Moral Reasoning Under Uncertainty

The Anh Han[1⋆], Ari Saptawijaya[1⋆⋆,2], and Luís Moniz Pereira[1]

[1] Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
[2] Fakultas Ilmu Komputer, Universitas Indonesia, Kampus UI Depok 16424, Indonesia
h.anh@campus.fct.unl.pt, ar.saptawijaya@campus.fct.unl.pt, lmp@di.fct.unl.pt

Abstract. We present a Logic Programming framework for moral reasoning under uncertainty. It is enacted by a coherent combination of our two previously implemented systems, Evolution Prospection for decision making, and P-log for probabilistic inference. It allows computing available moral judgments via distinct kinds of prior and post preferences. In introducing various aspects of uncertainty into cases of classical trolley problem moral dilemmas, we show how they may appropriately influence moral judgments, allowing decision makers to opt for different choices, and for these to be externally appraised, even when subject to incomplete evidence, as in courts.

Keywords. Moral Reasoning, Uncertainty Reasoning, Evolution Prospection, Logic Programming, P-log.

1 Introduction

There has been growing interest in understanding morality from a scientific point of view, arising from diverse fields, e.g. primatology [6], cognitive sciences [12,16], neuroscience [26], and other interdisciplinary perspectives [14]. The study of morality has attracted the artificial intelligence community too. Research on modeling moral reasoning computationally have been reported assiduously since the AAAI 2005 Fall Symposium on Machine Ethics [2], and recently in book form [3,27]. We remit to these references and our own previous work [22,23] for detailed background motivation, techniques, and promises of this burgeoning field.

In prior work we exploit features of logic programming, e.g. default negation, abduction and preferences, to model moral reasoning, and employ prospective logic programming with evolution prospection [21,20]. Possible decisions in a moral dilemma are modeled as abducible hypotheses. Abductive solutions (cf. Def. 1 below) are then computationally generated which capture hypothetical decisions and their consequences. The solutions violating integrity constraints, e.g. those containing actions involving intentional killing, are ruled out. Finally, a posteriori preferences single out those generated hypothetical decisions that characterize preferred moral decisions, including the use of utility functions.

⋆ TAH acknowledges the support from FCT-Portugal, grant SFRH/BD/62373/2009.
⋆⋆ AS acknowledges the support from FCT-Portugal, grant SFRH/BD/72795/2010.

This paper aims to show how evolution prospection of conceivable scenarios can be extended to handle moral judgments under uncertainty, by employing a combination of our XSB-Prolog system with P-log [4,11] for computing scenarios' probabilities and utilities. It extends our previous work [22,23] in now further enabling judgmental reasoning under uncertainty concerning the facts, the effects, and even the actual actions performed. For illustration, the newly introduced extensions effectively show in detail how to declaratively model and computationally deal with uncertainty in prototypical classic moral situations known generically as the trolley problem [7].

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 paper is composed by first introducing the trolley problem and its moral dilemmas, in Section 2, followed by a formal description of scenario evolution prospection under uncertainty, in Section 3. Forthwith, illustrative trolley problem examples of decision making under uncertainty are detailed, in Section 4. In Section 5 we adopt instead the external observer's point of view when passing moral judgment on an agent's choices, even if its actions, circumstances, and available evidence are uncertain to a degree. There follows, in Section 6, a discussion of the examples' results in the light of well-known moral principles, and of how these are respected. Some conclusions are then drawn, in Section 7.

This work is neither a proposal for machine incorporated ethics nor an ethics for humans who use machines, as often addressed in the literature.

2 The Trolley Problem and The Principle of Double Effect

The trolley problem presents several moral dilemmas that inquire whether it is permissible to harm one or more individuals for the purpose of saving others. It has the following initial circumstance [12]: "*There is a trolley and its conductor has fainted. The trolley is headed toward five people walking on the track. The banks of the track are so steep that they will not be able to get off the track in time.*" Given this circumstance, there exist several cases of moral dilemmas [16]. The following three are considered here (see Figure 22.2 (1)-(3) of [23] for graphical illustration):

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?

Loop Track. Ned is standing next to a switch that can temporarily turn the trolley onto a side track, without stopping, only to join the main track again. There is a heavy man on the side track. If the trolley hits the man, he will slow down the trolley, giving time

for the five to escape. Ned can throw the switch, killing the man; or he can refrain from doing so, letting the five die. Is it morally permissible for Ned to throw the switch?

The trolley problem suite has been used in tests to assess moral judgments of subjects from demographically diverse populations [12,16]. Interestingly, although all three cases have the same goal, i.e. to save five albeit killing one, subjects come to different judgments on whether the action to reach the goal is permissible or impermissible, i.e. permissible for the Bystander case, but impermissible for the Footbridge and Loop Track cases. As reported by [16], the judgments appear to be widely shared among demographically diverse populations.

Although subjects have difficulty to uncover which moral rules they apply for reasoning in the above cases, their judgments appear to be consistent with the so-called the principle of double effect, enunciated as follows [12]:

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

The key expression is *intended means*, i.e. performing a (harming) action intentionally to attain greater good. Humans must not deliberately be harmed as means to an end.

3 Evolution Prospection under Uncertainty with P-log

We describe how P-log can be integrated into an evolving prospective agent system. We start by briefly recalling the constructs of the Evolution Prospection (EP) system and P-log, to the extent we use them here.

3.1 Evolution Prospection

The implemented EP system has proven useful for decision making [20,24], under different application domains, including Elder Care and Ambient Intelligence in home environment [10,9,24]. The ease in expressing preferences in EP [19,20] enables to take into account agents' preferences with precision. The EP system is implemented on top of ABDUAL, a preliminary implementation of [1], using XSB Prolog [28].

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 used to express that the atom is false by default (closed world assumption). A domain rule in $\mathcal{L}$ is of the form: $A \leftarrow L_1, \ldots, L_t\ (t \geq 0)$, where A is a domain atom and $L_1, \ldots, L_t$ are domain literals. An integrity constraint (IC) in $\mathcal{L}$ is a rule with an empty head. A program P over $\mathcal{L}$ is a set of domain rules and integrity constraints, standing for all their ground instances.

In this paper, we only consider Normal Logic Programs (NLPs), i.e. the head of a rule is an atom or empty. We focus furthermore on abductive logic programs, i.e. NLPs allowing for abducibles – user-specified positive literals without rules, whose truth-value is not fixed. Abducibles instances or their default negations may appear in bodies of rules, like any other literal. They stand for hypotheses, each of which may independently be assumed true, in positive literal or default negation form, as the case may be, in order to produce an abductive solution to a query:

Definition 1 (Abductive Solution). *An abductive solution is a consistent collection of abducible instances or their negations that, when replaced by true everywhere in P,*

affords a model of P (for the specific semantics used on P), which satisfies the query and the ICs – a so-called abductive model.

Active Goals In each cycle of its evolution the agent has a set of active goals or desires. We introduce the $on\_observe/1$ predicate, which we consider as representing active goals or desires that, once triggered by the observations figuring in its rule bodies, cause the agent to attempt their satisfaction by launching all the queries standing for them, or using preferences to select them. The rule for an active goal AG is of the form: $on\_observe(AG) \leftarrow L_1, ..., L_t$ $(t \geq 0)$, where L_1,...,L_t are domain literals. During evolution, an active goal may be triggered by some events, previous commitments or some history-related information.

When starting a cycle, the agent collects its active goals by finding all the $on\_observe(AG)$ that hold under the initial theory without performing any abduction, then finds abductive solutions for their conjunction.

Preferring Abducibles A declared abducible A can be assumed only if it is a considered one, i.e. if it is expected in the given situation, and, moreover, there is no expectation to the contrary. The *A* in the body indicates it is a collected abducible by a search attempt for a query's abductive solution whenever this rule is used.

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

The rules about expectations are domain-specific knowledge contained in the theory of the program, and effectively constrain the abducible hypotheses available in a situation. 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. An *a posteriori* preference 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. Optionally, in the body of the preference rule there can be any Prolog predicate used to quantitatively compare the consequences of the two abductive solutions.

A posteriori preferences between two abductive solutions is enacted by comparing a pair of consequences of each abductive solution. However, more often than not, one abductive solution might have several relevant consequences that contribute to make it either more or less preferred than the other abductive solution it is being compared to. All those relevant consequences are needed to be taken into account for decision making. This is similar to the problem addressed in standard decision theory [8] which is to decide between two actions by evaluating each action's relevant consequences based on an utility function mapping consequences to utilities, typically assumed to be real valued, and a given decision rule. The only difference is that here we need to evaluate consequences of sets of actions which represent the abductive solutions. The decision rule uses the utility to choose among actions, or abductive solutions in this case. Technically, a decision rule maps one abductive solution to a real value based on the values of its consequences. Then, the one having greater utility is preferred to the one having less.

There have been many decision rules studied in the literature. The best-known one is *expected utility maximization*. In general, as for any type of decision rules, it has a set of relevant consequences and a real-valued utility function mapping those consequences to real numbers are given [8]. This decision rule also requires a probability measure that characterizes the decision maker's uncertainty with respect to the consequences of a hypothetical abductive solution. It orders the abductive solutions according to the expected utility [3] of their consequences given the probability measure. Thus, *a posteriori* preferences using *expected utility maximization* decision rule 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 P-log

The P-log system in its original form [4] uses 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 [5] makes the P-log system sometimes computationally redundant. A new implementation of P-log [11], which we deploy in this work, uses the XASP package of XSB Prolog [28] for interfacing with Smodels [18], an answer set solver. The power of ASP allows the representation of both classical and default negation, to produce 2-valued models. Moreover, using XSB as the underlying processing platform enables collecting the relevant

[3] Expected utility of a set of consequences C given a probability measure Pr mapping the consequences to probability values, i.e. $Pr : C \rightarrow [0, 1]$, and an utility function U mapping consequences to real-value utilities, i.e. $U : C \rightarrow \mathbb{R}$, is obtained by the formula: $E(C, Pr, U) = \sum_{X \in C} Pr(X)U(X)$.

abducibles for a query, obtained by need with top-down search. Furthermore, XSB permits to embed arbitrary Prolog code for recursive definitions. Consequently, it allows more expressive queries not supported in the original version, such as meta queries (probabilistic built-in predicates can be used as usual XSB predicates, thus allowing the full power of probabilistic reasoning in XSB) and queries in the form of any XSB predicate expression [11]. In addition, the tabling mechanism of XSB [25] significantly improves the performance of the system.

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 (IC) formed using literals of Σ. An IC is encoded as a XSB rule with the `false` literal in the head.

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

$$random(RandomName, a(\bar{t}), DynamicRange) \texttt{ :- } Body$$

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:

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

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 EP 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 $pr/2$ [11]. 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 EP programs, including active goals, preferences, and integrity constraints. What is more, since P-log [11] allows to code Prolog probabilistic meta-predicates (Prolog predicates that depend on $pr/2$ predicates), we also can directly use probabilistic meta-information in EP programs. We will illustrate those features with several examples below.

4 Moral Reasoning under Uncertainty

We modify the trolley problems to introduce different aspects of uncertainty, and show how that can be modeled in our framework. 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. In the following, we introduce uncertainty into the above mentioned trolley problems. Uncertainty is modeled using probability.

4.1 Revised Bystander Case

The first aspect present in every trolley problem 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.

4.2 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 (IC 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. $PrRV \geq 0.6$), 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 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. [17]. 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 possible verdict, 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 10). 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 0.79, and, by lines 4 and 10, no abductive solution is preferred. This translates into the need for more evidence as the available one is not enough to issue judgment.

6 Discussion

We discuss other aspects of the trolley problems when uncertainty may occur. In the Loop Track Case, we can consider Ned's uncertainty about how probable the man standing on the side track with his back turned can realize that the trolley is going toward him, and get out of the track on time. In this case, one question is whether the action of throwing the switch, which possibly kills the man on the side track, is considered as intentional killing like in the original version. We argue that it should, because Ned's intention is to use the man as a means to stop the trolley, even if he is not sure his intention is achievable. If he threw the switch, he must hope that there would be some chance for the trolley to be slowed down by the man's body. Otherwise he would never do it.

Related to the question of permissibility of actions, let us come back to the Revised Footbridge Case. In the original version, the action of shoving a heavy man from the bridge to stop the trolley in order to save five people is impermissible according two both the double and triple effect principles [13,23] since the action is performed in order to bring about an evil: Ian hopes the shoved man's body will stop the trolley. The situation is clearer if we suppose that the shoved man has some chance of surviving and leaving the track on time, so that the trolley still goes forward onto the five people. In this situation, Ian would hope the shoved man dead. Thus, we can see clearly that Ian's intention is to kill the man to stop the trolley. In the revised version, although Ian shoved the man as an intended means to stop the trolley, his intention is not to kill the man to stop the trolley, but rather on the hope the man survives to throw the switch to stop the trolley. Thus, the action of shoving the man in this version is permissible by the third effect principle, despite still being impermissible by the double effect one. In the examples analysis we rely on the double effect principle, and the triple effect one too.

7 Conclusions

This work is neither a proposal for machine incorporated ethics nor an ethics for humans who use machines, as often addressed in the literature. In contradistinction, it purports to be a proof of principle that our understanding of ethical behavior, even under uncertainty, can in part be computationally modeled and implemented. To be sure, it can (1) be a starting point for imbuing machines with ethics but, beyond that, (2) provide a testing ground for our understanding and experimentation with ethical theories for decision making and moral judgment, and (3) afford us an initial tool to empower the generation of ethical problems for the teaching and explanation of ethical judgment [15].

References

1. 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.
2. M. Anderson and S. L. Anderson. The status of machine ethics: a report from the AAAI Symposium. *Minds and Machines*, 17:1–10, March 2007.
3. M. Anderson and S. L. Anderson. *Machine Ethics*. Cambridge U. P., 2011.

4. C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming*, 9(1):57–144, 2009.
5. L. Castro, T. Swift, and D. S. Warren. XASP: Answer set programming with XSB and Smodels, 2007. http://xsb.sourceforge.net/shadow_site/manual2/node129.html.
6. F. de Waal. *Primates and Philosophers, How Morality Evolved*. Princeton U. P., 2006.
7. P. Foot. The problem of abortion and the doctrine of double effect. *Oxford Review*, 5:5–15, 1967.
8. J.Y. Halpern. *Reasoning about Uncertainty*. MIT Press, 2005.
9. T. A. Han and L. M. Pereira. Collective intention recognition and elder care. In *AAAI 2010 Fall Symposium on Proactive Assistant Agents (PAA 2010)*. AAAI, 2010.
10. T. A. Han and L. M. Pereira. Proactive intention recognition for home ambient intelligence. In *IE Workshop on AI Techniques for Ambient Intelligence, Ambient Intelligence and Smart Environments*, volume 8, pages 91–100. IOS Press, 2010.
11. T. A. Han, C. K. Ramli, and C. V. Damásio. An implementation of extended P-log using XASP. In *Procs. Intl. Conf. on Logic Programming (ICLP'08)*. Springer LNCS 5366, 2008.
12. M. D. Hauser. *Moral Minds, How Nature Designed Our Universal Sense of Right and Wrong*. Little Brown, 2007.
13. F. M. Kamm. *Intricate Ethics: Rights, Responsibilities, and Permissible Harm*. Oxford U. P., 2006.
14. L. D. Katz, editor. *Evolutionary Origins of Morality, Cross-Disciplinary Perspectives*. Imprint Academic, 2002.
15. G. Lopes and L. M. Pereira. Prospective storytelling agents. In *Procs. 12th Intl. Symp. Practical Aspects of Declarative Languages (PADL'10)*. Springer LNCS 5937, 2010. Demo at http://centria.di.fct.unl.pt/∼lmp/publications/slides/padl10/quick_moral_robot.avi.
16. J. Mikhail. Universal moral grammar: Theory, evidence, and the future. *Trends in Cognitive Sciences*, 11(4):143–152, April 2007.
17. 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.
18. I. Niemelä and P. Simons. Probabilistic reasoning with answer sets. In *LPNMR4*, pages 420–429. Springer LNAI 4483, 1997.
19. L. M. Pereira, P. Dell'Acqua, A. M. Pinto, and G. Lopes. Inspecting and preferring abductive models. In *Handbook on Reasoning-based Intelligent Systems*. World Scientific Publishers, *forthcoming*, 2011. http://centria.fct.unl.pt/∼lmp/publications/online-papers/rbis.pdf.
20. L. M. Pereira and T. A. Han. Evolution prospection in decision making. *Intelligent Decision Technologies*, 3(3):157–171, 2009.
21. L. M. Pereira and G. Lopes. Prospective logic agents. In *Procs. 13th Portuguese Intl.Conf. on Artificial Intelligence (EPIA'07)*. Springer LNAI 4784, 2007.
22. L. M. Pereira and A. Saptawijaya. Moral decision making with ACORDA. In *Short Paper LPAR'07*, 2007.
23. L. M. Pereira and A. Saptawijaya. Modelling morality with prospective logic. In *Machine Ethics*, pages 398–421. Cambridge U. P., 2011.
24. L. P. Pereira and T. A. Han. Intention recognition with evolution prospection and causal bayesian networks. In A. Madureira et al., editor, *Computational Intelligence for Engineering Systems: Emergent Applications*, volume 46, pages 1–33. Springer, 2011.
25. T. Swift. Tabling for non-monotonic programming. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):210–240, 1999.
26. L. Tancredi. *Hardwired Behavior, What Neuroscience Reveals about Morality*. Cambridge U. P., 2005.
27. W. Wallach and C. Allen. *Moral Machines: Teaching Robots Right from Wrong*. Oxford U. P., 2009.
28. XSB. The XSB system version 3.2 vol. 2: Libraries, interfaces and packages, March 2009.