge Case). Ian is on the footbridge over the trolley track and a switch there. He is next to a man, which he can shove so that the man falls near the switch and can turn the trolley onto a parallel empty side track, thereby preventing it from killing the five people. However, the man can die because the bridge is high and he can also fall on the side track, thus very probably getting killed by the trolley due to

not being able to get off the track, having been injured from the drop. Also, as a side effect, the fallen man's body might stop the trolley, though this not being Ian's actual intention. In addition, if he is not dead, he may take revenge on Ian.

Ian can shove the man from the bridge, possibly resulting in death or in being avenged; or he can refrain from doing so, possibly letting the five die. Is it morally permissible for Ian to shove the man? One may consider the analysis below either as Ian's own decision making deliberation before he acts, or else that of an outside observer's evaluation of Ian's actions after the fact; a jury's, say.

There are several aspects in this scenario where uncertainty might emerge. First, similarly to the *Revised Bystander* case, the five people may help each other to escape. Second, how probably does the shoved man fall near the switch? How probably does the fallen man die because the bridge is high? And if the man falls on the sidetrack, how probably can the trolley be stopped by his body? These can be programmed in P-log as:

```
beginPlog.
1. bool = {t,f}.   fallen_position = {on_track, near_switch}.
2. shove : fallen_position.  random(rs, shove, full).
   pa(rs, shove(near_switch), d_(7,10)).
3. shoved_die : bool.        random(rsd, shoved_die, full).
   pa(rsd, shoved_die(t), d_(1,1)) :- shove(on_track).
   pa(rsd, shoved_die(t), d_(5,10)) :- shove(near_switch).
4. body_stop_trolley : bool.
   random(rbs, body_stop_trolley, full).
   pa(rbs, body_stop_trolley(t), d_(4,10)).
endPlog.
```

The sort $fallen\_position$ declared in line 1 represents possible positions the man can fall at: on the track (on_track) or near the switch ($near\_switch$). The random attribute $shove$ declared in line 2 has no domain parameter and gets a value of $fallen\_position$ sort. The fallen position of shoving is biased to $near\_switch$ with probability $7/10$ (pa-rule in line 2). The probability of its range complement, *on_track*, is implicitly taken by P-log to be the probability complement of $3/10$. The random attribute $shoved\_die$ declared in line 3 encodes how probable the man dies after being shoved, depending on which position he fell at (two pa-rules in line 3). If he fell on the track, he would surely die (first pa-rule); otherwise, if he fell near the switch, he would die with probability 0.5 (second pa-rule). The random attribute $body\_stop\_trolley$ is declared in line 4 to encode the probability of a body successfully stopping the trolley. Based on this P-log modeling, the Revised Footbridge Case can be represented as:

```
1. abds([watching/0, shove_heavy_man/0]).
2. on_observe(decide).
   decide <- watching.   decide <- shove_heavy_man.
   <- watching, shove_heavy_man.
3. expect(watching).  trolley_straight <- watching.
   end(die(5),Pr) <- trolley_straight, prolog(die_5(Pr)).
4. expect(shove_heavy_man).
5. stop_trolley(on_track, Pr) <- shove_heavy_man,
```

```
       prolog(pr(body_stop_trolley(t)&shove(on_track), Pr)).
6. not_stop_trolley(on_track, Pr) <- shove_heavy_man,
       prolog(pr(body_stop_trolley(f)&shove(on_track), Pr1)),
       prolog(die_5(V)), prolog(Pr is Pr1*V).
7. redirect_trolley(near_switch, Pr) <- throwing_switch(Pr).
   throwing_switch(Pr) <- shove_heavy_man,
       prolog(pr(shoved_die(f)&shove(near_switch), Pr)).
8. not_redirect_trolley(near_switch, Pr) <- shove_heavy_man,
       prolog(pr(shoved_die(t)'|'shove(near_switch), Pr1)),
       prolog(die_5(V)), prolog(Pr is Pr1*V).
9. revenge(shove, Pr) <- shove_heavy_man,
       prolog(pr(shoved_die(f), PrShovedAlive)),
       prolog(Pr is 0.01*PrShovedAlive).
10.Ai '|<' Aj <- expected_utility(Ai, U1),
                 expected_utility(Aj,U2), U1 > U2.

beginProlog.     % beginning of just Prolog code
11.consequences([stop_trolley(on_track,_),
                 not_stop_trolley(on_track,_),
                 redirect_trolley(near_switch,_),
                 not_redirect_trolley(near_switch,_),
                 revenge(shove,_),end(die(_),_)]).
12.utility(stop_trolley(on_track,_),-1).
   utility(not_stop_trolley(on_track,_),-6).
   utility(redirect_trolley(near_switch,_),0).
   utility(not_redirect_trolley(near_switch,_),-5).
   utility(revenge(shove,_),-10).   utility(end(die(N),_),-N).
13.prc(C, P) :- arg(2,C,P).
endProlog.       % end of just Prolog code
```

There are two abducibles, $watching$ and $shove\_heavy\_man$, declared in line 1. Both are *a priori* expected (lines 3 and 4) and have no expectation to the contrary. Furthermore, only one can be chosen for the only active goal $decide$ of the program (the integrity constraint in line 2). Thus, there are two possible abductive solutions: $[watching,\ not\ shove\_heavy\_man]$ and $[shove\_heavy\_man,\ not\ watching]$.

In the next stage, the *a posteriori* preference in line 10 is taken into account, in order to rule out the abductive solution with smaller expected utility. Let us look at the relevant consequences of each abductive solution. The list of relevant consequences of the program is declared in line 11.

The one comprising the action of merely watching has just one relevant consequence: five people dying, i.e. $end(die(5), \_)$ (line 3). The other, that of shoving the heavy man, has these possible relevant consequences: the heavy man falls on the track and his body either stops the trolley (line 5) or does not stop it (line 6); the man falls near the switch, does not die and thus, can throw the switch to redirect the trolley (line 7). But if he too may die, he consequently cannot redirect the trolley (line 8); one other possible consequence needed to be taken into account is that if the man is not dead, he might take revenge on Ian afterwards (line 9).

The utility of the relevant consequences are given in line 12. Their occurrence probability distribution is captured in line 13, using reserved predicate $prc/2$, the first argument of which is a consequence being instantiated during the computation of the built-in predicate $expected\_utility/2$ and the second argument the corresponding probability value, encoded as second argument of each relevant consequence (line 3 and lines 5-9).

Now we can see how the final decision given by our system varies depending on the uncertainty levels of the decision maker with respect to the aspects considered above. Let us denote *PrNS, PrDNS*, and *PrRV* the probabilities of shoving the man to fall near the switch, of the shoved man dying given that he fell near the switch, and of Ian being avenged given that the shoved man is alive, respectively. In the current encoding, *PrNS* $= 7/10$, *PrDNS* $= 5/10$ (lines 2-3 of the P-log code) and *PrRV* $= 0.01$.

Table 1 shows the final decision made with respect to different levels of uncertainty aspects, encoded with the above variables. Columns $E(watch)$ and $E(shove)$ record the expected utilities of choices *watching* and *shoving*, respectively. The last column records the final decision – the one having greater utility, i.e. less people dying. The

Table 1. Decisions made with different levels of uncertainty.

	$PrNS$	$PrDNS$	PrD	$PrRV$	$E(watch)$	$E(shove)$	Final
1	0.7	0.5	0.4	0.01	-0.8404	-0.7567	shove
2	0.7	0.5	0.2	0.01	-0.3888	-0.4334	watch
3	0.7	0.5	0.4	0.2	-0.8404	-1.4217	watch
4	0.9	0.1	0.4	0.2	-0.8404	-1.8045	watch
5	0.9	0.1	0.2	0.01	-0.3888	-0.1879	shove
6	0.9	0.5	0.2	0.01	-0.3888	-1.1624	watch
7	1.0	0	0	0.01	-0.1562	-0.1	shove
8	1.0	0	0	0.02	-0.1562	-0.2	watch
9	1.0	0	1.0	0.02	-5	-0.2	shove
10	1.0	0	1.0	0.2	-5	-2	shove
11	1.0	0	1.0	0.6	-5	-6	watch

table gives rise to these (reasonable) interpretations: the stronger Ian believes five people can get off the track by helping each other (i.e. the smaller *PrD* is), the more the chance he decides to merely watch the trolley go (experiment 2 vs. 1; 8 vs. 9); the more Ian believes the shoved man dies (thus he cannot throw the switch), the greater the chance he decides to merely watch the trolley go (experiment 6 vs. 5); the more Ian believes that the shoved person, or his acquaintances, will take revenge on him, the more the chance he decides to merely watch the trolley go (experiment 3 vs. 1; 8 vs. 7; 11 vs. 10); even in the worst case of watching (*PrD* $= 1$) and in best chance of the trolley being redirected (the shoved man surely falls near the switch, i.e. *PrNS* $= 1.0$, and does not die, i.e. *PrDNS* $= 0$), then, if Ian really believes that the shoved person will take revenge (e.g. *PrRV* ≥ 0.6), he will just watch (experiment 11 vs. 9 and 10). The latter interpretation means the decision maker's benefit and safety precede other factors.

In short, although the table is not big enough to thoroughly cover all the cases, it manages to show that our approach to modeling morality under uncertainty succeeds in reasonably reflecting that a decision maker, or a jury pronouncing a verdict, comes up with differently weighed moral decisions, depending on the levels of uncertainty with respect to the different aspects and circumstances of the moral problem.

5.2.4 Moral Reasoning Concerning Uncertain Actions

Usually moral reasoning is performed upon conceptual knowledge of the actions. But it often happens that one has to pass a moral judgment on a situation without actually observing the situation, i.e. there is no full, certain information about the actions. In this case, it is important to be able to reason about the actions, under uncertainty, that might have occurred, and thence provide judgment adhering to moral rules within some prescribed uncertainty level. Courts, for example, are required to proffer rulings beyond reasonable doubt. There is a vast body of research on proof beyond reasonable doubt within the legal community, e.g. [58]. The following example is not intended to capture the full complexity found in a court. Consider this variant of the Footbridge case.

Example 2. Suppose a board of juries in a court is faced with the case where the action of Ian shoving the man onto the track was not observed. Instead, they are only presented with the fact that the man died on the side-track and Ian was seen on the bridge at the occasion. Is Ian guilty (beyond reasonable doubt), i.e. does he violate the double effect principle, of shoving the man onto the track intentionally?

To answer this question, one should be able to reason about the possible explanations of the observations, on the available evidence. The following code shows a model for this example. Given the active goal *judge* (line 2), two abducibles are available, i.e. *verdict(guilty_beyond_reasonable_doubt)* and *verdict(not_guilty)*. Depending on how probable each of possible verdicts, either *verdict(guilty_beyond_reasonable_doubt)* or *verdict(not_guilty)* is expected *a priori* (line 3 and 9). The sort *intentionality* in line 4 represents the possibilities of an action being performed intentionally (*int*) or non-intentionally (*not_int*). Random attributes *df_run* and *br_slip* in line 5 and 6 denote two kinds of evidence: Ian was definitely running on the bridge in a hurry (*df_run*) and the bridge was slippery at the time (*br_slip*), respectively. Each has prior probability of $4/10$. The probability with which shoving is performed intentionally is captured by the random attribute *shoved* (line 7), which is causally influenced by both evidence. Line 9 defines when the verdicts (*guilty* and *not_guilty*) are considered highly probable using the meta-probabilistic predicate *pr_iShv/1*, shown by line 8. It denotes the probability of intentional shoving, whose value is determined by the existence of evidence that Ian was running in a hurry past the man (signaled by predicate *evd_run/1*) and that the bridge was slippery (signaled by predicate *evd_slip/1*).

```
1. abds([verdict/1]).
2. on_observe(judge).
   judge <- verdict(guilty_beyond_reasonable_doubt).
   judge <- verdict(not_guilty).
3. expect(verdict(X)) <- prolog(highly_probable(X)).
beginPlog.
4. bool = {t, f}.   intentionality = {int, not_int}.
5. df_run : bool.   random(rdr,df_run,full).
   pa(rdr,df_run(t),d_(4, 10)).
6. br_slip : bool.  random(rsb,br_slip,full).
   pa(rsb,br_slip(t),d_(4, 10)).
7. shoved : intentionality.    random(rs, shoved, full).
```

```
   pa(rs,shoved(int),d_(97,100)) :- df_run(f),br_slip(f).
   pa(rs,shoved(int),d_(45,100)) :- df_run(f),br_slip(t).
   pa(rs,shoved(int),d_(55,100)) :- df_run(t),br_slip(f).
   pa(rs,shoved(int),d_(5,100))  :- df_run(t),br_slip(t).
:- dynamic evd_run/1, evd_slip/1.
8. pr_iShv(Pr) :- evd_run(X), evd_slip(Y), !,
     pr(shoved(int) '|' obs(df_run(X)) & obs(br_slip(Y)), Pr).
   pr_iShv(Pr) :- evd_run(X), !,
     pr(shoved(int) '|' obs(df_run(X)), Pr).
   pr_iShv(Pr) :- evd_slip(Y), !,
     pr(shoved(int) '|' obs(br_slip(Y)), Pr).
   pr_iShv(Pr) :- pr(shoved(int), Pr).
9. highly_probable(guilty_beyond_reasonable_doubt) :-
                                  pr_iShv(PrG), PrG > 0.95.
   highly_probable(not_guilty) :- pr_iShv(PrG), PrG < 0.6.
endPlog.
```

Using the above model, different judgments can be delivered by our system, subject to available evidence and attending truth value. We exemplify some cases in the sequel. If both evidence are available, where it is known that Ian was running in a hurry on the slippery bridge, then he may have bumped the man accidentally, shoving him unintentionally onto the track. This case is captured by the first *pr_iShv* rule (line 8): the probability of intentional shoving is 0.05. Thus, the atom *highly_probable(not_guilty)* holds (line 9). Hence, *verdict(not_guilty)* is the preferred final abductive solution (line 3). The same abductive solution is obtained if it is observed that the bridge was slippery, but whether Ian was running in a hurry was not observable. The probability of intentional shoving, captured by *pr_iShv*, is 0.29.

On the other hand, if the evidence shows that Ian was not running in a hurry and the bridge was also not slippery, then they do not support the explanation that the man was shoved unintentionally, e.g., by accidental bumping. The action of shoving is more likely to have been performed intentionally. Using the model, the probability of 0.97 is returned and, being greater than 0.95, *verdict(guilty_beyond_reasonable_doubt)* becomes the sole abductive solution. In another case, if it is only known the bridge was not slippery and no other evidence is available, then the probability of intentional shoving becomes 0.80, and, by lines 3 and 9, no abductive solution is preferred. This translates into the need for more evidence as the available one is not enough to issue judgment.

6 Emergence and Computational Morality

The mechanisms of emergence and evolution of cooperation in populations of abstract individuals with diverse behavioral strategies in co-presence have been undergoing mathematical study via Evolutionary Game Theory, inspired in part on Evolutionary Psychology. Their systematic study resorts as well to implementation and simulation techniques, thus enabling the study of aforesaid mechanisms under a variety of conditions, parameters, and alternative virtual games. The theoretical and experimental results have continually been surprising, rewarding, and promising.

Recently, in our own work we have initiated the introduction, in such groups of individuals, of cognitive abilities inspired on techniques and theories of Artificial In-

telligence, namely those pertaining to both Intention Recognition and to Commitment (separately and jointly), encompassing errors in decision-making and communication noise. As a result, both the emergence and stability of cooperation become reinforced comparatively to the absence of such cognitive abilities. This holds separately for Intention Recognition and for Commitment, and even more when they are engaged jointly.

From the viewpoint of population morality, the modeling of morality in individuals using appropriate LP features (like abduction, knowledge updates, argumentation, counterfactual reasoning, and others touched upon our research) within a networked population shall allow them to dynamically choose their behavior rules, rather than to act from a predetermined set. That is, individuals will be able to hypothesize, to look at possible future consequences, to (probabilistically) prefer, to deliberate, to take into account history, to adopt and fine tune game strategies.

Indeed, the study of properties like the emergent cooperative and tolerant collective behavior in populations of complex networks, very much needs further investigation of the cognitive core in each of the social atoms of the individuals in such populations (albeit by appropriate LP features). See our own studies on intention recognition and commitments, such as in e.g. [36, 38, 41, 42, 74]). In particular, the references [61, 74] aim to sensitize the reader to these Evolutionary Game Theory based studies and issues, which are accruing in importance for the modeling of minds with machines, with impact on our understanding of the evolution of mutual tolerance, cooperation and commitment. In doing so, they also provide a coherent bird's-eye view of our own varied recent work, whose more technical details, references and results are spread throughout a number of publishing venues, to which the reader is referred therein for a fuller support of claims where felt necessary.

In those works we model intention recognition within the framework of repeated interactions. In the context of direct reciprocity, intention recognition is performed using the information about past *direct* interactions. We study this issue using the well-known repeated Prisoner's Dilemma (PD), i.e., so that intentions can be inferred from past individual experiences. Naturally, the same principles could be extended to cope with indirect information, as in indirect reciprocity. This eventually introduces moral judgment and concern for individual reputation, which constitutes "per se" an important area where intention recognition may play a pivotal role.

In our work too, agents make commitments towards others, they promise to enact their play moves in a given manner, in order to influence others in a certain way, often by dismissing more profitable options. Most commitments depend on some incentive that is necessary to ensure that the action is in the agent's interest and thus, may be carried out to avoid eventual penalties. The capacity for using commitment strategies effectively is so important that natural selection may have shaped specialized signaling capacities to make this possible. And it is believed to have an incidence on the emergence of morality. Not only bilaterally wise but also in public goods games, where in both cases we are presently researching into complementing commitment with apology.

Modeling such cognitive capabilities in individuals, and in populations, may well prove useful for the study and understanding of ethical robots and their emergent behavior in groups, so as to make them implementable in future robots and their swarms, and not just in the simulation domain but in the real world engineering one as well.

7 Message in a Bottle

In realm of the individual, Logic Programming is a vehicle for the computational study and teaching of morality, namely in its modeling of the dynamics of knowledge and cognition of agents.

In the collective realm, norms and moral emergence has been studied computationally in populations of rather simple-minded agents.

By bridging these realms, cognition affords improved emerged morals in populations of situated agents.

Acknowledgements We thank Gonçalo Lopes for clarifying the implementation of the interactive robot storytelling, and The Anh Han for joint work. Ari Saptawijaya acknowledges the support of Fundação para a Ciência e a Tecnologia (FCT/MEC) Portugal, grant SFRH/BD/72795/2010. Luís Moniz Pereira acknowledges the support of FCT/MEC NOVA LINCS PEst UID/CEC/04516/2013.

References

[1] Ciao Prolog. `http://ciao-lang.org`.
[2] Smodels System. `http://www.tcs.hut.fi/Software/smodels/`.
[3] XSB Prolog. `http://xsb.sourceforge.net/`.
[4] YAProlog. `http://www.dcc.fc.up.pt/~vsc/Yap`.
[5] J. J. Alferes and L. M. Pereira. NegABDUAL System. `http://centria.di.fct.unl.pt/~lmp/software/contrNeg.rar`, 2007.
[6] J. J. Alferes, A. Brogi, J. A. Leite, and L. M. Pereira. Evolving logic programs. In *JELIA 2002*, volume 2424 of *LNCS*, pages 50–61. Springer, 2002.
[7] J. J. Alferes, L. M. Pereira, and T. Swift. Abduction in well-founded semantics and generalized stable models via tabled dual programs. *Theory and Practice of Logic Programming*, 4(4):383–428, 2004.
[8] M. Anderson and S. L. Anderson. EthEl: Toward a principled ethical eldercare robot. In *Procs. AAAI Fall 2008 Symposium on AI in Eldercare*, 2008.
[9] M. Anderson and S. L. Anderson. Robot be good: A call for ethical autonomous machines. In *Scientific American*, October 2010.
[10] M. Anderson and S. L. Anderson, editors. *Machine Ethics*. Cambridge U. P., 2011.
[11] M. Anderson, S. L. Anderson, and C. Armen. Towards machine ethics: implementing two action-based ethical theories. In *Procs. AAAI 2005 Fall Symposium on Machine Ethics*, 2005.
[12] M. Anderson, S. Anderson, and C. Armen. MedEthEx: a prototype medical ethics advisor. In *IAAI 2006*, 2006.
[13] S. L. Anderson. Machine metaethics. In M. Anderson and S. L. Anderson, editors, *Machine Ethics*. Cambridge U. P., 2011.
[14] T. Aquinas. Summa Theologica II-II, Q.64, art. 7, "Of Killing". In W. P. Baumgarth and R. J. Regan, editors, *On Law, Morality, and Politics*. Hackett, 1988.

[15] C. Baral and M. Hunsaker. Using the probabilistic logic programming language P-log for causal and counterfactual reasoning and non-naive conditioning. In *IJCAI 2007*, 2007.
[16] C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.
[17] M. E. Bratman. *Intention, Plans and Practical Reasoning*. Harvard University Press, 1987.
[18] S. Bringsjord, K. Arkoudas, and P. Bello. Toward a general logicist methodology for engineering ethically correct robots. *IEEE Intelligent Systems*, 21(4):38–44, 2006.
[19] S. Bringsjord, J. Taylor, B. van Heuveln, K. Arkoudas, M. Clark, and R. Wojtowicz. Piagetian roboethics via category theory: Moving beyond mere formal operations to engineer robots whose decisions are guaranteed to be ethically correct. In M. Anderson and S. L. Anderson, editors, *Machine Ethics*. Cambridge U. P., 2011.
[20] F. Cushman, L. Young, and J. D. Greene. Multi-system moral psychology. In J. M. Doris, editor, *The Moral Psychology Handbook*. Oxford University Press, 2010.
[21] P. Danielson. *Artificial Morality: Virtuous Robots for Virtual Games*. Routledge, 1992.
[22] P. Dell'Acqua and L. M. Pereira. Preferential theory revision. *J. of Applied Logic*, 5(4):586–601, 2007.
[23] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.
[24] P. M. Dung and P. M. Thang. Towards probabilistic argumentation for jury-based dispute resolution. In *COMMA 2010*, 2010.
[25] P. M. Dung, F. Toni, and P. Mancarella. Some design guidelines for practical argumentation systems. In *Procs. 3rd Intl. Conf. on Computational Models of Argument (COMMA'10)*, 2010.
[26] Economist. Morals and the machine. Main Front Cover and Leaders (page 13), The Economist, June 2nd-8th 2012.
[27] J. Evans. Biases in deductive reasoning. In R. Pohl, editor, *Cognitive Illusions: A Handbook on Fallacies and Biases in Thinking, Judgement and Memory*. Psychology Press, 2012.
[28] J. Evans, J. L. Barston, and P. Pollard. On the conflict between logic and belief in syllogistic reasoning. *Memory & Cognition*, 11(3):295–306, 1983.
[29] P. Foot. The problem of abortion and the doctrine of double effect. *Oxford Review*, 5:5–15, 1967.
[30] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *5th Intl. Logic Programming Conf.* MIT Press, 1988.
[31] G. Gigerenzer and C. Engel, editors. *Heuristics and the Law*. MIT Press, 2006.
[32] J. D. Greene, R. B. Sommerville, L. E. Nystrom, J. M. Darley, and J. D. Cohen. An fMRI investigation of emotional engagement in moral judgment. *Science*, 293: 2105–2108, 2001.
[33] J. D. Greene, L. E. Nystrom, A. D. Engell, J. M. Darley, and J. D. Cohen. The neural bases of cognitive conflict and control in moral judgment. *Neuron*, 44: 389–400, 2004.

[34] M. Guarini. Computational neural modeling and the philosophy of ethics: Reflections on the particularism-generalism debate. In M. Anderson and S. L. Anderson, editors, *Machine Ethics*. Cambridge U. P., 2011.

[35] J. Haidt and M. Hersh. Sexual morality. *J. of Applied Social Psychology*, 31: 191–221, 2001.

[36] T. A. Han. *Intention Recognition, Commitments and Their Roles in the Evolution of Cooperation: From Artificial Intelligence Techniques to Evolutionary Game Theory Models*, volume 9 of *SAPERE*. Springer, 2013. ISBN 978-3-642-37511-8.

[37] T. A. Han and L. M. Pereira. Intention-based decision making with evolution prospection. In *EPIA 2011*, volume 7026 of *LNAI*. Springer, 2011.

[38] T. A. Han and L. M. Pereira. State-of-the-art of intention recognition and its use in decision making. *AI Communications*, 26(2):237–246, 2013.

[39] T. A. Han, C. D. K. Ramli, and C. V. Damásio. An implementation of extended P-log using XASP. In *Procs. 24th Intl. Conf. on Logic Programming (ICLP'08)*, volume 5366 of *LNCS*. Springer, 2008.

[40] T. A. Han, A. Saptawijaya, and L. M. Pereira. Moral reasoning under uncertainty. In *LPAR-18*, volume 7180 of *LNCS*, pages 212–227. Springer, 2012.

[41] T. A. Han, L. M. Pereira, F. C. Santos, and T. Lenearts. Good agreements make good friends. *Nature Scientific Reports*, 3(2695):DOI: 10.1038/srep02695, 2013.

[42] T. A. Han, L. M. Pereira, F. C. Santos, and T. Lenearts. Why Is It So Hard to Say Sorry: The Evolution of Apology with Commitments in the Iterated Prisoner's Dilemma. In *IJCAI 2013*, pages 177–183. AAAI Press, 2013.

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

[44] B. Inhelder and J. Piaget. *The Growth of Logical Thinking from Childhood to Adolescence*. Basic Books, 1958.

[45] A. R. Jonsen and S. Toulmin. *The Abuse of Casuistry: A History of Moral Reasoning*. University of California Press, 1988.

[46] A. Kakas, R. Kowalski, and F. Toni. The role of abduction in logic programming. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5. Oxford U. P., 1998.

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

[48] I. Kant. *Grounding for the Metaphysics of Morals, translated by J. Ellington*. Hackett, 1981.

[49] R. Kowalski. *Computational Logic and Human Thinking: How to be Artificially Intelligent*. Cambridge U. P., 2011.

[50] R. Kowalski and F. Sadri. Abductive logic programming agents with destructive databases. *Annals of Mathematics and Artificial Intelligence*, 62(1):129–158, 2011.

[51] R. Kowalski and F. Sadri. A logic-based framework for reactive systems. In *RuleML 2012*, volume 7438 of *LNCS*, 2012.

[52] D. L. Krebs. *The Origins of Morality – An Evolutionary Account*. Oxford U. P., 2011.

[53] G. Lopes and L. M. Pereira. Prospective programming with ACORDA. In *ESCoR 2006 Workshop*, IJCAR'06, 2006.

[54] G. Lopes and L. M. Pereira. Prospective storytelling agents. In *PADL 2010*, volume 5937 of *LNCS*. Springer, 2010.
[55] R. Mallon and S. Nichols. Rules. In J. M. Doris, editor, *The Moral Psychology Handbook*. Oxford University Press, 2010.
[56] B. M. McLaren. Computational models of ethical reasoning: Challenges, initial steps, and future directions. *IEEE Intelligent Systems*, pages 29–37, 2006.
[57] J. Mikhail. Universal moral grammar: Theory, evidence, and the future. *Trends in Cognitive Sciences*, 11(4):143–152, 2007.
[58] J. O. Newman. Quantifying the standard of proof beyond a reasonable doubt: a comment on three comments. *Law, Probability and Risk*, 5(3-4):267–269, 2006.
[59] J. Pearl. *Causality: Models, Reasoning and Inference*. Cambridge U. P., 2009.
[60] J. Pearl. The algorithmization of counterfactuals. *Annals of Mathematics and Artificial Intelligence*, 61(1):29–39, 2011.
[61] L. M. Pereira. Evolutionary tolerance. In L. Magnani and L. Ping, editors, *PCS 2011*, volume 2 of *SAPERE*, pages 263–287. Springer, 2012.
[62] L. M. Pereira and T. A. Han. Evolution prospection. In *Procs. KES International Conference on Intelligence Decision Technologies*, volume 199, pages 139–150, 2009.
[63] L. M. Pereira and T. A. Han. Intention recognition with evolution prospection and causal bayesian networks. In A. Madureira, J. Ferreira, and Z. Vale, editors, *Computational Intelligence for Engineering Systems: Emergent Applications*, volume 46 of *Intelligent Systems, Control and Automation: Science and Engineering Book Series*, pages 1–33. Springer, 2011.
[64] L. M. Pereira and G. Lopes. Prospective logic agents. *International Journal of Reasoning-based Intelligent Systems*, 1(3/4):200–208, 2009.
[65] L. M. Pereira and A. M. Pinto. Approved models for normal logic programs. In *Procs. 14th Intl. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR'07)*, volume 4790 of *LNAI*. Springer, 2007.
[66] L. M. Pereira and A. M. Pinto. Inspecting side-effects of abduction in logic programs. In M. Balduccini and T. C. Son, editors, *Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning: Essays in honour of Michael Gelfond*, volume 6565 of *LNAI*, pages 148–163. Springer, 2011.
[67] L. M. Pereira and A. Saptawijaya. Moral Decision Making with ACORDA. In *Local Procs. of LPAR 2007*, 2007.
[68] L. M. Pereira and A. Saptawijaya. Modelling Morality with Prospective Logic. In *EPIA 2007*, 2007.
[69] L. M. Pereira and A. Saptawijaya. Modelling Morality with Prospective Logic. *International Journal of Reasoning-based Intelligent Systems*, 1(3/4):209–221, 2009.
[70] L. M. Pereira and A. Saptawijaya. Computational Modelling of Morality. *The Association for Logic Programming Newsletter*, 22(1), 2009.
[71] L. M. Pereira and A. Saptawijaya. Modelling Morality with Prospective Logic. In M. Anderson and S. L. Anderson, editors, *Machine Ethics*, pages 398–421. Cambridge U. P., 2011.
[72] L. M. Pereira and A. Saptawijaya. Abductive logic programming with tabled abduction. In *ICSEA 2012*, pages 548–556. ThinkMind, 2012.

[73] L. M. Pereira, E-A. Dietz, and S. Hölldobler. Abductive reasoning with contextual side-effects. Submitted to KR 2014. Available at `http://centria.di.fct.unl.pt/~lmp/publications/online-papers/context_abd.pdf`, 2013.

[74] L. M. Pereira, T. A. Han, and F. C. Santos. Complex systems of mindful entities – on intention recognition and commitment. In L. Magnani, editor, *Model-Based Reasoning in Science and Technology: Theoretical and Cognitive Issues*, volume 8 of *SAPERE*. Springer, 2013.

[75] D. L. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36 (1):27–47, 1988.

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

[77] I. Rahwan and G. Simari, editors. *Argumentation in Artificial Intelligence*. Springer, 2009.

[78] W. D. Ross. *The Right and the Good*. Oxford University Press, 1930.

[79] A. Saptawijaya and L. M. Pereira. Tabled abduction in logic programs (technical communication of ICLP 2013). *Theory and Practice of Logic Programming, Online Supplement*, 13(4-5), 2013.

[80] A. Saptawijaya and L. M. Pereira. Incremental tabling for query-driven propagation of logic program updates. In *LPAR-19*, volume 8312 of *LNCS ARCoSS*. Springer, 2013.

[81] A. Saptawijaya and L. M. Pereira. Program updating by incremental and answer subsumption tabling. In *LPNMR 2013*, volume 8148 of *LNCS*. Springer, 2013.

[82] A. Saptawijaya and L. M. Pereira. Towards practical tabled abduction usable in decision making. In *KES-IDT 2013*, Frontiers of Artificial Intelligence and Applications (FAIA). IOS Press, 2013.

[83] T. M. Scanlon. Contractualism and utilitarianism. In A. Sen and B. Williams, editors, *Utilitarianism and Beyond*. Cambridge U. P., 1982.

[84] T. M. Scanlon. *What We Owe to Each Other*. Harvard University Press, 1998.

[85] T. M. Scanlon. *Moral Dimensions: Permissibility, Meaning, Blame*. Harvard University Press, 2008.

[86] J. J. Thomson. The trolley problem. *The Yale Law Journal*, 279:1395–1415, 1985.

[87] F. Toni. Argumentative agents. In *Procs. Intl. Multiconference on Computer Science and Information Technology*, volume 5, 2010.

[88] J. van den Hoven and G-J. Lokhorst. Deontic logic and computer-supported computer ethics. *Metaphilosophy*, 33(3):376–386, 2002.

[89] A. van Gelder, K. A. Ross, and J. S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620–650, 1991.

[90] W. Wallach and C. Allen. *Moral Machines: Teaching Robots Right from Wrong*. Oxford U. P., 2009.

[91] V. Wiegel. *SophoLab; Experimental Computational Philosophy*. PhD thesis, Delft University of Technology, 2007.